DISS. ETH NO. 29746

Connecting the Dots: Combining Points and Lines for more Robust and Accurate Local Features

A thesis submitted to attain the degree of

Doctor of Sciences

(Dr. sc. ETH Zurich)

presented by

Rémi Pautrat

MSc in Computer Science
ETH Zurich, Switzerland

born on 20th of January, 1994

accepted on the recommendation of

Prof. Dr. Marc Pollefeys
Prof. Dr. Viktor Larsson
Prof. Dr. Vincent Lepetit

2023

Acknowledgements

Pursuing a doctorate is a long journey, which would not have been as enjoyable and fruitful without all the persons that accompanied me during these four years.

I would like to start by thanking Marc, my thesis supervisor, for creating the perfect environment to conduct my research and for allowing me to join the Computer Vision and Geometry (CVG) lab in the first place. By accepting me in the team, he offered me a great opportunity to meet a large gathering of bright minds on a daily basis, and to be continuously inspired and motivated to perform cutting-edge science. Marc gave me the freedom and flexibility to do my research at my own pace, with the right people, and with the right resources, so I could not have dreamt of better conditions to achieve a PhD.

Among all my supervisors, Viktor is the one that followed me from the very beginning with the Trimbot project, until the end by accepting to join my doctoral committee, and throughout all my research projects. It is probably from him that I learned the most about how to do good research, but also how to supervise students in both a relaxed and open way. My research would not have gone very far without Martin and Daniel B. either, and I would like to warmly thank them as well for supervising me on most of this journey, and for providing enriching feedback and relevant suggestions throughout our common projects. I would also like to thank Prof. Dr. Vincent Lepetit for accepting to join my defense committee, and for thinking about me whenever he had an interesting review of a journal paper about local features.

Moving on with my colleagues, I would like to thank Shaohui, who embarked with me on the journey to the world of lines more than two years ago, and with whom we have kept collaborating on a number of line-related projects since then. Thank you for your trust and continuous optimism, as well as for the regular chats about our respective lives and for introducing me to the Chinese culture. Thank you Iago for your memorable visit to the CVG lab and for the great summer we spent together at the time, which was only the beginning of a great collaboration over several years. Starting as colleagues, we quickly became friends and this friendship is very precious to me. To all my other co-authors, Juan-Ting, Petr, and Yifan, I want to say thank you for the shared efforts and the long hours spent together to brainstorm ideas, refine our methods, and polish our papers.

I would probably not have joined the field of local features without you, Paul-Edouard, and while our common stay in CVG was only a step in the middle of our longer friendship, becoming colleagues brought a novel dimension to our relationship. You were an inspiration for me in the master studies already, and it did not stop during the PhD. Thank you also

for all the dinners, parties, and adventures spent together, across the craggy mountains of Corsica, through the dense jungles of Costa Rica, and in the wilderness of the Canadian Rockies. To my friends and fellow colleagues of local features and matching, Mihai, Philipp and Prune, I say thank you. Not only for our shared interests and work-related chats, but also for the hiking and mountaineering experiences with Mihai, the skiing and fun card games with Philipp, and all the dinners and parties with Prune. Thank you to Songyou and Silvan for making the lab such a fun and friendly place, and for having the possibility to chat about both serious and casual topics. And also for being great travel buddies, whether it is throughout Sicily and Galicia with Songyou, or across Costa Rica with Silvan. Thanks to Luca too, who shared the burdens of Visual Computing with me, and for the long hours of grading. To all the colleagues of CVG, I want to say a big thank you for making it such an enriching place to work. Thank you Boyang, Cathrin, Daniel T., Denys, Fangjinhua, Francis, Hermann, Ian, Iro, Jonas, Julia, Katarina, Linfei, Lubor, Marcel, Peidong, Sandro, Taein, Xingxing, Yao, Zhaopeng, Zuoyue, and Zuria. Thank you to Ayse too, who organizes the life in the CVG lab behind the scenes, but always in a smooth way and with a constant energy.

During my stay in the lab, I not only learned a lot from my supervisors and colleagues, but I also gained a lot of experience through all the students that I supervised. I would like to thank all of them, in chronological order: Sepideh, Gio, Zheyu, Christian, Florian, Matthias, Senthuran, Szymon, Fabian, Sebastian, Haokai, Stéphane, Thomas, Yidan, Junchi, Alan, Zador, Lei, and all the teams of the 3D Vision and Mixed Reality classes. I hope they will have learned at least as much as I learned from them, and I wish them a successful and blooming career.

Finally, work is only a part of one's life, and an important other part are the friends and family. Drawing up the list of all my dear friends that continuously supported me during these four years and allowed me to escape from work during an evening or weekend would be too long, but they will recognize themselves here. A big thank you to all of you. To my dear Julie who joined me in the very last part of the journey, I want to say thank you for lightening up these last months by your presence and for our daily chats. Last but not least, I would like to thank my beloved parents. Because they contributed the most to make me become who I am now, because they continuously supported me throughout the years, being always there when I needed support and comfort, you have all my thanks and love.

Rémi Pautrat
Zürich, September 2023

Contents

Abstract

Local features are the building block of most multi-view applications such as localization and mapping, 3D reconstruction, and neural rendering, to obtain a highly accurate 3D geometry of a scene. Thus, local features are expected to be accurate, but also to be efficient and versatile to different tasks and conditions. This thesis explores some of the limitations of modern local features and offers solutions, with a particular focus on accuracy and robustness. It starts by reviewing a common issue of the widely used feature points: the trade-off between generalization and discrimination capabilities. A light-weight solution is proposed to select the most adapted invariance among several feature descriptors. Its hierarchical approach offers fine-grained robustness, while remaining compatible with both handcrafted and learned descriptors. In a second part, we tackle the challenges of the emerging line features and demonstrate their benefits in multiple tasks. We first propose a joint neural network to detect and describe line segments in images with high accuracy and robustness. Trained without ground truth labels, our line features are also equipped with a mechanism to handle partial occlusion. We then further improve the line detection with a hybrid approach combining deep learning for its robustness, and handcrafted strategies for their high accuracy. In a final part, we draw the conclusion that points and lines are complementary features and that they should be used in combination. We thus propose a joint deep matcher of points and lines, and show that leveraging the connectivity between all features is highly beneficial for a robust matching. We finally demonstrate how the combination of points and lines can be incorporated in a wide range of geometrical tasks and can boost their downstream performance.

Résumé

Les caractéristiques locales forment la pierre d'angle de la plupart des applications de vision par ordinateur, telles que la localisation et cartographie visuelles, reconstruction 3D, et le rendu neuronal, afin d'obtenir une représentation géométrique en 3D de haute précision. Ainsi, les caractéristiques locales se doivent d'être précises, mais aussi d'être efficientes et polyvalentes pour s'adapter à des tâches et conditions variées. Cette thèse explore certaines des limitations des caractéristiques locales modernes et offre des solutions, avec un intérêt particulier pour la précision et la robustesse. Elle commence par passer en revue un des problèmes notoires des points caractéristiques : le compromis entre capacités de généralisation et de distinction. Une solution peu onéreuse est proposée pour sélectionner l'invariance la plus adaptée parmi plusieurs descripteurs de caractéristiques. Son approche hiérarchique offre une robustesse à plusieurs niveaux, tout en restant compatible avec à la fois les descripteurs traditionnels et neuronaux. Dans une deuxième partie, nous nous attaquons aux problématiques des lignes caractéristiques, alors tout juste émergentes, et nous démontrons leurs avantages parmi plusieurs applications. Nous proposons d'abord un réseau neuronal pour détecter et décrire les segments de droite dans des images avec haute précision et robustesse. Supervisées sans annotations, nos lignes caractéristiques sont aussi équipées d'un mécanisme pour gérer les occlusions partielles. Nous améliorons ensuite la détection de lignes avec une approche hybride combinant l'apprentissage profond pour sa robustesse, et des stratégies traditionnelles pour leur haute précision. Dans la partie finale, nous concluons que les points et les lignes sont des caractéristiques complémentaires et qu'elles devraient être utilisées conjointement. Nous proposons donc un matcheur neuronal couplé pour les points et les lignes, et nous montrons que l'exploitation de la connectivité entre toutes ces caractéristiques est extrêmement bénéfique pour une association robuste entre images. Nous démontrons enfin comment le couplage des points et des lignes peut être incorporé dans une vaste gamme d'applications géométriques et peut stimuler leur performance finale.

Chapter 1

Introduction

1.1 Motivation

1.1.1 The Building Block of 3D Vision

Vision is the primary sensor in nature. In order to obtain a perception in 3 dimensions (3D), evolution has equipped most species with two eyes, to create a small viewpoint variation giving cues about the depth of the environment. Humans are no exception to this, and they can even build an internal 3D representation of the world in their brain while moving and observing the scene through novel views. But how do humans achieve that? A fundamental insight is that they do associations between different views. They recognize objects and shapes, realize that they have already seen them before, and then *match* these objects across different views. Given these associations, it is then a simple geometrical problem (implicitly solved in the brain) to recover the 3D structure of what is being seen.

Thus, humans excel at recognizing objects, especially when these are unique and can be easily discriminated against other objects. Shapes, colors, and textures are the main *features* used by humans to categorize such objects. All these visual characteristics can then be *described* through language and communicated to other humans, to share a common perception of the 3D world. Object-level associations help to understand where we are approximately and to interpret what we are seeing, but to know our exact location, or to be able to reconstruct the world with very fine details, higher precision is needed. In that case, one can rely on smaller structures, such as contours of objects, salient edges, and unique small patterns. These cues are both *local* and *discriminative*, meaning that they only require a low-level understanding of the scene to *detect* them, but at the same time they are easy to recognize. Among the simplest geometrical shapes to represent these patterns, are points and straight lines.

This is what we call *local features* in the remaining of this thesis: small structures, such as points and lines, which are well localized in the image - there is no uncertainty on their position and they are stable across views - and with a discriminative local neighborhood - they can be easily recognized from one view to another. Due to their simplicity and locality, these features are easily detected by machines, and that is why most multi-view applications of computer systems are relying on these local features as their primary building block. Extracting local features from images is a long-standing task in computer vision, and the

next chapter provides an overview of some of their applications.

1.1.2 Applications of Local Features

Similarly as for humans, machines and computers also rely heavily on vision sensors to understand and interpret the world around them. While machines can also leverage depth sensors to directly recover the 3D structure of the world, cameras are still cheaper and more widely available. Therefore, many computer vision applications are leveraging multi-view data to obtain a 3D perception of the world. For most of these applications, extracting and matching local features are the very first steps to get access to multi-view information, as illustrated in Figure 1.1. The next paragraphs describes a few of these applications.

When it comes to matching structures between images, an obvious task is the image stitching and panorama creation. Whether the goal is to assemble several images together to form a bigger picture, or to create an actual panorama from a collection of photographs, local features are at the basis of the stitching process [Milgram, 1975, Brown and Lowe, 2003, Szeliski, 2004, Brown and Lowe, 2007, Barath et al., 2021]. After matching the features between two images, a homography transform is usually computed to model the displacement of each pixels from one image to the other. This assumes either a planar scene or a purely rotational movement, which is the case for the generation of a panorama. Once the pixel-to-pixel transform is known, it becomes easy to stitch the two images. The process can then be generalized to more images to form a panorama.

Local features, and in particular lines, can also provide 3D information from a single image only. This is made possible by the acquisition of vanishing points, the intersection of the principal directions of parallel 3D lines, after projection into the image. In a typical human-made environment, there are three such main directions, classically represented as x, y, and z directions. This is called the Manhattan world assumption [Coughlan and Yuille, 1999]. Given lines in an image and its calibration, one can then associate each line to a vanishing point, and recover the x, y, and z directions, thus yielding the orientation of the camera that took the image [Bazin et al., 2012, Bazin and Pollefeys, 2012, Zhang et al., 2015, Li et al., 2019, Pautrat et al., 2023b].

Another application where local features play a central role is for localization and mapping. A typical use case is in robotics and Augmented Reality (AR) / Virtual Reality (VR) devices that often first build a map of their environment, to be able to later relocalize into it. In the mapping stage, also called Structure-from-Motion (SfM), local features are typically extracted from a collection of images, matched, and then triangulated to lift them to 3D [Heinly et al., 2015, Schonberger and Frahm, 2016]. The map thus consists of a cloud of 3D features. In a later step, the robot or AR / VR device needs to find its position in the map, given its current view. A coarse position is typically obtained through image retrieval techniques [Arandjelović et al., 2016, Sarlin et al., 2019], and local features can be leveraged to re-rank the retrieved images and to find the closest one [Noh et al., 2016, Cao et al., 2020]. Finally, a precise pose is obtained by finding correspondences between the observed features in the current view and the 3D features of the map [Quan and Lan, 1999, Lepetit et al., 2009].

The mapping and localization can also be performed simultaneously and in real time, and

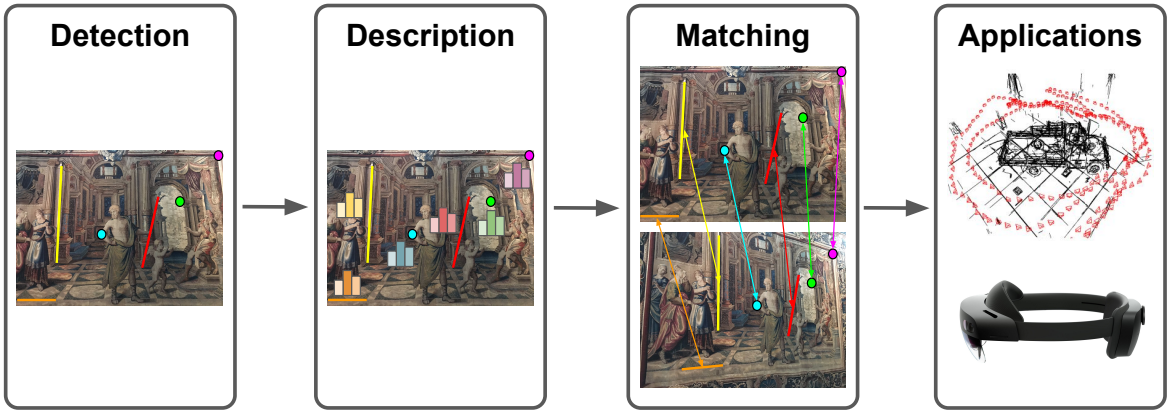


Figure 1.1: Local features extraction pipeline. Features are first detected in images, described with a vector embedding, matched across images, and finally used in multi-view applications.

is referred as Simultaneous Localization and Mapping (SLAM) [Mur-Artal et al., 2015, Qin et al., 2018]. This is especially useful for robots and devices such as drones. Given the real time constraint and the fact that such devices have usually a low compute budget, the extraction of local features must become very efficient. Binary features are then often used, for their low storage and efficient run times [Calonder et al., 2010, Rublee et al., 2011]. Since the data stream is sequential in these scenarios, the features can also be efficiently tracked from one frame to another, avoiding expensive matching.

Furthermore, an SfM reconstruction based on local features does not only provide a map, but it also returns the pose of the cameras that generated the initial set of images. These poses can then be used by other applications requiring posed images, such as Multi-View Stereo (MVS) [Furukawa and Ponce, 2010, Schönberger et al., 2016] and neural rendering [Park et al., 2019, Mildenhall et al., 2020].

Last but not least, all the traditional tasks of computer vision mentioned earlier, SfM, SLAM, MVS, and neural rendering, are used in a multitude of real-life applications: in construction to get a digital model of a building, in the automatic inspection of infrastructures, in home robotics, in autonomous driving, in cinema with the digitization of characters and scenes, etc.

In conclusion, local features are at the basis of a wide range of computer vision applications. Designing efficient, accurate, and robust features is thus essential and can have a large impact in many domains.

1.1.3 Challenges of Local Features

Since local features are the basis of so many applications, they can be employed in a great diversity of environments: indoors and outdoors, with illumination and seasonal changes, on embedded platforms or in the cloud, etc. Thus, designing a generic feature extractor that can work for as many tasks as possible is arduous. We list in the following some of the challenges coming with the design of local features.

- Features must be as repeatable as possible, meaning that if a feature is detected in an image, one wants to find it again in any other views of the same scene, possibly

with different cameras and resolutions. Thus, the detector needs to be covariant to distortions in the image, scale changes, and illumination changes for example. The location of a feature in the image must also be as accurate as possible, so that its position should not move more than a pixel between views for instance. This is a requirement in geometrical tasks requiring high precision like localization and mapping.

- As mentioned above, the descriptor should be as discriminative and unique as possible. Again, one must be able to match descriptors under large viewpoint and appearance changes, including rotation, scale, illumination, and seasonal changes. Robustness to wide-baseline matching and to these strong appearance variations is still a challenge nowadays. On the other hand, being invariant to too many changes can be detrimental (as we will see in Chapter 2), and the degree of invariance required may vary according to the task.

- Detecting many features per image often helps in some applications, as it can provide denser reconstructions and more samples in the Random Sample Consensus (RANSAC) [Fischler and Bolles, 1981a]. However, this makes it harder to distinguish between close-by features and may cause matching errors. Furthermore, having too many features becomes costly in large scenes. As the extraction of local features is usually the initial step in many computer vision pipelines, the inference should also be as fast and light-weight as possible.

- The definition of local features is sometimes ambiguous and can lead to spurious detections. For example, one may want to extract all corners of an image as feature points, but noise and small patterns in a texture can already contain a lot of corners, which will certainly be unstable from one camera to another. Similarly, detecting any number of line segments in an image, including noisy ones, will be detrimental to later applications.

- Line features come with additional challenges. They have a longer extent in the image and are thus less local, so require global context to extract them. Describing a single point is local and easy, in comparison to summarizing the whole area around a line in a single descriptor. Lines also suffer from unstable endpoints, which can often slightly shift along the line direction, or a line can be broken down into smaller sub-segments. While points are either visible or completely occluded, a line can also be partially occluded, making it shorter or longer according to the viewing direction.

- Finally, some features are more fitted to some scenarios than others. For instance, points can sometimes be scarce indoors where there is no texture, while lines may be noisier in natural environments.

All these challenges are still prevalent in modern local features, and we will explore most of them in the next chapters. Each time, we will strive to propose solutions to these issues and we will support our contributions with a wide range of experiments and applications.

1.2 Definition and Evolution of Local Features

We introduce in the following the main concepts and designs of local features that have been developed in the past. Starting with point features (Section 1.2.1), we then introduce line features (Section 1.2.2), before offering a glimpse on the benefits and applications of the combination of both features (Section 1.2.3).

1.2.1 Point Features

1.2.1.1 Detection

Handcrafted detectors. The first step in local feature extraction is usually to detect some specific shapes and to localize them in the image. The simplest form of point features are corners, i.e. the intersection of two lines. Corners are indeed very stable across views, well defined in space, and are very common in most images. The traditional approach first computes the gradient of the image, by convolving it with specific filters such as the Sobel filter [Sobel and Feldman, 1968]. The next step is to find local maxima in the image gradient. A local maximum in one direction usually corresponds to a pixel on a line, and would be too unstable to be a good keypoint. Thus, most methods are searching for maxima in multiple directions, indicating that the pixel is on a corner [Moravec, 1977, Harris and Stephens, 1988, Shi and Tomasi, 1994, Rosten and Drummond, 2006].

But feature points are not limited to corners. Subsequent methods started detecting blobs in images, adding the scale and rotation information to the features. Novel cues have been added to the image gradient to achieve this: the Laplacian of Gaussian (LoG) by computing the Laplacian of an image convolved with Gaussian kernels [Lindeberg, 1998], an approximation of the LoG by computing a Difference of Gaussians (DoG) [Lowe, 2004], the Determinant of the Hessian (DoH) [Bay et al., 2008], or by computing Maximally Stable Extremal Regions (MSER) [Matas et al., 2002]. Some features are also robust to affine transforms, such as the Harris affine detector and its improved version, the Hessian affine region detector [Mikolajczyk and Schmid, 2002]. These features aim at detecting the same regions of the image, even after the image has undergone an affine transform.

Learned detectors. While all these approaches can be considered as "handcrafted", as they are based on specific image filters, the introduction of learning-based methods has ushered in a resurgence of local features. Learned detectors indeed overcame one of the main limitations of traditional detectors: the invariance to weather and lighting changes. Early works processed patches taken from images and ran them through a learned regressor with the Temporally Invariant Learned DEtector (TILDE) [Verdie et al., 2014] or later through a Convolutional Neural Networks (CNN) with the Learned Invariant Feature Transform (LIFT) [Yi et al., 2016].

After the democratization of CNNs, two trends emerged to extract local image features. While both are processing the whole image in a single forward pass and outputting both the point detections and description, they differ in the order of processing. In the *detect-and-describe* approach, a joint backbone is used for both the detector and descriptor, and is

split in two at the end. This allows to share most computations, and can create synergies between the keypoints and their descriptors [DeTone et al., 2018, Revaud et al., 2019]. As there are no obvious ground truth keypoints, the supervision of these methods is a challenge. While SuperPoint [DeTone et al., 2018] used a self-supervised approach by first training a network on a synthetic dataset and then generalized it to real images, following methods minimized the reprojection error of keypoints between views as unsupervised signal [Revaud et al., 2019, Christiansen et al., 2019, Tang et al., 2020, Zhao et al., 2021, Zhao et al., 2023, Wang et al., 2023, Sun et al., 2023]. Other methods included handcrafted filters and scale space images in the backbone [Barroso-Laguna et al., 2019] or supervised the network with reinforcement learning [Tyszkiewicz et al., 2020]. The other approach corresponds to the *describe-then-detect* framework: dense descriptors are first extracted from an image, then a keypoint heatmap is inferred from it, by selecting the most salient pixels in descriptor space. Introduced in D2-Net [Dusmanu et al., 2019], the concept has then been extended in subsequent works [Luo et al., 2020, Tian et al., 2020, Wang et al., 2021].

In this thesis, we use off-the-shelf keypoints among the most famous ones: the Scale-Invariant Feature Transform (SIFT) [Lowe, 2004] for handcrafted keypoints, and SuperPoint [DeTone et al., 2018] for learned ones.

1.2.1.2 Description and Matching

Handcrafted descriptors. In order to match keypoints across images, the traditional approach is to first describe the surrounding region of the point, and to compare the descriptors of all points, before finding the closest descriptor in the other image. Early works leveraged a Histogram of Oriented Gradients (HOG) [William T. Freeman, 1994] to describe a patch around a point. Classifying the gradient orientation and magnitude was later reused in SIFT [Lowe, 2004], probably the most widely used local feature until now. Later works improved the speed while achieving high performance, by maintaining the histograms more efficient [Tola et al., 2010a], or by leveraging Haar wavelet responses [van Drongelen, 2007] in the Speeded Up Robust Features (SURF) [Bay et al., 2008]. Handcrafted local descriptors can be made even faster, with binary descriptors such as the Binary Robust Independent Elementary Features (BRIEF) [Calonder et al., 2010] and its rotated version, the Oriented FAST and Rotated BRIEF (ORB) [Rublee et al., 2011].

Learned descriptors. All the previous methods are describing a patch of the image around each keypoint. Thus, the early deep descriptors naturally followed this scheme by running a CNN on a small image patch to output the descriptor [Yi et al., 2016, Tian et al., 2017, Mishchuk et al., 2017, Ono et al., 2018]. The patch is not restricted to square areas, but can encode spatial transforms, such as affine [Mishkin et al., 2018] and polar [Ebel et al., 2019] ones. The network is often optimized with a triplet loss using heuristics to extract positive and negative patches [Han et al., 2015, Balntas et al., 2016a, Luo et al., 2018, Tian et al., 2019], or by directly maximizing the average precision (AP) [He et al., 2018]. Sparse features also give the possibility to leverage both the visual context of the image and the spatial relationships between the keypoint locations [Luo et al., 2019].

More recently, descriptors extracted densely by CNN architectures from full images have shown both fast inference time and high performance on matching and retrieval tasks, and can jointly detect a heatmap of keypoints [DeTone et al., 2018, Dusmanu et al., 2019, Revaud et al., 2019]. The Repeatable and Reliable Detector and Descriptor (R2D2) [Revaud et al., 2019] adds a reliability branch to the descriptor to keep track of the most informative locations in the image. The supervision is often obtained from reprojecting keypoints from one image to another through homographies [DeTone et al., 2018, Revaud et al., 2019] or using known pose and depth [Dusmanu et al., 2019], but can also come from pose only [Wang et al., 2020]. In the latest works, directly optimizing the negative log likelihood of the matching probabilities has revealed to be high performing [Sarlin et al., 2020a, Zhao et al., 2023].

Matching. After extracting one descriptor per keypoint, the features are usually matched by computing the nearest neighbors across images: the L2 distance of all descriptors from one image to the ones of the other image is computed, and the closest point in descriptor space is selected. Additional checks can further filter out the matches, such as the ratio test [Lowe, 2004] or by only keeping mutually matched keypoints. Outlier matches can also be filtered by handcrafted heuristics [Cavalli et al., 2020] or learned ones [Zhang et al., 2019a]. More recently, a new trend has emerged, with the emergence of deep matchers. SuperGlue [Sarlin et al., 2020a] was the first to propose taking points and their descriptors as input, and to match them within a Graph Neural Network (GNN) to encode context and allow communication between all features. Subsequent works proposed ways to make the inference more efficient [Chen et al., 2021, Shi et al., 2022, Lindenberger et al., 2023]. Chapter 5 will offer another extension of this work to combine the matching of feature points as well as lines.

1.2.2 Line Features

1.2.2.1 Detection

Handcrafted line detectors. Detecting line segments in images is traditionally performed based on the image gradient. Early methods threshold the gradient magnitude to keep only strong edges [Canny, 1986] and search for aligned sets of pixels sharing the same gradient angle. The Line Segment Detector (LSD) [Von Gioi et al., 2008] grows line regions, fits a rectangle to the resulting set of pixels, and finally extracts a line segment. EDLines [Akinlar and Topal, 2011] grows the line regions in one direction only, orthogonal to the image gradient. Several extensions of these methods have been proposed, such as the multi-scale version of LSD, MLSD [Salaün et al., 2016], and the Enhanced Line SEgment Drawing (ELSED) [Suárez et al., 2022], a faster version of EDLines which avoids breaking lines in case of small discontinuities. AG3Line [Zhang et al., 2021b] proposes to actively group the seed points and adds line geometry constraints. Another approach consists in detecting full lines with the Hough transform [Hough, 1962] in a first step, then finding segments within these lines [Elder et al., 2020]. Since all these methods rely on low-level details of the image, they are highly accurate and fast, but lack robustness to noise and low illumination.

Learned line detectors. Deep line detection was first introduced through the task of

wireframe parsing, i.e. estimating the structural lines of a scene [Huang et al., 2018]. Several approaches have been proposed to parameterize and represent the line segments, e.g., with two endpoints [Zhou et al., 2019a], a middle point, direction and length [Dai et al., 2021], attraction fields [Xue et al., 2019, Xue et al., 2020, Xue et al., 2022], center and offset to the endpoints [Huang et al., 2020], graphs [Zhang et al., 2019b, Meng et al., 2020], and transformers [Xu et al., 2021a]. Wireframes can be further improved through a Deep Hough transform [Lin et al., 2020]. All these methods are trained on a single dataset, the Wireframe dataset [Huang et al., 2018], and they are not necessarily suitable for other tasks such as visual localization and SfM.

Generic deep line segment detectors have also been proposed, with a focus on efficiency [Dai et al., 2021, Gu et al., 2022], and can improve visual localization with points and lines [Gao et al., 2021]. However, these methods are again trained solely on the Wireframe dataset and their predicted lines are biased towards structural lines and indoor scenes. The Efficient Line Segment Detector and Descriptor (ELSD) [Zhang et al., 2021a] and the Learnable Line Detector and Descriptor (L2D2) [Abdellali et al., 2021] both propose similar networks, but ELSD is again trained on the Wireframe dataset, while L2D2 uses a novel process to extract a line ground truth from Light Detection and Ranging (LiDAR) scans. Though these approaches are a first step towards unsupervised line detection, they still lack accuracy, as we will wee in the rest of the thesis.

1.2.2.2 Description and Matching

Handcrafted line descriptors. While early line descriptors are based on simple color histograms [Bay et al., 2005], most handcrafted descriptors leverage the image gradient [Wang et al., 2009b, Wang et al., 2009c]. The most common approach is thus to extract a line support region around each line and to summarize gradient information in sub-regions [Wang et al., 2009b, Wang et al., 2009c, Hirose and Saito, 2012, Zhang and Koch, 2013, Verhagen et al., 2014]. The Line Band Descriptor (LBD) [Zhang and Koch, 2013] is the most famous of them, but it still underperforms under large viewpoint and appearance changes.

Learned line descriptors. It is only recently that line description has been tackled with deep learning. One approach is to extract a patch around the line and to compute a low dimensional embedding optimized through a triplet loss, as in the Deep Line Descriptor (DLD) [Lange et al., 2019b] and its improved version, the Wavelet Line Descriptor (WLD) [Lange et al., 2020]. On the other hand, a line descriptor can be considered as a collection of point descriptors, following the idea of Liu *et al.* [Liu et al., 2010]. The Learnable Line Descriptor (LLD) [Vakhitov and Lempitsky, 2019] thus samples and describes multiple points along each line. Designed to be fast and to be used for SLAM, it is however not invariant to rotations and its performance quickly degrades for large viewpoint changes. One of the latest line descriptors was LineTR [Yoon and Kim, 2021], that enriches the line descriptor through self attention layers within and across lines. As we will see in the following, line descriptors are still suffering from multiple issues, to which we propose a few solutions in this thesis.

Line matching. For all the previously mentioned methods, the traditional approach for matching is to compute the nearest neighboring line in descriptor space, similarly as for feature points. Since descriptor-based matching for line segments is generally more difficult than for points, several methods have complemented the descriptor matching with geometric scene information [Li et al., 2016a]: global rotation between images [Zhang and Koch, 2013]; properties of pairs of matched lines like the angle between segments, intersection ratios or projection ratios [Zhang and Koch, 2013, Wang et al., 2009a]; line-point invariants [Fan et al., 2012a]; cross-ratio [Ramalingam et al., 2015a] or consistency with a fundamental matrix estimated from points [Schmid and Zisserman, 1997a, Li et al., 2016c].

1.2.3 Combinations of Points and Lines

Points and lines are often used independently for different applications, but some methods have tried combining them to improve the overall performance. Combinations of a line and multiple points have for example been leveraged in matching to filter out wrong matches [Fan et al., 2012a, Li et al., 2016c]. When connecting matched points by virtual lines, one can then mine for additional point matches along the line to improve wide-baseline matching [Ramalingam et al., 2015a]. Lines can also be used in point-based reconstructions to reduce the drift, thanks to their large extent [Holynski et al., 2020]. Starting from an existing 3D point reconstruction, one can also triangulate lines and add them to the reconstruction, while leveraging the points to solve singularities arising in line triangulation [Liu et al., 2023]. Points and lines can also complement each other to refine the estimation of a camera pose [Gao et al., 2021, Liu et al., 2023]. Finally, there has been a large collection of works exploring SLAM based on points and lines [Zuo et al., 2017, Pumarola et al., 2017, Gomez-Ojeda et al., 2019, Fu et al., 2020b, Lim et al., 2021, Zhou et al., 2022, Ren et al., 2022, Xu et al., 2023]. All these applications often require matching both points and lines, which is classically done independently, and thus, inefficiently. We show in Chapter 5 how to combine points and lines in the matching step already.

1.3 Contributions

As introduced in Section 1.1.3, local features come with multiple challenges. We chose to focus in this thesis on the point and line features, as they are the most basic and versatile kinds of local features. In the following chapters, we will explore some of the limitations of both points and lines in more detail, then show how both types of local features can complement each other to solve some of these issues. Here is a summary of the main contributions of the thesis:

- We show that there is a trade-off between the invariance of feature descriptors to viewpoint and appearance changes, and their power of discrimination. We propose to overcome this issue by automatically evaluating the right amount of invariance necessary for each task, and to select the most suited descriptor for the task in an online fashion.

- We propose the first joint network to detect and describe line segments in images. Our method is self-supervised and does not require hand-labelled ground truth. We show that our predicted lines are repeatable, accurate, and that their descriptor is robust to partial occlusion.
- We introduce a hybrid method fusing handcrafted and learned approaches to improve the accuracy of line detectors, while maintaining a strong robustness to appearance changes. This new line detector can be trained by bootstrapping existing ones and is sufficiently generic, accurate, and robust to be applied in a wide variety of tasks. We also offer a tool to refine any existing lines and to significantly improve their localization accuracy.
- Finally, we propose to jointly match points and lines in a single graph neural network. By leveraging the connectivity between points, we improve the matching of points in texture-less areas, while also strongly improving the performance of line matching compared to previous approaches. We further demonstrate the power of combining points and lines in multiple tasks, and show that these features are complementary and can help each other.

Together with all these contributions, we released the open-source code and pre-trained models for the benefit of the community. The material in this thesis builds upon and originates from the following peer-reviewed publications:

- [Pautrat et al., 2020] **Pautrat, R.**, Larsson, V., Oswald, M. R., and Pollefeys, M. (2020). Online Invariance Selection for Local Feature Descriptors. In *European Conference on Computer Vision (ECCV)*.
- [Pautrat et al., 2021] **Pautrat, R.**, Lin, J.-T., Larsson, V., Oswald, M. R., and Pollefeys, M. (2021). SOLD2: Self-Supervised Occlusion-Aware Line Description and Detection. In *Computer Vision and Pattern Recognition (CVPR)*.
- [Pautrat et al., 2023a] **Pautrat, R.**, Barath, D., Larsson, V., Oswald, M. R., and Pollefeys, M. (2023). DeepLSD: Line Segment Detection and Refinement with Deep Image Gradients. In *Computer Vision and Pattern Recognition (CVPR)*.
- [Pautrat et al., 2023c] **Pautrat, R.**, Suárez, I., Yu, Y., Pollefeys, M., and Larsson, V. (2023). GlueStick: Robust Image Matching by Sticking Points and Lines Together. In *International Conference on Computer Vision (ICCV)*.

Moreover, the author of this thesis also contributed to the following peer-reviewed papers in the course of the doctoral studies:

- [Liu et al., 2023] Liu, S., Yu, Y., **Pautrat, R.**, Pollefeys, M., and Larsson, V. (2023). 3D Line Mapping Revisited. In *Computer Vision and Pattern Recognition (CVPR)*.
- [Pautrat et al., 2023b] **Pautrat, R.**, Liu, S., Hruby, P., Pollefeys, M., and Barath, D. (2023). Vanishing Point Estimation in Uncalibrated Images with Prior Gravity Direction. In *International Conference on Computer Vision (ICCV)*.

1.4 Outline

This thesis is structured in the following way. In a first part, Chapter 2 explores some of the limitations of point features and introduces a method to overcome them. The second part offers a view on the current state of line segment detection and description, explores its limitations, and proposes solutions to these problems. On the one hand, Chapter 3 takes inspiration from the recent progress in feature points learning and applies it to line segments to jointly detect and describe lines in a single network. On the other hand, Chapter 4 goes one step further to improve line detectors and to make them both accurate and robust to image changes. In a third part, Chapter 5 combines both types of features into a single deep matcher, and shows the benefits of such a joint approach. Lastly, Chapter 6 draws conclusions about this work, discusses the complementary nature of points and lines in geometrical tasks, and finally suggests future avenues of research in this topic.

Part I

Limits of Fully Invariant Feature Points

Chapter 2

Online Invariance Selection for Local Feature Descriptors

This chapter explores an important limitation of current local feature descriptors: the trade-off between generalization and discriminative power. We show that too much invariance yields less informative descriptors, and we propose to overcome this limitation with an online selection of the most appropriate invariance, given the context. Our framework consists in a joint learning of multiple local descriptors with different levels of invariance and of meta descriptors encoding the regional variations of an image. The similarity of these meta descriptors across images is used to select the right invariance when matching the local descriptors. Our approach, named Local Invariance Selection at Runtime for Descriptors (LISRD), enables descriptors to adapt to adverse changes in images, while remaining discriminative when invariance is not required. We demonstrate that our method can boost the performance of current descriptors and outperforms state-of-the-art descriptors in several matching tasks, when evaluated on challenging datasets with day-night illumination as well as viewpoint changes. This chapter builds upon the following publication: [Pautrat et al., 2020], and the corresponding code is available at `https://github.com/rpautrat/LISRD`.

2.1 Motivation

In this chapter, we review existing local feature descriptors and show that they suffer from a trade-off between their robustness to image changes, and their capacity to be highly discriminative. Local descriptors are often designed to be invariant to as many transformations as possible, in order to be able to generalize to unknown situations. A vision system relying on feature points to perform autonomous driving is for instance expected to be robust to adverse weather changes (such as sunny, overcast, rainy, foggy, and snowy days), lighting changes (dawn, daylight, dusk, and night), as well as seasonal changes (e.g. from winter to summer). While more invariance can be achieved by relying more on semantics or by designing filters that are invariant under certain transforms, such techniques come with a loss in discriminative power: it becomes harder to distinguish between two very similar patches of an image. Typically, such a descriptor would gain robustness in case of adverse changes, but would lose accuracy when it comes to repeated patterns or semantic classes (e.g. localizing among several similar-looking buildings).

We propose to tackle this issue by automatically deciding at test time which invariance is necessary for the given scenario, and to select the minimum amount of invariance required. Thus, our system is able to automatically adapt to adverse conditions, while remaining highly discriminative the rest of the time.

The rest of this chapter motivates the design of our method (Section 2.2), studies how previous works have tried to gain invariance and to tackle the aforementioned challenge (Section 2.3), introduces our proposed solution to it (Section 2.4), evaluates our method on a wide diversity of benchmarks for local features (Section 2.5), and finally closes on a discussion of this inherent limitation of feature points and of other existing limitations (Section 2.6).

2.2 Introduction

As previously described, sparse feature points are at the root of many computer vision tasks: SfM, SLAM, image retrieval, tracking, etc. They offer a compact representation in terms of memory storage and allow for efficient image matching, and are thus well suited for large-scale applications [Heinly et al., 2015, Schonberger and Frahm, 2016, Schönberger et al., 2017, Sattler et al., 2017]. These features should however be able to cope with real world conditions such as day-night changes [Zhou et al., 2016], seasonal variations [Sattler et al., 2018] and matching across large baselines [Tola et al., 2010b].

To be able to do matching in extreme scenarios, the successive feature detectors and descriptors have become more and more invariant [Mikolajczyk et al., 2005]. The Harris corner detector [Harris and Stephens, 1988] was already invariant to rotations, but not to scale. The SIFT detector and descriptor [Lowe, 2004] was one of the first to achieve invariance with respect to scale, rotation and uniform light changes. More recently, learned descriptors have been able to encode invariance without handcrafting it. On the one hand, patch-based descriptors can become invariant to transforms when estimating the shape of the patch [Yi et al., 2016, Ono et al., 2018, Mishkin et al., 2018, Ebel et al., 2019]. On the other hand,

recent dense descriptors leverage the power of large CNNs to become more general and invariant. Most of them are trained on images with many variations in the training set, either obtained through data augmentation [DeTone et al., 2018], with large databases of challenging images [Dusmanu et al., 2019] or with style transfer [Revaud et al., 2019]. They can also directly encode the invariance in the network itself [Liu et al., 2019]. The general trend in descriptor learning is thus to capture as much invariance as possible.

While feature detectors should generally be invariant to be repeatable under different scenarios [Zhou et al., 2016], the same is not necessarily true for descriptors [Wu et al., 2008]. There is a direct trade-off for descriptors between generalization and discriminative power. More invariance allows a better generalization, but produces descriptors that are less informative. Figure 2.1 shows that the rotation variant descriptor Upright SIFT performs better than its invariant counterpart SIFT when only small rotations are present in the data. We argue that the best level of invariance depends on the situation. As a consequence, this questions the recent trend of jointly learning detector and descriptor: they may have to be dissociated if one does not want the descriptor to be as invariant as the detector.

In this chapter, we focus on learning descriptors only and propose to select at runtime the right invariance given the context. Instead of learning a single generic descriptor, we compute several descriptors with different levels of invariance. We then propose a method to automatically select the most suitable invariance during matching. We achieve this by leveraging the local descriptors to learn meta descriptors that can encode global information about the variations present in the image. At matching time, the local descriptors distances are weighted by the similarity of these meta descriptors to produce a single descriptor distance. Matches based on this distance can then be filtered using standard heuristics such as ratio test, mutual nearest neighbor, or learned matchers.

Overall, our method, named Local Invariance Selection at Runtime for Descriptors (LISRD), brings flexibility and interpretability into the feature description. When some image variations are known to be limited for a given application, one may directly use the most discriminative descriptor among all our learned local descriptors. However, it is usually hard to make such an assumption about the inter-image variations, and LISRD can instead automatically select the best invariance independently for each local region. Hence we are able to distinguish between different levels of variations within the same image (e.g. if half of the image is in the shadow but not the other half) and we show that this can improve the matching capabilities in comparison to using a single descriptor. The meta descriptors formulation is also not restricted to our proposed learned local descriptors, but can be easily generalized to most keypoint detectors and descriptors, as shown in Figure 2.1 where it is applied to SIFT and Upright SIFT. Furthermore, the meta description only adds a small overhead to the current pipelines of keypoint detection and description in terms of runtime and memory consumption, which makes it suitable for real time applications. We can summarize the contributions of this chapter as follows:

- We show how to learn several local descriptors with **multiple variance properties** through a single network, in a similar spirit as in multi-task learning.

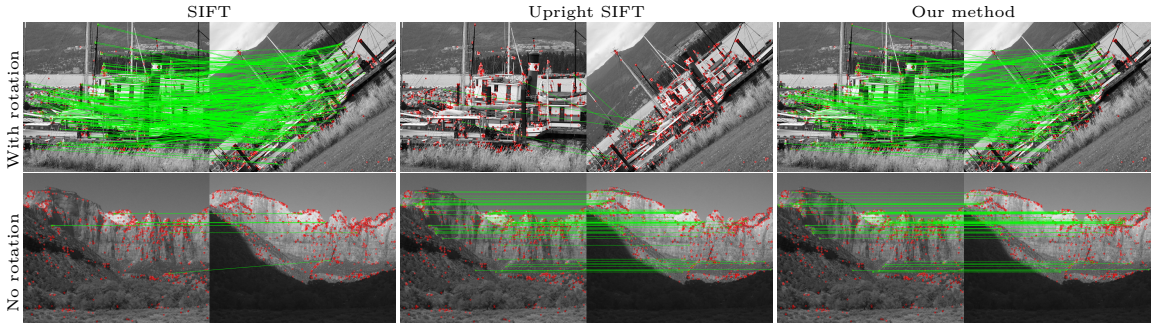


Figure 2.1: Importance of invariance among descriptors. SIFT descriptors (left) perform well on rotated images (top), but are outperformed by Upright SIFT descriptors (middle) when no rotation is present (bottom). We propose a method (right) that automatically selects the proper invariance during matching time.

- We propose a light-weight meta descriptor approach to automatically **select the best invariance** of the local descriptors given the context.
- Our concept of meta descriptor and general approach of invariance selection **can be easily transferred to most feature point detectors and descriptors**, which we demonstrate for learned as well as traditional handcrafted descriptors.

2.3 Related Work

Learned local feature descriptors. Local features have evolved from handcrafted filters to learned ones trained with deep learning. For an overview of recent methods, we refer the reader to Section 1.2.1.

Invariance in feature descriptors. Selecting an online invariance for binary descriptors is the core idea of the Binary Online Learned Descriptor (BOLD) [Balntas et al., 2015], where a subset of the binary tests is chosen at runtime for each image patch to maximize the invariance to small affine transformations. Similarly, the general trend of most recent learned methods is to obtain descriptors as invariant as possible to any image variations. LIFT [Yi et al., 2016] mimics SIFT to achieve rotation invariance by estimating the keypoints, their orientation and finally their descriptor. Invariance to specific geometric changes can be achieved through group convolutions [Cohen and Welling, 2016] by clustering the different geometrical transformations into specific groups [Liu et al., 2019]. However, the usual strategy is to incorporate as much diversity in the training data as possible. Illumination invariance can for example be obtained by training on images with multiple lighting conditions [Kaliroff and Gilboa, 2019]. Photometric and homographic data augmentations also increase robustness to illumination and viewpoint changes [DeTone et al., 2018]. Similarly, R2D2 [Revaud et al., 2019] improves the robustness to day-night changes by synthesizing night images with style transfer and also to viewpoint changes by leveraging flow between close-by images [Revaud et al., 2015]. Methods like D2-Net [Dusmanu et al., 2019] leverage a large database of images with multiple conditions and non planar viewpoint changes thanks to SfM data [Li and Snavely, 2018]. In this work, we adopt a mixture of the previously mentioned methods, namely the same synthesized night images as in [Revaud et al., 2019], homographic augmentation, and

training on datasets with multiple illumination changes [Murmann et al., 2019].

Multi-task learning in description and matching tasks. Using a single network to achieve multiple and related tasks in feature description and matching is not new. Jointly learning the detector and descriptor [DeTone et al., 2018, Dusmanu et al., 2019, Revaud et al., 2019] is already multi-task learning that makes the descriptors more discriminative at the predicted keypoint locations. The Hierarchical Feature Network (HF-Net) [Sarlin et al., 2019] unifies the detection of feature points, local and also global descriptors for image retrieval using multi-task distillation with a teacher network. Methods such as SuperGlue [Sarlin et al., 2020b] and ContextDesc [Luo et al., 2019] can leverage both visual and geometric context in their descriptors in order to get a more consistent matching between images. UR2KID [Yang et al., 2020] bypasses the need of keypoint supervision during training and directly optimizes the descriptors jointly for local matching and image retrieval. In our approach, multiple descriptors are also learned in parallel, but instead of differing in their scope, they differ in their level of invariance. Furthermore, unlike previous hierarchical global-to-local approaches, our method relies on local descriptors first and leverages global information only to refine the local matching.

2.4 Method

Our approach to select the most relevant variance for local feature descriptors consists in two steps. First, we design a network to learn several dense descriptors, each with a different type of invariance (see Section 2.4.1). Second, we propose a strategy in Section 2.4.2 to determine the best invariance to use when matching the local descriptors. Figure 2.2 provides an overview of the full architecture.

2.4.1 Disentangling Invariance for Local Descriptors

Many properties of an image have an influence on descriptors, but disentangling all of them would be intractable. We focus here on two factors known to have a large impact on descriptors performance: rotation and illumination. Our framework can however be generalized to other kinds of variations, for instance scaling. Since each of the two factors can either be variant or invariant, there are four possible combinations of variance with respect to illumination and rotation. We show in the following that the variant versions of descriptors are more discriminative since they are more specialized, while the invariant ones are trading the discriminative power for better generalization capabilities.

Network architecture. Our network is inspired by SuperPoint [DeTone et al., 2018], with slight modifications. It takes RGB images as input, computes semi-dense features with a shared backbone of convolutions and is then divided into 4 heads predicting a semi-dense descriptor each, one per combination of variance, as shown in Figure 2.2. Since most computations are redundant between the 4 local descriptors, the shared backbone reduces the number of weights in the network and offers an inference time competitive with the current learned descriptors.

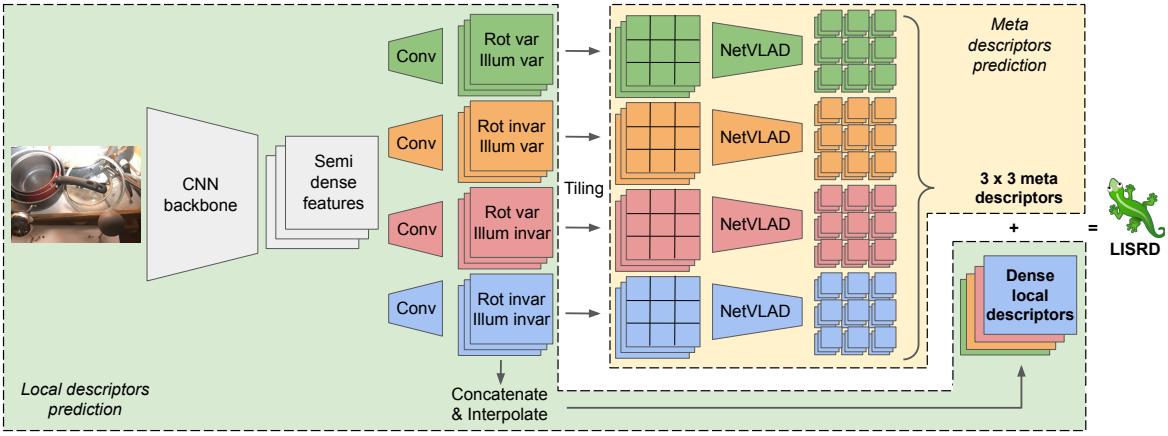


Figure 2.2: Overview of our network architecture. Our network computes four local dense descriptors with diverse invariances and aggregates them through a Network for Vector of Locally Aggregated Descriptors (NetVLAD) layer [Arandjelović et al., 2016] to obtain a regional description of the variations of the image.

Dataset preparation. The training dataset is composed of triplets of images. The first one, the *anchor image* I^A, is taken from a large database of real images. The *variant image* I^V is a warped version of the anchor by a homography without rotation and with equal illumination to train variant descriptors. Finally, the *invariant image* I^I used for invariant descriptors is also related to the anchor by a homography, but its orientation and illumination can differ from the anchor.

Training losses. The local descriptors are trained using variants of the margin triplet ranking loss [Balntas et al., 2016b, Mishchuk et al., 2017], depending on whether the descriptor should be invariant or not to the variations present in I^I. The dense descriptors are first sampled on selected keypoints of the images, they are L2-normalized and the losses are computed on the resulting set of feature descriptors. Since we focus on descriptors only, we use SIFT keypoints during training to propagate the gradient in informative areas of the image only. Any kind of keypoint can be used at inference time nonetheless, as demonstrated in Section 2.5.5.

Formally, given two images I^a and I^b related by a homography $\mathcal{H}$ and n keypoints $\mathbf{x}^a_{1..n}$ in image I^a, we warp each point to image I^b using the homography: $\mathbf{x}^b_{1..n} = \mathcal{H}(\mathbf{x}^a_{1..n})$. This yields a set of n correspondences between the two images, where we can extract the descriptors from each dense descriptor map: $\mathbf{d}^a_{1..n}$ and $\mathbf{d}^b_{1..n}$. Let us define a generic triplet loss $L_T(I^a, I^b, \text{dist})$ between I^a and I^b, given a descriptor distance $\text{dist}(\mathbf{x}^a, \mathbf{x}^b)$. The triplet loss first enforces a correct correspondence $(\mathbf{x}^a_i, \mathbf{x}^b_i)$ to be close in descriptor space through a positive distance

$$p_i = \text{dist}(\mathbf{x}^a_i, \mathbf{x}^b_i) \ . \tag{2.1}$$

Additionally, the triplet loss increases the negative distance n_i between $\mathbf{x}^a_i$ and the closest point in I^b which is at least at a distance T from the correct match $\mathbf{x}^b_i$. This distance is computed symmetrically across the two images and the minimum is kept:

$$n_i = \min(\text{dist}(\mathbf{x}^a_i, \mathbf{x}^b_{n_b(i)}), \text{dist}(\mathbf{x}^b_i, \mathbf{x}^a_{n_a(i)})) \ , \tag{2.2}$$

with $n_b(i) = \arg\min_{j\in[1,n]}(\text{dist}(\mathbf{x}_i^a, \mathbf{x}_j^b))$ s.t. $||\mathbf{x}_i^a - \mathbf{x}_j^b||_2 > T$, and similarly for $n_a(i)$. Given a margin M, the triplet margin loss is then defined as

$$L_T(I^a, I^b, \text{dist}) = \frac{1}{n}\sum_{i=1}^{n} \max(M + p_i^2 - n_i^2, 0) \ . \tag{2.3}$$

In our case, the loss L_I for invariant descriptors is an instance of this generic triplet loss between the anchor image I^A and the invariant image I^I, for the L2 descriptor distance:

$$L_I = L_T(I^A, I^I, ||\mathbf{d}^A - \mathbf{d}^I||_2) \ . \tag{2.4}$$

The loss L_V for variant descriptors is based on the full triplet of images: I^A, I^I and I^V. It enforces variant descriptors to be different between the anchor and the invariant image, while preserving similarity between the anchor and the variant image. Its positive loss is the distance in descriptor space of positive matches between I^A and I^V, and similarly for the negative distance between I^A and I^I:

$$L_V = \frac{1}{n}\sum_{i=1}^{n} \max(fM + ||\mathbf{d}_i^A - \mathbf{d}_i^V||_2^2 - ||\mathbf{d}_i^A - \mathbf{d}_i^I||_2^2, 0) \ , \tag{2.5}$$

where f is a factor controlling at which point the anchor and the invariant images are different. For rotation changes, $f = \min(1, \frac{\theta_I}{\theta_{max}})$, where θ_I is the absolute angle of rotation between the anchor and the invariant image and θ_{max} is a hyper-parameter representing the threshold beyond which the two images should be considered different. This threshold ensures that only large rotations are penalized by the loss. It is hard to quantify the difference in illumination between two real images, so we set $f = 1$ when the illumination differs between the anchor and invariant image.

When a descriptor d in the set $\mathcal{D}$ of descriptors is supposed to be invariant to all changes (illumination and/or rotation) between I^A and I^I, we use L_I. Otherwise, L_V is used. We define $L_{I/V}(d)$ as the selected loss and the total loss for local descriptors as

$$L_l = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} L_{I/V}(d) \ . \tag{2.6}$$

2.4.2 Online Selection of the Best Invariance

Given the local descriptors of the previous section, this section explores how to pick the most relevant invariance when matching images. Since it would be costly to recompute and compare the image variations for every pair of images to be matched, we propose to rely solely on the information contained in the descriptors to perform the selection. A naive approach would be to separately compute the similarity of the different local descriptors and to pick the most similar ones. However, the invariance selection would gain by having more context than the information of a single local descriptor and should be consistent with neighboring descriptors. Therefore, we propose to extract regional descriptors from the local ones and to use them to guide the invariance selection.

The local descriptors are thus gathered in neighboring areas through a NetVLAD layer [Arandjelović et al., 2016] to get a meta descriptor sharing the same kind of invariance as the subset of local descriptors, but with more context than a single local descriptor. Thus, having similar meta descriptors means sharing the same level of variations. The neighboring areas are created by tiling the image into a $c \times c$ grid and computing a meta descriptor for each tile. Hence, we get four meta descriptors per tile, which are then L2 normalized.

When matching the local descriptors of a tile, the four similarities between the meta descriptors are computed with a scalar product and we can rank the four local descriptors according to these similarities. Instead of making a hard choice by taking only the closest local descriptor, we use a soft assignment. A softmax operation is applied to the four similarities, to get four weights summing to one. These weights are then used to compute the distance between the local descriptors as shown in Figure 2.3. More precisely, suppose that we want to compute the distance in descriptor space between point $\mathbf{x}^a$ in image I^a and point $\mathbf{x}^b$ in image I^b. Point $\mathbf{x}^a$ is associated with 4 local descriptors $\mathbf{d}^a_{1..4}$ and 4 meta descriptors $\mathbf{m}^a_{1..4}$ corresponding to the region where $\mathbf{x}^a$ lies, and similarly for $\mathbf{x}^b$. Then the final descriptor distance between $\mathbf{x}^a$ and $\mathbf{x}^b$ is

$$\text{dist}(\mathbf{x}^a, \mathbf{x}^b) = \sum_{i=1}^{4} \frac{\exp\left((\mathbf{m}^a_i)^\intercal \cdot \mathbf{m}^b_i\right)}{\sum_{j=1}^{4} \exp\left((\mathbf{m}^a_j)^\intercal \cdot \mathbf{m}^b_j\right)} ||\mathbf{d}^a_i - \mathbf{d}^b_i||_2 \ . \tag{2.7}$$

Thus, the similarity of the meta descriptors acts as a weighting of the local descriptors distances and can put a stronger emphasis on one specific variance when the corresponding meta descriptors have a high similarity. Matching is then performed with this descriptor distance, and can easily be refined with ratio test [Lowe, 2004] or mutual nearest neighbor.

Training loss. The 4 NetVLAD layers are trained with a weak supervision based on another instance of the triplet loss L_T between I^A and I^I with the distance defined above:

$$L_m = L_T(I^A, I^I, \text{dist}) \ . \tag{2.8}$$

Thanks to this weak supervision, there is no need to explicitly supervise the meta descriptors, which would require knowing the amount of rotation and illumination for every tile in the image. The total loss of the network is finally a combination of the local and meta descriptors,

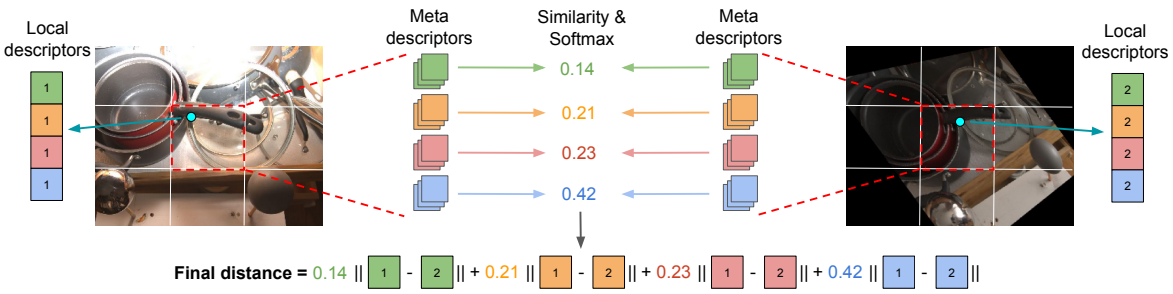


Figure 2.3: The LISRD descriptor distance between two points is the sum of the four local descriptors distances, weighted by the similarity of the meta descriptors.

weighted by a factor λ:

$$L = L_l + \lambda L_m \ . \tag{2.9}$$

2.4.3 Training Details

Datasets. To train descriptors with different levels of variance in terms of rotation and illumination, datasets presenting all possible combinations of changes are needed. Control over the amount of changes is also required in order to know which loss between L_I and L_V should be used for each descriptor. We use in total four datasets to accomplish that. Illumination variations are obtained through the multi illumination dataset in the wild [Murmann et al., 2019] and the style transferred night images of the Aachen day dataset [Revaud et al., 2019]. Both offer pairs of images with fixed viewpoint and different illuminations. Images with fixed illumination come from the Microsoft Common Objects in COntext (MS-COCO) dataset [Lin et al., 2014] and the day flow images from the Aachen dataset [Revaud et al., 2019]. For all datasets except the latter, the images are augmented with random homographies containing translation, scaling, rotation and perspective distortion, similarly as in [DeTone et al., 2018]. For the day images of Aachen, the flow is used to create the correspondences and we consider that these images contain only small rotations and no major illumination changes. Overall, there is an equal distribution of images with and without illumination changes, and of rotated and non rotated images.

Implementation details. We describe here the details of our architecture. The backbone network, inspired by the VGG16 [Simonyan and Zisserman, 2014], is composed of successive 3×3 convolutional layers with channel size 64-64-64-64-128-128-256-256. Each conv layer is followed by a Rectified Linear Unit (ReLU) activation and batch normalization. Every two layers, a 2×2 average pooling with stride 2 is applied to reduce the spatial resolution by 2. For an image of size $H \times W \times 3$, the output feature map will have a size of $H/8 \times W/8 \times 256$. The local descriptor heads are all composed of the following operations: 3×3 conv of channel size 256 - ReLU - Batch Norm - 1×1 conv of channel size 128. The final dimension of each local descriptor is thus $H/8 \times W/8 \times 128$, and each concatenated descriptor is 512-dimensional. The semi-dense descriptors can then be bilinearly interpolated to the locations of any keypoint. Note that in order to achieve a better robustness to scale changes, one can also detect the keypoints and describe them at multiple image resolutions and aggregate the results in the original image resolution, similarly as in [Dusmanu et al., 2019] and [Revaud et al., 2019]. The NetVLAD layers consists in 8 clusters of 128-dimensional descriptors, hence a meta descriptor size of 1024. We used $c \times c = 3 \times 3$ tiles per image.

The network is trained on RGB images resized to 240×320 with the following hyper-parameters: distance threshold $T = 8$, $\theta_{max} = \frac{\pi}{4}$, margin $M = 1$, loss factor $\lambda = 1$. It comprises roughly 3.7M parameters, which are optimized with the Adam solver [Kingma and Ba, 2014] (learning rate $= 0.001$ and $\beta = (0.9, 0.999)$). In practice, the local descriptors are pre-trained first and then fine-tuned by an end-to-end training with the meta descriptors. At test time, a single forward pass on a GeForce RTX 2080 Ti with 480×640 images takes 6ms on average.

2.5 Experimental Results

We present here experiments validating the relevance of our method. Section 2.5.2 highlights the importance of learning different invariances, validates the proposed approach with an ablation study, and shows that LISRD can be extended to other descriptors such as SIFT and Upright SIFT. LISRD is then compared to the state of the art on a benchmark homography dataset (Section 2.5.3), on a challenging dataset with diverse conditions where the presence or lack of invariance is essential (Section 2.5.4), with different keypoint detectors (Section 2.5.5), across a full day (Section 2.5.6), and on a visual localization task in the real world (Section 2.5.7).

2.5.1 Metrics

Since we want to compare the performance of the descriptors only, all the following metrics are computed on SIFT keypoints if not stated otherwise. The metrics are computed on pairs of images resized to 480×640 and related by a known homography. Resizing is performed by upscaling/downscaling the images to have each edge greater or equal respectively to 480 and 640, and a central crop is applied to get the target resolution. We keep a maximum of 1000 points among the keypoints shared between the two views and matches are obtained after mutual nearest neighbor filtering.

Homography estimation. We follow the procedure of [DeTone et al., 2018] to compute a homography estimation score. Given a pair of images, RANSAC is used to fit a homography between the clouds of matched keypoints. The score is obtained by warping the four corners of the first image $\hat{c}_{1\dots4}$ with the predicted homography and comparing their distance to the same points $c_{1\dots4}$ warped by the ground truth homography. The homography is considered as correct when the average distance is below a threshold ϵ, which is set to 3 pixels in all experiments: $\text{HEstim} = \frac{1}{4}\sum_{i=1}^{4} ||\hat{c}_i - c_i||_2 \leq \epsilon$.

Precision. Precision (also known as mean matching accuracy) is the percentage of correct matches over all the predicted matches [Dusmanu et al., 2019, Revaud et al., 2019]. We use by default a threshold of 3 pixels to consider a match to be correct.

Recall. Recall is the ratio of correctly predicted matches over the total number of ground truth matches, where a ground truth correspondence is the *closest* point within an error threshold of 3 pixels. A predicted match with the *second closest* point but still within the correct threshold is considered as incorrect.

2.5.2 Method Validation

Impact of the different invariances. One can check the validity of our approach by comparing the 4 local descriptors. We use the HPatches dataset [Balntas et al., 2017], which is standard in descriptor evaluation. It is composed of 116 sequences of 5 pairs of images, with either viewpoint changes (given by a known homography) or illumination changes with fixed viewpoint. Figure 2.4 shows the comparison between the 4 descriptors in terms of precision. On viewpoint changes, the illumination variant descriptors are superior as the

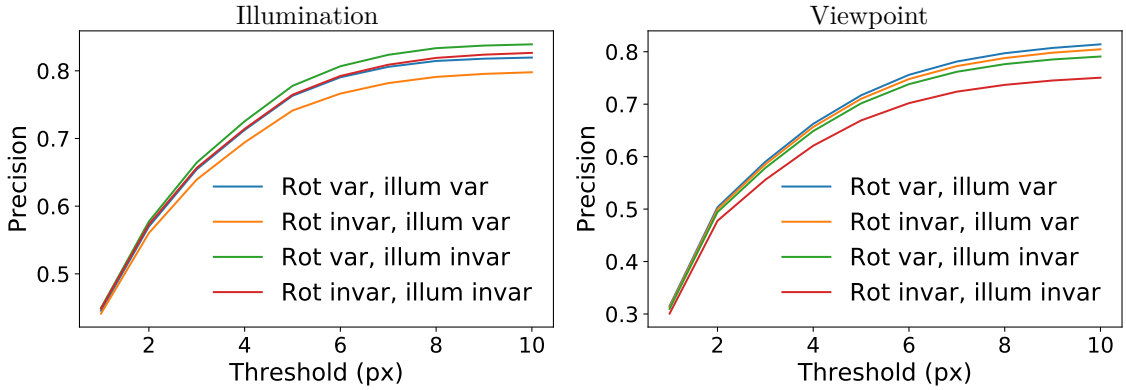


Figure 2.4: Precision on HPatches of the 4 local descriptors. Variant ones are better when invariance is not needed (e.g. rotation for the illumination dataset).

lighting is fixed in these images and they are thus more discriminative. Since HPatches contains few rotations, there is no significant difference in terms of rotation invariance and being rotation variant brings a small advantage on average. The precision on illumination changes shows that the best performing descriptors are the illumination invariant ones and that being rotation variant helps since the viewpoint is fixed. Thus, there is no descriptor outperforming the others in all cases, and our hypothesis that variant descriptors are more discriminative than invariant ones is validated.

Ablation study. To confirm the benefit of our online selection of invariance and choice of parameters, we compare LISRD on homography estimation on the HPatches dataset [Balntas et al., 2017] with other selection methods of the local descriptors as well as with variants of our approach (Table 2.1). *Best of the 4* computes the metrics for the 4 local descriptors separately and picks the best score. *Greedy* computes the pairwise distances of all points for each local descriptor and greedily chooses the local descriptor with smallest distance for each pair of points. *Hard assignment* selects the local descriptor that maximizes the meta descriptor similarity, instead of choosing a soft assignment as in our proposed method. *No tiling* and 5×5 *tiles* are variants of our method with no tiling or with 5×5 tiles for the meta descriptors. Finally, *Single desc* is a descriptor trained with exactly the same architecture as ours, but with the 4 local descriptors concatenated and trained with invariance in both illumination and rotation.

On the full HPatches dataset, *Best of the 4* corresponds to the descriptor invariant to both illumination and rotation, as both changes are present. However, our selection method can still leverage the other descriptors: for example an illumination variant descriptor for the viewpoint part. The disparity between LISRD and *Greedy* and *Hard assignment* highlights the added value of the meta descriptors, and shows that a soft assignment can better leverage the 4 descriptors at the same time. Finally, the comparison with *Single desc* confirms our hypothesis that disentangling the types of invariance is beneficial compared to learning a single invariant descriptor with the same number of weights.

Generalization to other descriptors. LISRD can be easily generalized to other kinds of descriptors, and not only to our proposed learned local descriptors. We demonstrate this by applying our approach to the duo of local descriptors SIFT and Upright SIFT – SIFT without

	HEstim
Best of the 4	0.778
Greedy	0.774
Hard assignment	0.762
No tiling	0.752
5×5 tiles	0.773
Single desc	0.766
LISRD (ours)	**0.784**

Table 2.1: Ablation study of LISRD on the HPatches dataset [Balntas et al., 2017].

rotation invariance, as presented in Figure 2.1. Instead of having four local descriptors, there are only two of them, one invariant to rotation and one variant, and similarly for the meta descriptors. Our method is evaluated against SIFT and Upright SIFT on the viewpoint part of HPatches. This dataset contains indeed sequences with no rotation, where Upright SIFT performs better, and other sequences with strong rotations, where SIFT takes over. Figure 2.5 shows that our method can effectively leverage both SIFT and Upright SIFT and outperforms the two.

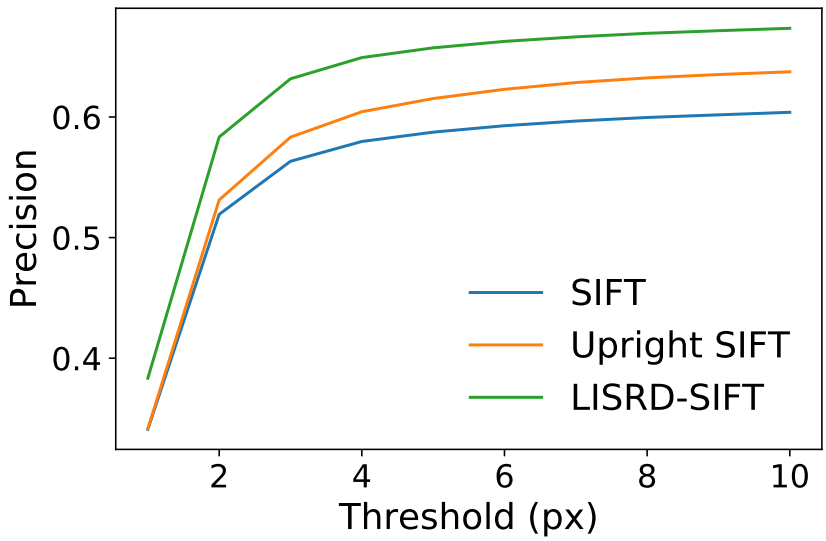


Figure 2.5: Variants of SIFT vs. our method fusing them (LISRD-SIFT). Precision is computed on HPatches viewpoint. Our method can leverage invariance to rotation only when necessary, and thus outperforms the other baselines.

2.5.3 Descriptor Evaluation on HPatches

This section compares the performance of LISRD against state-of-the-art local descriptors on the benchmark dataset HPatches [Balntas et al., 2017]. Since our approach requires global context from full images, we cannot run it on the patch level dataset. We use the full sequences of images instead, similarly as in [DeTone et al., 2018, Dusmanu et al., 2019, Revaud et al., 2019]. We consider the following baselines: Root SIFT with the default Kornia[1] implementation; HardNet [Mishchuk et al., 2017] (trained on the PS-dataset [Mitra et al., 2018]), SOSNet [Tian et al., 2019] (trained on the Liberty dataset of UBC Phototour [Brown et al.,

[1] https://kornia.github.io/

2010]), SuperPoint (SP) [DeTone et al., 2018], D2-Net [Dusmanu et al., 2019], R2D2 [Revaud et al., 2019] and the Group Invariant Feature Transform (GIFT) [Liu et al., 2019]. We use the authors implementation. Since we want to evaluate the descriptors only, SIFT keypoints are detected in the images and for each method, we extract the local descriptors at these locations. For Root SIFT, HardNet and SOSNet, we sample 32×32 patches at each SIFT keypoint and rotate them according to the SIFT orientation. As we want to evaluate the impact of rotation and illumination invariance only, we use single scale implementations for all methods[2]. Our method could however be made scale invariant using similar multi-scale approaches as in [Dusmanu et al., 2019, Revaud et al., 2019].

The results are summarized in Table 2.2. Overall, LISRD ranks among the two best methods in precision and recall. The possibility to leverage rotation variant descriptors on the fixed pairs of the illumination part and to alternatively select the right level of lighting invariance given the amount of illumination changes probably explains our superior performance on the illumination part. Note the comparison with SuperPoint, whose architecture and training procedure are very similar to LISRD, and where our method displays better results in all metrics, thus showing the gain of our approach. The weaker results in homography estimation can be explained by a limitation of our method. Since our meta descriptors have a very coarse spatial resolution (3×3 grid), if one of them fails to pick the right invariance, this will impact all the matches of its region. Thus, the correct matches predicted by LISRD can in that case become very concentrated in a specific part of the image, which makes the homography estimation with RANSAC less accurate. This issue could be avoided with a finer tiling of the meta descriptors, but at the price of a reduced global context.

2.5.4 Evaluation in Challenging and Cross-Modal Situations

The HPatches dataset offers a fair benchmark, but is limited to only few rotations and medium illumination changes. Our approach is designed to be used in a variety of scenarios and with changing conditions, so that all our local descriptors can be leveraged. In order to evaluate our method on such a versatile task, we designed a new benchmark dataset, based on the Day-Night Image Matching (DNIM) dataset [Zhou et al., 2016]. This dataset is composed of sequences of images of a fixed camera taking pictures at regular time intervals and across day and night, with a total of 1722 images. For each sequence, the image with timestamp

		Root SIFT	HardNet	SOSNet	SP	D2-Net	R2D2	GIFT	Ours
HP Illum	HEstim	0.898	0.884	<u>0.919</u>	0.877	0.818	0.916	**0.923**	0.884
	Precision	0.554	0.574	0.591	0.629	0.650	**0.666**	0.573	<u>0.665</u>
	Recall	0.431	0.483	0.519	0.565	0.564	<u>0.580</u>	0.521	**0.655**
HP View	HEstim	0.644	0.688	**0.742**	0.651	0.553	0.627	<u>0.715</u>	0.688
	Precision	0.515	0.582	<u>0.598</u>	0.595	0.564	0.550	0.552	**0.626**
	Recall	0.350	0.422	<u>0.448</u>	0.446	0.382	0.371	0.429	**0.495**

Table 2.2: Comparison to the state of the art on HPatches [Balntas et al., 2017]. Homography estimation, precision and recall are computed for error thresholds of 3 pixels. The best score is in bold and the second best one is underlined.

[2]In the case of GIFT, which is both rotation and scale invariant, we sample images with scale 1 to make it rotation invariant only.

closest to noon is taken as day reference and the image closest to midnight as night reference. We create two benchmarks, where the images of each sequence are paired with either the day reference or the night one. We then synthetically warp the pairs with the same homography sampling scheme as in [DeTone et al., 2018] with an equal distribution of homographies with and without rotations. More details about this dataset, called the Rotated Day-Night Image Matching (RDNIM) dataset, can be found in Appendix A.

Table 2.3 and Figure 2.6 show the evaluation with the state-of-the-art descriptors, using SuperPoint keypoints. LISRD can adapt its invariance to illumination and rotations to alternatively select the most relevant descriptor and it outperforms the other methods by a large margin both in terms of precision and recall.

		Root SIFT	HardNet	SOSNet	SP	D2-Net	R2D2	GIFT	Ours
Day ref	HEstim	0.121	**0.199**	0.178	0.146	0.094	0.170	0.187	0.198
	Precision	0.188	0.232	0.228	0.195	0.195	0.175	0.152	**0.291**
	Recall	0.112	0.194	0.203	0.178	0.117	0.162	0.133	**0.317**
Night ref	HEstim	0.141	**0.262**	0.211	0.182	0.145	0.196	0.241	**0.262**
	Precision	0.238	0.366	0.297	0.264	0.259	0.237	0.236	**0.371**
	Recall	0.164	0.323	0.269	0.255	0.182	0.216	0.209	**0.384**

Table 2.3: Evaluation on a use case where invariance selection matters. Homography estimation, precision and recall are computed with SuperPoint keypoints on a dataset with day-night changes and various levels of rotation. Selecting the relevant variant or invariant descriptors boosts the precision and recall of our method compared to the previous state-of-the-art methods.

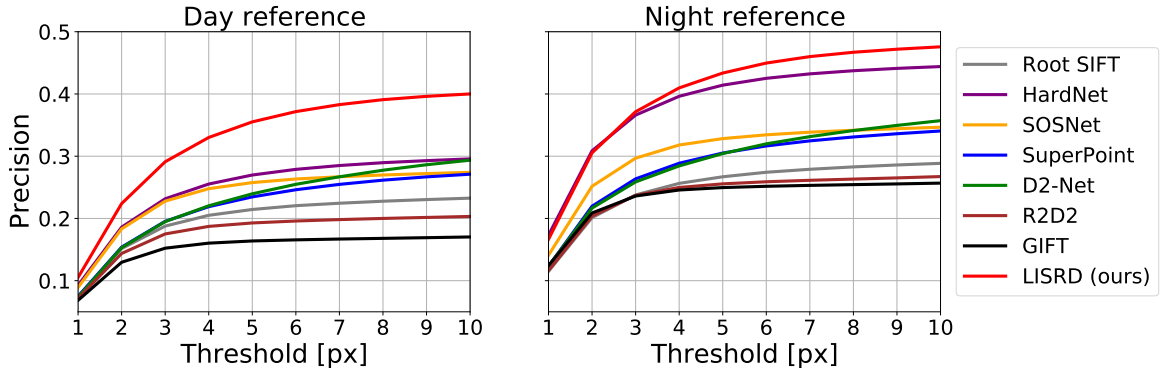


Figure 2.6: Precision curves on the DNIM dataset [Zhou et al., 2016] augmented with rotations. LISRD leverages its variant and more discriminative descriptors whenever possible and is thus more accurate than the state-of-the-art descriptors for all pixel error thresholds.

2.5.5 Generalization to Different Keypoint Detectors

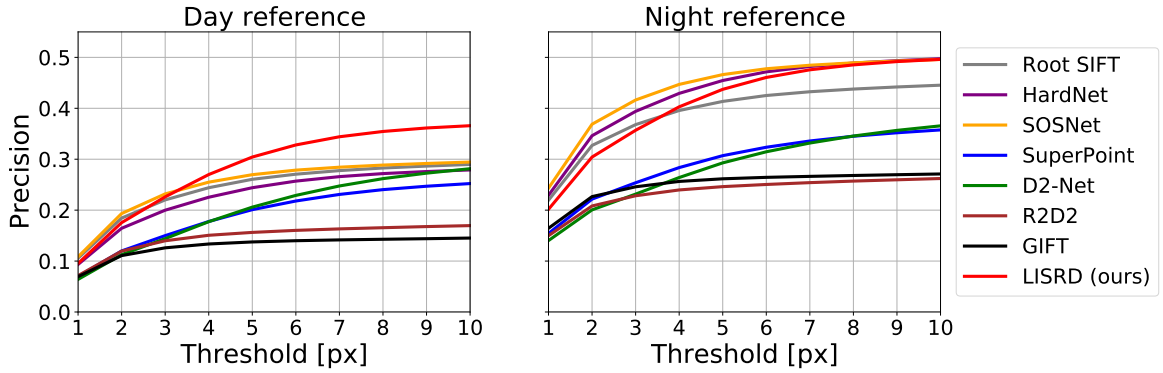
LISRD was trained using SIFT keypoints [Lowe, 2004], but it can be used at test time with any other keypoints (KP). We demonstrate this by providing additional comparisons to the state of the art on the RDNIM dataset, using SIFT, SuperPoint [DeTone et al., 2018] and R2D2 [Revaud et al., 2019] keypoints. Table 2.4 presents the evaluation with homography estimation, precision and recall for an error threshold of 3 pixels and Figure 2.7 shows the precision curves at multiple error thresholds. Overall, LISRD is competitive with the state of the art (HardNet and SOSNet) on SIFT keypoints and ranks first with learned keypoints

on most metrics. Note the improved homography estimation score with learned keypoints, probably because these keypoints are well spread across the image and the limitation of LISRD mentioned in Section 2.5.3 is curtailed. Indeed, this limitation was due to RANSAC producing bad estimates when the invariance selection failed in some regions and all the matches became concentrated in a small area. This phenomenon is less likely to happen when the keypoints are covering the whole image, and LISRD is thus able to get a more accurate homography estimation. Note that for each keypoint, the associated descriptor is not necessarily performing better, except for R2D2 that gets a slight improvement in precision when evaluated on their own keypoints. This is due to the reliability map used during their training, which makes their descriptors more discriminative at their keypoint locations.

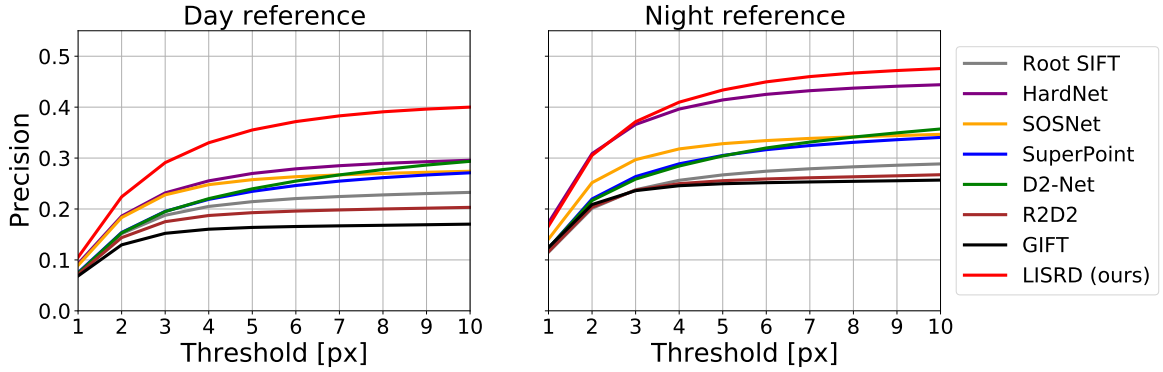
As a feature direction of work, LISRD would benefit from learning its own keypoints with an additional head. This single head would predict invariant keypoints trained on images with multiple lightings and rotations and could be used with all descriptors - whether they are variant or not. This would ensure a better correlation between the keypoints and their descriptors and offer a faster prediction, instead of predicting separately keypoints and descriptors as is currently the case.

			Root SIFT	HardNet	SOSNet	SP	D2-Net	R2D2	GIFT	LISRD (Ours)
SIFT KP	Day ref	HEstim	0.166	0.170	**0.215**	0.084	0.057	0.121	0.145	0.127
		Precision	0.220	0.200	**0.232**	0.150	0.144	0.140	0.126	0.226
		Recall	0.113	0.155	0.197	0.114	0.081	0.107	0.108	**0.212**
	Night ref	HEstim	0.255	0.278	**0.307**	0.156	0.118	0.167	0.215	0.204
		Precision	0.368	0.394	**0.416**	0.254	0.231	0.228	0.246	0.357
		Recall	0.212	0.288	**0.316**	0.183	0.135	0.162	0.183	0.284
SP KP	Day ref	HEstim	0.121	**0.199**	0.178	0.146	0.094	0.170	0.187	0.198
		Precision	0.188	0.232	0.228	0.195	0.195	0.175	0.152	**0.291**
		Recall	0.112	0.194	0.203	0.178	0.117	0.162	0.133	**0.317**
	Night ref	HEstim	0.141	**0.262**	0.211	0.182	0.145	0.196	0.241	**0.262**
		Precision	0.238	0.366	0.297	0.264	0.259	0.237	0.236	**0.371**
		Recall	0.164	0.323	0.269	0.255	0.182	0.216	0.209	**0.384**
R2D2 KP	Day ref	HEstim	0.107	0.187	0.181	0.140	0.093	0.135	0.157	**0.193**
		Precision	0.162	0.201	0.192	0.166	0.171	0.210	0.118	**0.237**
		Recall	0.093	0.167	0.172	0.168	0.101	0.076	0.102	**0.290**
	Night ref	HEstim	0.135	**0.196**	0.168	0.145	0.101	0.132	0.183	0.189
		Precision	0.200	**0.302**	0.244	0.221	0.221	0.241	0.166	0.291
		Recall	0.132	0.260	0.215	0.230	0.149	0.110	0.147	**0.335**

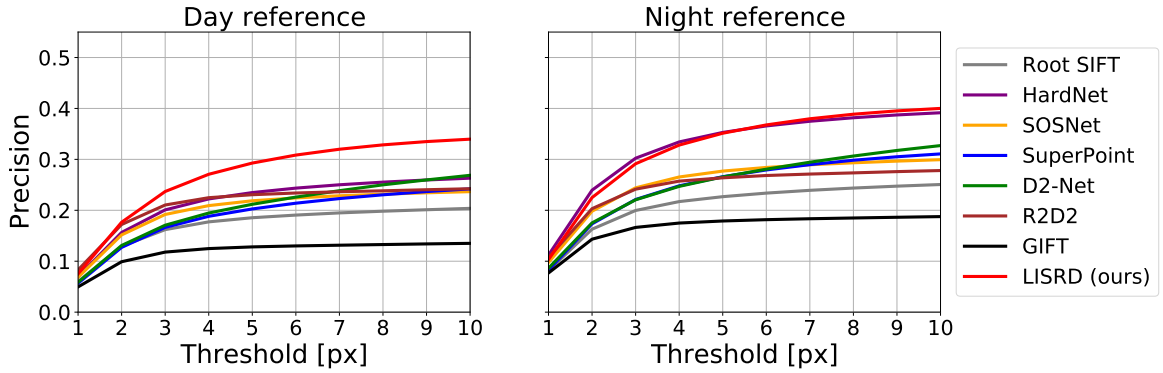
Table 2.4: Evaluation with SIFT [Lowe, 2004], SuperPoint (SP) [DeTone et al., 2018] and R2D2 [Revaud et al., 2019] keypoints on the RDNIM dataset. Homography estimation, precision and recall are computed for an error threshold of 3 pixels. LISRD is not restricted to the SIFT keypoints that were used during its training, but can be generalized to any keypoints (KP). The best score is in bold and the second best one is underlined.



(a) SIFT keypoints.



(b) SuperPoint keypoints.



(c) R2D2 keypoints.

Figure 2.7: Precision curves with SIFT [Lowe, 2004], SuperPoint [DeTone et al., 2018] and R2D2 [Revaud et al., 2019] keypoints on the RDNIM dataset. The discriminative power of LISRD descriptors is not limited to SIFT keypoint locations, but also shows a high precision compared to the state of the art on other keypoints.

2.5.6 Evaluation Across a Full Day

The evaluation on the RDNIM dataset shows the global performance across a mix of day-day and day-night, or night-night and night-day images. But it is also interesting to study the performance at various times during the day. Figure 2.8 displays the precision and recall curves on the RDNIM dataset along a full day. For every image in the second pair, we extract the hour at which the picture was taken from the timestamp, and round it to the closest integer. For each hour, the precision and recall are then computed and averaged across all images corresponding to this time and these averaged numbers are then plotted over the twenty-four hours of a day. We naturally get two peak curves, one centered at noon for the pairs with the day reference and the other centered at midnight for the night reference. LISRD is overall better than the other descriptors and, interestingly, the largest improvements come from the time intervals with day-night illumination changes. Thus, LISRD leverages its illumination variant and more discriminative descriptors when the timestamp of both images of the pair are close, and it switches to the invariant and more general ones when the images are taken at different times of the day.

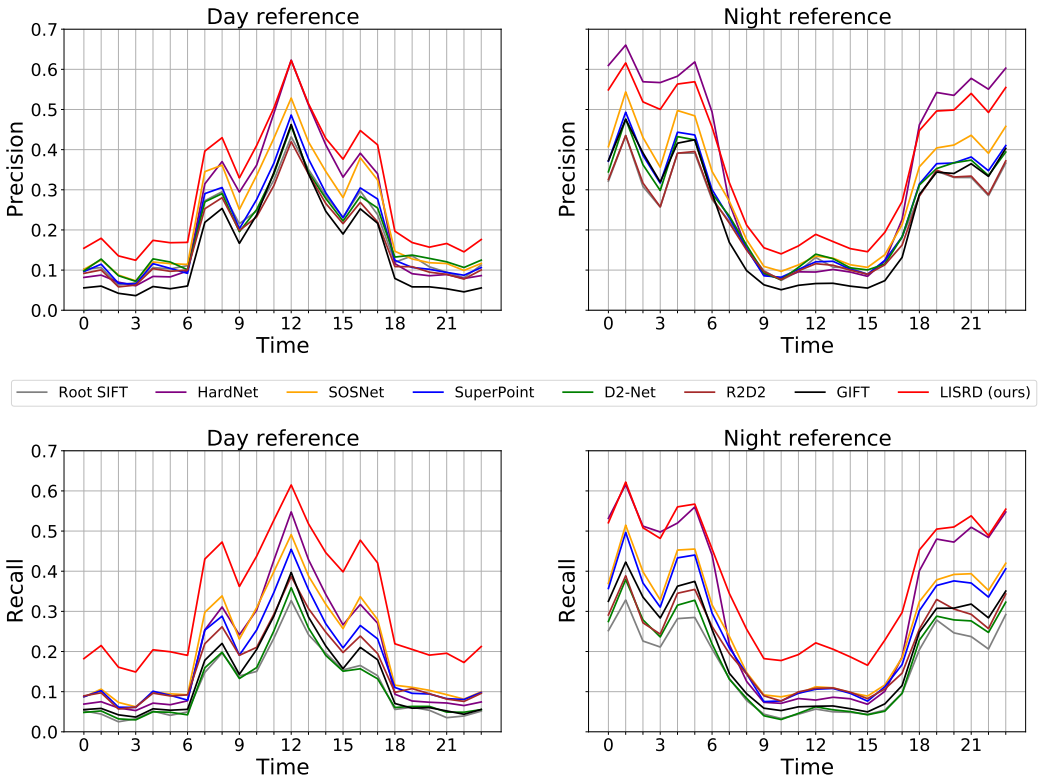


Figure 2.8: Precision and recall across the day. Precision and recall are computed on the RDNIM dataset with SuperPoint keypoints and an error threshold of 3 pixels. They are averaged for each hour of the day, based on the timestamp of the second image. The performance gradually degrades when the timestamp of the second image moves away from the reference time (noon for the day reference and midnight for the night reference). For close timestamps, LISRD leverages its illumination variant descriptors, but switches to the invariant ones when the timestamps differ too much. Thus, LISRD remains competitive with state-of-the-art descriptors for close timestamps and outperforms them when significant illumination changes are present.

2.5.7 Application to Localization in Challenging Conditions

A typical application of image matching including adverse conditions such as strong illumination changes and wide baselines is the visual localization task. We evaluate our method on the local feature challenge of the Conference on Vision and Pattern Recognition (CVPR) 2019 based on the Aachen Day-Night dataset [Sattler et al., 2018]. The goal is to localize 98 night time query images as accurately as possible, given 20 day images per query with known camera pose. As the keypoint quality is essential in this task, we compare our method with other descriptors for various types of keypoints: SIFT, SuperPoint and D2-Net multi-scale (MS). The numbers for the baseline methods are taken from the benchmark on the official website[3]. The results in Table 2.5 show that our method is not limited to SIFT keypoints and can effectively improve the performance of local descriptors in challenging conditions. Note in particular the improvement over SuperPoint, which shares a similar architecture as ours.

Error threshold	SIFT KP		SuperPoint KP		D2-Net KP	
	Up-Root SIFT	Ours	SuperPoint	Ours	D2-Net (MS)	Ours (MS)
$0.5m, 2°$	54.1	**72.4**	73.5	**78.6**	67.3	**73.5**
$1m, 5°$	66.3	**82.7**	79.6	**86.7**	87.8	**88.8**
$5m, 10°$	75.5	**94.9**	88.8	**98.0**	**100.0**	99.0

Table 2.5: Visual localization performance on the Aachen Day-Night dataset [Sattler et al., 2018]. We report the percentage of correctly localized queries for various distance and orientation error thresholds for SIFT, SuperPoint and D2-Net MS. Our method shows a good generalization when evaluated on different keypoints and can improve the original descriptor performance.

2.5.8 Qualitative Examples

We provide additional qualitative examples of matches based on SIFT keypoints and LISRD descriptors. All matches are filtered with mutual nearest neighbor, followed by a homography fitting with RANSAC [Fischler and Bolles, 1981a]. Figure 2.9 brings a visualization of the invariance selection, with a different color for each kind of invariance that was selected. Since the selection is in practice based on a soft weighting, we only show the color of the learned descriptors that contributed the most in the matching decision. These sample images show that in some situations, a single invariance is sufficient for the full image, but in other cases multiple invariances can be leveraged within the same image, demonstrating the need of tiled meta descriptors. This is for example useful when the overall illumination is constant in a pair of images, but one part of an image (e.g. a building) is overexposed or in the shadow.

Finally, Figure 2.10 displays a selection of matches in challenging scenarios, for example with day-night and/or with strongly rotated images.

[3] https://www.visuallocalization.net/

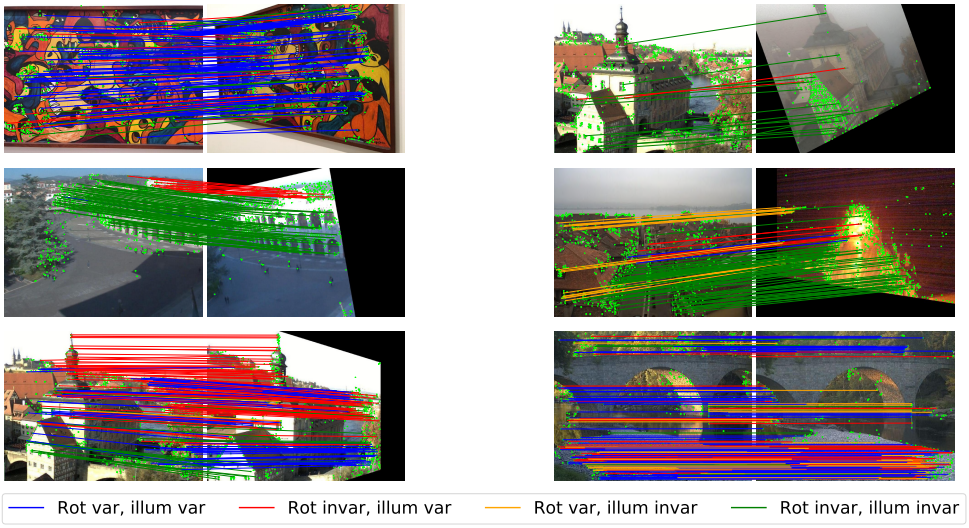


Figure 2.9: Visualization of the selected invariance. Matches of SIFT keypoints with LISRD descriptors are filtered with mutual nearest neighbor and RANSAC [Fischler and Bolles, 1981a]. Since our method uses a soft weighting of the invariances, each color corresponds only to the invariance that contributed the most to validate the match. First line: one type of invariance predominates in the whole image. Second line: two invariances are relevant in the same image (on the left, rotation invariance is needed, but the building in the top right corner is overexposed in both images and illumination invariance is not needed in this area ; on the right, illumination invariance is needed, but the image is upright on the left side, while the distortion creates a rotation on the central part and rotation invariance becomes necessary). Third line: multiple different invariances can be leveraged in the same image (on the left image, the right part of the image is mainly upright and with constant illumination, while the house in the lower left corner is overexposed and rotated, hence the fully invariant descriptor is selected ; on the right image, most of the selected descriptors are rotation variant as the viewpoint is fixed, but the left pier of the bridge has a constant illumination while the right pier has a different illumination and the illumination invariant descriptor predominates).

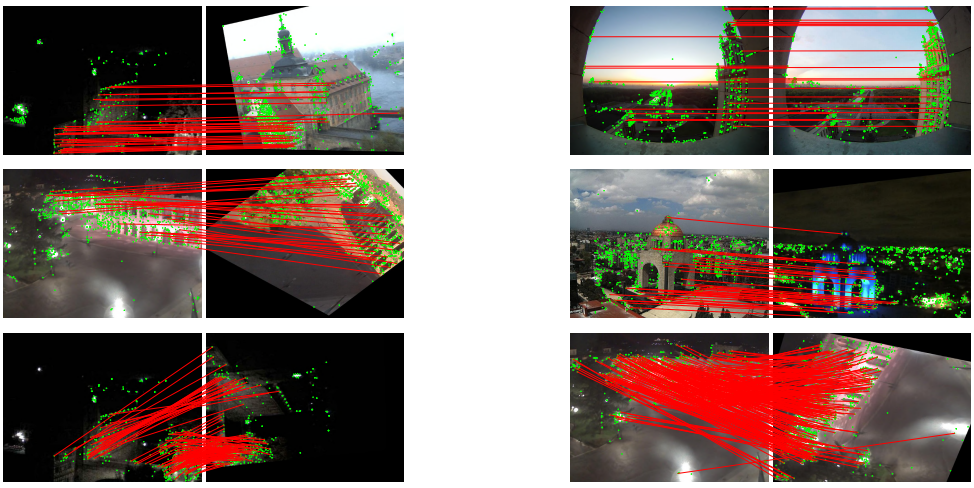


Figure 2.10: Matches in challenging situations. SIFT keypoints are detected, matched with the LISRD descriptors, and mutual nearest neighbor and RANSAC [Fischler and Bolles, 1981a] are used to filter out wrong matches. A single red color is used for all the inlier matches, regardless of the chosen invariance. Matches based on LISRD descriptors are able to handle strong illumination changes such as day-night, inter-image illumination variations in day-day and night-night pairs, and small as well as strong rotations.

2.6 Discussion

Summary. In this chapter, we showed through multiple experiments that local feature descriptors are losing their discriminative power when they reach too much invariance. To tackle this trade-off between invariance and discriminativeness, we presented a novel approach to learn local feature descriptors able to adapt to multiple variations in images, while remaining discriminative. We unified the learning of several local descriptors with multiple levels of invariance and of meta descriptors leveraging regional context to guide the local descriptors matching.

Limitations and future works. While restricted to illumination and rotation invariance, our framework can be generalized to more variations, at the cost of an exponentially growing number of descriptors, however. A future direction of work would be to reduce the amount of redundancy between each descriptor by enforcing a stronger disentanglement separating each factors of variation. Instead of relying on the similarity of descriptor embeddings to select the right invariance, one could also explore the promising attention mechanisms instead. Finally, since our approach is able to enforce different levels of invariance, one could add another head to our network to predict invariant keypoints, while keeping discriminative descriptors. This would potentially solve the current issue in joint learning of invariant detectors and descriptors. Overall, this work is only a first step towards disentangled descriptors. Separating the types of invariances paves the way to a full disentanglement of the factors of variations of images and could lead to more flexible and interpretable local descriptors.

Closing remarks. The trade-off between invariance and discriminative power is only one of the limitations of current local feature points. A severe drawback of points is also that 2D points have no structure and lack relational information. Furthermore, feature points are very scarce in texture-less areas, making applications fail in these scenarios. We propose in the next chapters solutions to tackle these limiting issues.

Part II

Line Features: a Promising Alternative to Points

Chapter 3

SOLD2: Self-supervised Occlusion-aware Line Description and Detection

While points are the de facto sparse representations in geometrical tasks, they can be complemented by higher-order structures to tackle some of their limitations. In this chapter, we show that line features represent a promising complement to points for multi-view tasks. Lines are indeed well-defined by the image gradient, frequently appear even in poorly textured areas and offer robust structural cues. Taking inspiration from the recent progress on point features, we show how to handle the additional challenges brought by lines compared to points, and we introduce the first joint detection and description of line segments in a single deep network. Thanks to a self-supervised training, our method does not require any annotated line labels and can therefore generalize to any dataset. Our detector offers repeatable and accurate localization of line segments in images, departing from the wireframe parsing approach explored in previous works. Leveraging the recent progresses in descriptor learning, our proposed line descriptor is highly discriminative, while remaining robust to viewpoint changes and occlusions. We evaluate our approach against previous line detection and description methods on several multi-view datasets and display higher repeatability, localization accuracy and matching metrics. This chapter builds upon the following publication: [Pautrat et al., 2021], and the corresponding code is available at `https://github.com/cvg/SOLD2`.

3.1 Motivation

The previous chapter explored a limitation of existing local feature points and how to overcome it. However, feature points suffer from additional drawbacks, such as the lack of structure and relational information, as well as their scarcity on texture-less surfaces. We propose in this chapter to connect points and to form line segments in order to overcome these additional challenges. Lines are often present in human-made environments, even in low-textured areas. The connectivity of lines through their endpoints also offers a rich information to better understand the 3D structure of a scene and to match images. Furthermore, the large extent of lines in a scene helps to solve known issues of point-based applications, such as drift in SLAM and distortions in the 3D reconstructions of SfM.

On the other hand, line features come with additional challenges, which will be listed in the following section. Current line detectors and descriptors are also lagging behind the state-of-the-art feature points both in terms of accuracy and matching ability. Besides, line detection and description have so far been explored as two separate tasks, which is inefficient. We thus propose to revisit the task of line segment detection and description in a single deep network, and show how to handle current limitations of lines to make them suitable for vision tasks.

This chapter starts with an introduction to the problem (Section 3.2), lists the previous works for line detection, description and matching (Section 3.3), introduces our proposed method (Section 3.4), evaluates it against existing line features (Section 3.5), and finally discusses the results and studies the remaining limitations to solve (Section 3.6).

3.2 Introduction

As seen in the previous chapter, feature points are at the core of many computer vision tasks, due to their compact and robust representation. Yet, the world is composed of higher-level geometric structures which are semantically more meaningful than points. Among these structures, lines can offer many benefits compared to points. Lines are widespread and frequent in the world, especially in man-made environments, and are still present in poorly textured areas. In contrast to points, they have a natural orientation, and a collection of lines provide strong geometric clues about the structure of a scene [Weng et al., 1992, Taylor and Kriegman, 1995, Holynski et al., 2020]. As such, lines represent good features for 3D geometric tasks.

Previous methods detecting line segments in images often relied on image gradient information and handcrafted filters [Von Gioi et al., 2008, Akinlar and Topal, 2011]. Recently, deep learning has also enabled robust and real-time line detection [Huang et al., 2020]. Most learned line detectors are however tackling a closely related task: wireframe parsing, which aims at inferring the structured layout of a scene based on line segments and their connectivity [Huang et al., 2018, Zhou et al., 2019a, Xue et al., 2020, Zhou et al., 2019b]. These structures provide strong geometric cues, in particular for man-made environments. Yet, these methods have not been optimized for repeatability across images, a vital feature

for multi-view tasks, and their training requires ground truth lines that are cumbersome to manually label [Huang et al., 2018].

The traditional way to match geometric structures across images is to use feature descriptors. Yet, line descriptors face several challenges: line segments can be partially occluded, their endpoints may not be well localized, the scale of the area to describe around each line fluctuates a lot, and it can be severely deformed under perspective and distortion changes [Schmid and Zisserman, 1997b]. Early line descriptors focused on extracting a support region around each line and on computing gradient statistics on it [Wang et al., 2009c, Zhang and Koch, 2013]. More recently, motivated by the success of learned point descriptors [DeTone et al., 2018, Dusmanu et al., 2019, Revaud et al., 2019], a few deep line descriptors have been proposed [Lange et al., 2019b, Vakhitov and Lempitsky, 2019, Lange et al., 2020]. However, they are not designed to handle line occlusion and remain sensitive to poorly localized endpoints.

In this chapter, we propose to jointly learn the detection and description of line segments. To this end, we introduce a self-supervised network, inspired by the Line Convolutional Neural Network (LCNN) [Zhou et al., 2019a] and SuperPoint [DeTone et al., 2018], that can be trained on any image dataset without any labels. Pretrained on a synthetic dataset, our method is then generalized to real images. Our line detection aims at maximizing the line repeatability and at being as accurate as possible to allow its use in geometric estimation tasks. The learned descriptor is designed to be robust to occlusions (as illustrated in Figure 3.1), while remaining as discriminative as the current learned point descriptors. To do so, we introduce a novel line matching based on dynamic programming and inspired by sequence alignment in genetics [Needleman and Wunsch, 1970] and classical stereo matching [Dieny et al., 2011]. Thus, our Self-supervised Occlusion-aware Line Description and Detection (SOLD2) offers a generic pipeline that aims at bridging the gap with the learned feature points. Overall, the contributions of this chapter can be summarized as follows:

- We propose the **first deep network for joint line segment detection and description**.
- We show how to **self-supervise** our network for line detection, allowing **training on any dataset** of real images.
- Our line matching procedure is **robust to occlusion** and achieves **state-of-the-art results** on image matching tasks.

3.3 Related work

Line features Detecting edges and line segments in images is a long-standing quest in computer vision, and has been revisited many times, including with deep learning in the recent years. Describing lines in order to match them across frames has only been addressed more recently, due to the multiple challenges accompanying lines. For a full overview of previous line detectors and descriptors, we refer the reader to Section 1.2.2.

Joint detection and description of learned features. Jointly learned point detectors and descriptors [Ono et al., 2018, DeTone et al., 2018, Revaud et al., 2019, Luo et al.,

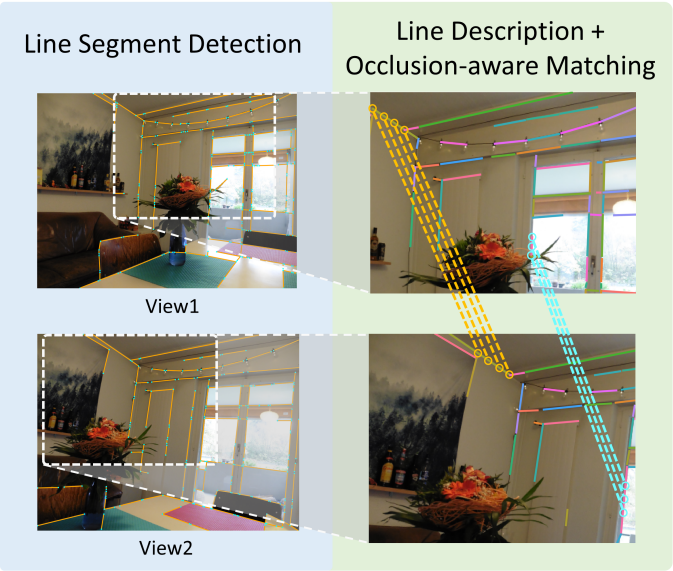


Figure 3.1: Line segment detection and matching. Our approach detects repeatable lines and is able to match sub-segments to handle partial occlusions. On the right, lines of the same color are matched together.

2020] propose to share computation between the keypoint detection and description to get fast inference and better feature representations from multi-task learning. The describe-then-detect trend first computes a dense descriptor map and then extracts the keypoints location from it [Dusmanu et al., 2019, Luo et al., 2020, Yang et al., 2020, Tian et al., 2020]. Supervision is provided by either pixel-wise correspondences from SfM [Dusmanu et al., 2019, Luo et al., 2020], or from image level correspondences only [Yang et al., 2020]. HF-Net [Sarlin et al., 2019] unifies keypoint detection, local and global description through a multi-task distillation with multiple teacher networks. Towards the fully unsupervised spectrum, recent methods tightly couple the detector and descriptor learning to output repeatable and reliable points [Christiansen et al., 2019, Revaud et al., 2019]. On the other hand, Superpoint [DeTone et al., 2018] first learned the concept of interest points by pretraining a corner detector on a synthetic dataset and later transferring it to real world images. We adopt here a similar approach extended to line segments.

Line matching. Beyond simply comparing descriptor similarities, several works tried to leverage higher-level structural cues to guide line matching [Li et al., 2016b]. One approach considers the neighboring lines/points and finds similar patterns across images, for instance through local clusters of lines [Wang et al., 2009a], intersections between lines [Kim and Lee, 2010] or line-junction-line structures [Li et al., 2014, Li et al., 2016d]. However, these methods cannot match isolated lines. Another direction is to find co-planar sets of lines and points and to leverage line-point invariants as well as simple point matching to achieve line matching [Lourakis et al., 1998, Fan et al., 2010, Fan et al., 2012b, Ramalingam et al., 2015b]. Finally, a last approach consists in matching points sampled along a line, using for example intensity information and epipolar geometry [Schmid and Zisserman, 1997b] or simply point descriptors [Vakhitov and Lempitsky, 2019]. Our work follows this direction but offers a flexible matching of the points along the line, which handles occlusions.

3.4 Method

We propose a unified network to perform line segment detection and description, allowing to match lines across different images. We achieve self-supervision in two steps. Our detector is first pretrained on a synthetic dataset with known ground truth. The full detector and descriptor can then be trained by generating pseudo ground truth line segments on real images using the pretrained model. We provide an overview of our training pipeline in Figure 3.2 and detail its parts in the following sections.

3.4.1 Problem Formulation

Line segments can be parameterized in many ways: with two endpoints; with a middle point, a direction and a length; with a middle point and offsets for the endpoints; with an attraction field; etc. In this work, we chose the line representation with two endpoints for its simplicity and compatibility with our self-supervision process discussed in Section 3.4.4. For an image I with spatial resolution $h \times w$, we denote $P = \{p_n\}_{n=1}^{N}$ the set of all junctions of I and $L = \{l_m\}_{m=1}^{M}$ the set of line segments. A line segment l_m is defined by a pair of endpoints $(e_m^1, e_m^2) \in P^2$.

3.4.2 Junction and Line Heatmap Inference

Our network takes grayscale images as input, processes them through a shared backbone encoder that is later divided into three different outputs. A junction map $\mathbf{J}$ predicts the probability of each pixel to be a line endpoint, a line heatmap $\mathbf{H}$ provides the probability of a pixel to be on a line, and a descriptor map $\mathbf{D}$ yields a pixel-wise local descriptor. We focus here on the optimization of the first two branches, while the following sections describe their combination to retrieve and match the line segments of an image.

We adopt a similar approach to SuperPoint's keypoint decoder [DeTone et al., 2018] for the junction branch, where the output is a coarse $\frac{h}{8} \times \frac{w}{8} \times 65$ feature map $\mathbf{J^c}$. Each 65-dimensional vector corresponds to an 8×8 patch plus an extra "no junction" dustbin. We define the ground truth junctions $\mathbf{y} \in \{1, ..., 65\}^{\frac{h}{8} \times \frac{w}{8}}$ indicating the index of the true junction position in each patch. A junction is randomly selected when several ground truth junctions land in the same patch and a value of 65 means that there is no junction. The junction loss is then a cross-entropy loss between $\mathbf{J^c}$ and $\mathbf{y}$:

$$\mathcal{L}_{junc} = \frac{64}{h \times w} \sum_{i,j=1}^{\frac{h}{8}, \frac{w}{8}} -\log \left(\frac{\exp(J^c_{ijy_{ij}})}{\sum_{k=1}^{65} \exp(J^c_{ijk})} \right) . \tag{3.1}$$

At inference time, we perform a softmax on the channel dimension and discard the 65th dimension, before resizing the junction map to get the final $h \times w$ grid.

The second branch outputs a line heatmap $\mathbf{H}$ at the image resolution $h \times w$. Given a binary ground truth $\mathbf{H^{GT}}$ with a value of 1 for pixels on lines and 0 otherwise, the line

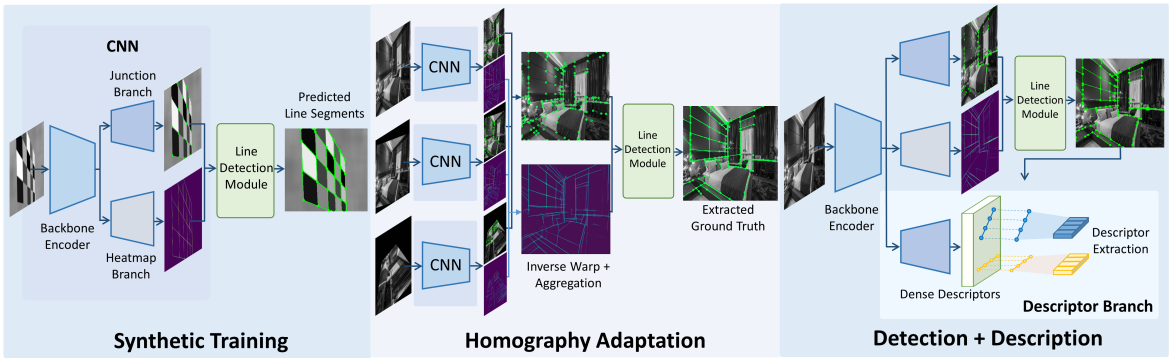


Figure 3.2: Training pipeline overview. Left: Our detector network is first trained on a synthetic dataset with known ground truth. **Middle:** A pseudo ground truth of line segments is then generated on real images through homography adaptation. **Right:** Finally, the full model with descriptors is trained on real images using the pseudo ground truth.

heatmap is optimized via a binary cross-entropy loss:

$$\mathcal{L}_{line} = \frac{1}{h \times w} \sum_{i,j=1}^{h,w} -H_{ij}^{GT} \log(H_{ij}) \ . \tag{3.2}$$

3.4.3 Line Detection Module

After inferring the junction map $\mathbf{J}$ and line heatmap $\mathbf{H}$, we threshold $\mathbf{J}$ to keep the maximal detections and apply a Non-Maximum Suppression (NMS) to extract the segment junctions $\hat{P}$. The line segment candidates set $\hat{L}_{cand}$ is composed of every pair of junctions in $\hat{P}$. Extracting the final line segment predictions $\hat{L}$ based on $\mathbf{H}$ and $\hat{L}_{cand}$ is non-trivial as the activations along a segment defined by two endpoints may vary a lot across different candidates. Our approach can be broken down into five parts: (1) line NMS, (2) regular sampling between endpoints, (3) adaptive local-maximum search, (4) average score, and (5) inlier ratio.

Line NMS: In some application requiring line matching, having multiple overlapping segments may hinder the matching as the descriptor will have a harder job at discriminating close lines. Therefore, an NMS mechanism is necessary for lines. Unlike point or bounding box NMS, there is no well-established procedure for line NMS. Contrary to usual NMS methods which are used as postprocessing steps, we implement our line NMS as a preprocessing step, which actually speeds up the overall line segment detection as it removes some line candidates. Starting from the initial line candidates set $\hat{L}_{cand}$ consisting of all pairs of junctions, we remove the line segments containing other junctions between their two endpoints. To identify whether a junction lies on a line segment, we first project the junction on the line and check if it falls within the segment boundaries. When it does, the junction is considered to be on the line segment if it is at a distance of less than ξ_{cs} pixels from the line.

Regular sampling between endpoints: Instead of fetching all the rasterized pixels between the two endpoints [Zhou et al., 2019b], we sample N_s uniformly spaced points (including the two endpoints) along the line segment.

Adaptive local-maximum search: Using bilinear interpolation to fetch the heatmap values at

the extracted points q_k may discard some candidates due to the misalignment between the endpoints and the heatmap, especially for long lines. To alleviate that, we search for the local maximal heatmap activation h_k around each sampled location q_k within a radius r proportional to the length of the line. Given a line segment $\hat{l} = (\hat{e}^1, \hat{e}^2)$ from the candidate set $\hat{L}$ in an image of size $h \times w$, the search radius r is defined as:

$$r = r_{min} + \lambda \frac{\left\| \hat{e}^1 - \hat{e}^2 \right\|}{\sqrt{h^2 + w^2}} , \tag{3.3}$$

where $r_{min} = \frac{\sqrt{2}}{2}$ is the minimum search radius and λ is a hyper-parameter to adjust the linear dependency on the segment lengths. The r_{min} parameter can be kept constant across different image resolutions, without performance degradation.
Average score: The average score is defined as the mean of all the sampled heatmap values: $y_{avg} = \frac{1}{N_s} \sum_{k=1}^{N_s} h_k$. Given a threshold ξ_{avg}, valid line segment candidates should satisfy $y_{avg} \geq \xi_{avg}$.
Inlier ratio: To remove spurious detections with only a few high activations and holes along the line, we also consider an inlier ratio $y_{inlier} = \frac{1}{N_s} |\{h_k | h_k \geq \xi_{avg}, h_k \in H\}|$. Given an inlier ratio threshold ξ_{inlier}, we only keep candidates satisfying $y_{inlier} \geq \xi_{inlier}$.

3.4.4 Self-Supervised Learning Pipeline

Inspired by the success of SuperPoint [DeTone et al., 2018], we extend their homography adaptation to the case of line segments. Let f_{junc} and f_{heat} represent the forward pass of our network to compute the junction map and the line heatmap. We start by aggregating the junction and heatmap predictions using a set of N_h homographies $(\mathcal{H}_i)_{i=1}^{N_h}$:

$$\hat{\mathbf{J}}(I; f_{junc}) = \frac{1}{N_h} \sum_{i=1}^{N_h} \mathcal{H}_i^{-1} \Big(f_{junc}(\mathcal{H}_i(I)) \Big) , \tag{3.4}$$

$$\hat{\mathbf{H}}(I; f_{heat}) = \frac{1}{N_h} \sum_{i=1}^{N_h} \mathcal{H}_i^{-1} \Big(f_{heat}(\mathcal{H}_i(I)) \Big) . \tag{3.5}$$

We then apply the line detection module to the aggregated maps $\hat{\mathbf{J}}$ and $\hat{\mathbf{H}}$ to obtain the predicted line segments $\hat{L}$, which are then used as ground truth for the next training round. Figure 3.2 provides an overview of the pipeline. Similar to Superpoint, this process can be iteratively applied to improve the label quality. However, we found that a single round of adaptation already provides sufficiently good labels.

3.4.5 Line Description

Describing lines in images is a problem inherently more difficult than describing feature points. A line can be partially occluded, its endpoints are not always repeatable across views, and the appearance of a line can significantly differ under viewpoint changes. To tackle these challenges, we depart from the classical description of a patch centered on the line [Lange et al., 2019b, Lange et al., 2020], that is not robust to occlusions and endpoints shortening.

Motivated by the success of learned point descriptors, we formulate our line descriptor as a sequence of point descriptors sampled along the line. Given a good coverage of the points along the line, even if part of the line is occluded, the points on the non-occluded part will store enough line details and can still be matched.

The descriptor head of our network outputs a descriptor map $\mathbf{D} \in \mathbb{R}^{\frac{h}{4} \times \frac{w}{4} \times 128}$ and is optimized through the classical point-based triplet loss [Balntas et al., 2016b, Mishchuk et al., 2017] used in other dense descriptors [Dusmanu et al., 2019]. Given a pair of images I_1 and I_2 and matching lines in both images, we regularly sample points along each line and extract the corresponding descriptors $(\mathbf{D_1^i})_{i=1}^n$ and $(\mathbf{D_2^i})_{i=1}^n$ from the descriptor maps, where n is total number of points in an image. The triplet loss minimizes the descriptor distance of matching points and maximizes the one of non-matching points. The positive distance is defined as

$$p_i = ||\mathbf{D_1^i} - \mathbf{D_2^i}||_2 \ . \tag{3.6}$$

The negative distance is computed between a point and its hardest negative example in batch:

$$n_i = \min \left(||\mathbf{D_1^i} - \mathbf{D_2^{h_2(i)}}||_2, \ ||\mathbf{D_1^{h_1(i)}} - \mathbf{D_2^i}||_2 \right) , \tag{3.7}$$

where $h_1(i) = \arg\min_{k \in [1,n]} ||\mathbf{D_1^k} - \mathbf{D_2^i}||$ such that the points i and k are at a distance of at least T pixels and are not part of the same line, and similarly for $h_2(i)$. The triplet loss with margin M is then defined as

$$\mathcal{L}_{desc} = \frac{1}{n} \sum_{i=1}^{n} \max(0, M + p_i - n_i) \ . \tag{3.8}$$

3.4.6 Multi-Task Learning

Detecting and describing lines are independent tasks with different homoscedastic aleatoric uncertainties and their respective losses can have different orders of magnitude. Thus, we adopt the multi-task learning proposed by Kendall *et al.* [Kendall et al., 2018] with a dynamic weighting of the losses, where the weights w_{junc}, w_{line} and w_{desc} are optimized during training [Kendall and Cipolla, 2017, Sarlin et al., 2019]. The total loss becomes:

$$\begin{aligned} \mathcal{L}_{total} = \ & e^{-w_{junc}} \mathcal{L}_{junc} + e^{-w_{line}} \mathcal{L}_{line} \\ & + e^{-w_{desc}} \mathcal{L}_{desc} + w_{junc} + w_{line} + w_{desc} \ . \end{aligned} \tag{3.9}$$

3.4.7 Line Matching

At inference time, two line segments are compared based on their respective collection of point descriptors sampled along each line. However, some of the points might be occluded or, due to perspective changes, the length of a line can vary and the sampled points may be misaligned. The ordering of the points matched along the line should nevertheless be constant, i.e. the line descriptor is an ordered sequence of descriptors, not just a set. To solve this sequence assignment problem, we take inspiration from nucleotide alignment in

bioinformatics [Needleman and Wunsch, 1970] and pixel alignment along scanlines in stereo vision [Dieny et al., 2011]. We thus propose to find the optimal point assignment through the dynamic programming algorithm originally introduced by Needleman and Wunsch [Needleman and Wunsch, 1970].

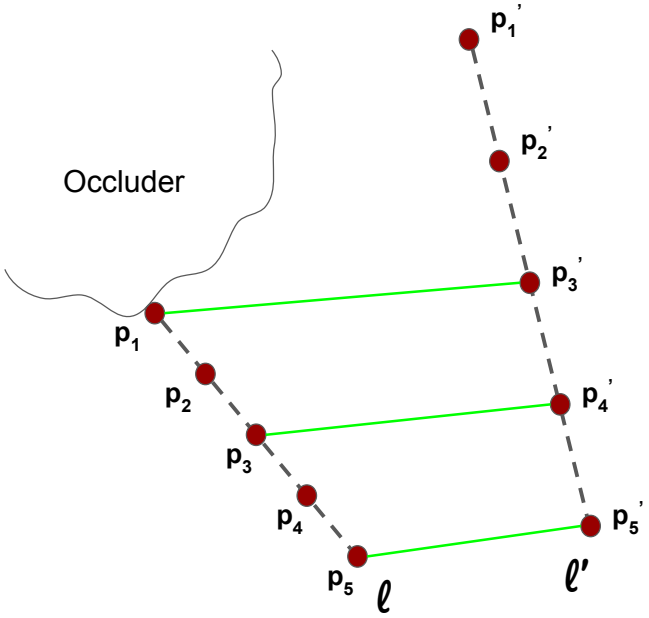
When matching two sequences of points, each point can be either matched to another one or skipped. The score attributed to a match of two points depends on the similarity of their descriptors (i.e. their dot product), so that a higher similarity gives a higher score. Skipping a point is penalized by a *gap* score, which has to be adjusted so that it is preferable to match points with high similarity but to skip the ones with low similarity. The total score of a line match is then the sum of all skip and match operations of the line points. The Needleman-Wunsch (NW) algorithm returns the optimal matching sequence maximizing this total score. This is achieved with dynamic programming by filling a matrix of scores row by row, as depicted in Figure 3.3. Given a sequence of m points along a line l, m' points along l', and the associated descriptors $\mathbf{D}$ and $\mathbf{D}'$, this score matrix $\mathbf{S}$ is an $(m+1) \times (m'+1)$ grid where $\mathbf{S}(i,j)$ contains the optimal score for matching the first i points of l with the first j points of l'. The grid is initialized by the *gap* score in the first row and column, and is sequentially filled row by row, using the scores stored in the left, top and top-left cells:

$$\begin{aligned}\mathbf{S}(i,j) = \max\Big(&\mathbf{S}(i-1,j) + gap, \mathbf{S}(i,j-1) + gap, \\ &\mathbf{S}(i-1,j-1) + \mathbf{D}^{\mathbf{i}^T}\mathbf{D}'^{\mathbf{j}}\Big) \ .\end{aligned} \tag{3.10}$$

Once the matrix is filled, we select the highest score in the grid and use it as a match score for the candidate pair of lines. Each line of the first image is then matched to the line in the second image with the maximum match score. Note that starting from the highest scoring cell of the grid, one can backtrack in the grid to determine which points were actually matched. The NW algorithm is thus actually able to match sub-segments of the full line, which is essential when a line is partially occluded.

3.4.8 Implementation Details

Network implementation. To have a fair comparison with most wireframe parsing methods [Zhou et al., 2019a, Xue et al., 2020, Lin et al., 2020], we use the same stacked hourglass network [Newell et al., 2016] for our backbone. Given an image with resolution $h \times w$, the output of the backbone encoder is a $\frac{h}{4} \times \frac{w}{4} \times 256$ feature map. The three heads of the network are implemented as follows:
Junction branch: It is composed of a 3×3 convolution with stride 2 and 256 channels, followed by a 1×1 convolution with stride 1 and 65 channels, leading to the $\frac{h}{8} \times \frac{w}{8} \times 65$ junction map.
Heatmap branch: To keep a light network and avoid artifacts from transposed convolutions, we perform two consecutive subpixel shuffles [Shi et al., 2016] blocks to perform a $\times 4$ upsampling. More precisely, we use two 3×3 conv layers of output channel sizes 256 and 64, each of them followed by batch normalization, ReLU activation and a $\times 2$ subpixel shuffle for upsampling. A final 1×1 convolution with output channel 1 and sigmoid activation is then used to get the final line heatmap of resolution $h \times w$.



		p_1'	p_2'	p_3'	p_4'	p_5'
	0	**0**	**0**	0	0	0
p_1	0	0	0.1	**0.8**	0.4	0.2
p_2	0	0	0.3	**0.8**	1.3	0.6
p_3	0	0	0.2	0.9	**1.4**	1.4
p_4	0	0	0.1	0.9	**1.4**	1.9
p_5	0	0	0	0.9	1.5	**2.1**

Figure 3.3: Computation of a line match score. The optimal path selected by the Needleman-Wunsch algorithm is shown in green for matches and blue for skipping a point, using here a *gap* score of zero.

Descriptor branch: The backbone encoding is processed by two consecutive convolutions of kernels 3×3 and 1×1, and output channels 256 and 128, to produce a $\frac{h}{4} \times \frac{w}{4} \times 128$ feature descriptor map. This semi-dense map can be later bilinearly interpolated at any point location. The triplet loss is optimized with a margin $M = 1$ and a minimal distance to the hardest negative of $T = 8$ pixels.

We use ReLU activations after each convolution and optimize the network with the Adam solver [Kingma and Ba, 2014] with a learning rate of 0.0005. Images are resized to a 512×512 resolution and converted to grayscale during training.

Line parameters. We use a junction threshold of $\frac{1}{65}$, a heatmap threshold $\xi_{avg} = 0.25$, an inlier threshold $\xi_{inlier} = 0.75$, an NMS threshold $\xi_{cs} = 3$ pixels, $\lambda = 3$ pixels in the adaptive local-maximum search, extract $N_s = 64$ samples along each line to compute the heatmap and inlier scores, and we use $N_h = 100$ homographies for the homography adaptation. The optimal line parameters were selected by a grid search on the validation set.

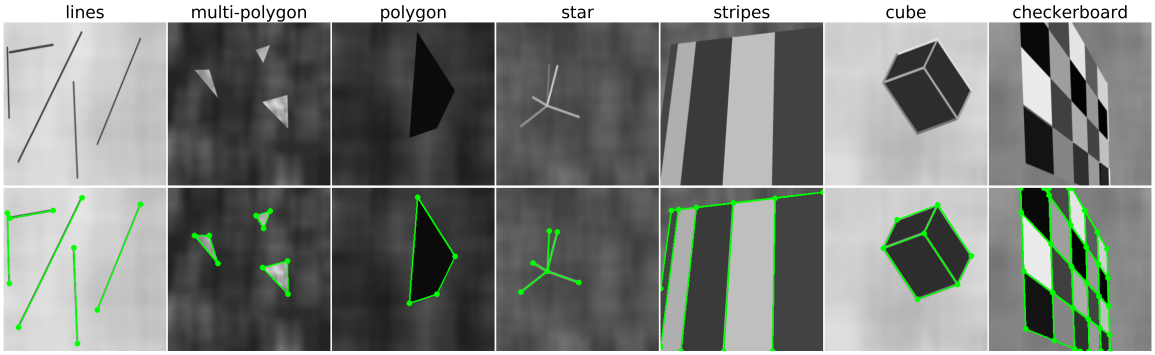


Figure 3.4: Image examples from the synthetic dataset. First row: rendered images. Second row: images with estimated junctions and line segment labels.

Matching details. The line descriptor is computed by regularly sampling up to 5 points along each line segment, but keeping a minimum distance of 8 pixels between each point. Since the ordering of the points might be reversed from one image to the other, we run the matching twice with one point-set flipped. A *gap* score of 0.1 empirically yields the best results during the NW matching. To speed up the line matching, we pre-filter the set of line candidates with a simple heuristic. Given the descriptor of the 5 points sampled on a line of I_1 to be matched, we compute the similarity with their nearest neighbor in each line of I_2, and average these scores for each line. This yields a rough estimate of the line match score, and we keep the top 10 best lines as candidates for the NW matching. Finally, we retain at matching time only the pairs that are mutually matched.

Training dataset. Our synthetic dataset consists of rendered 2D shapes including triangles, lines, rectangles, and stripes, similarly as in Superpoint [DeTone et al., 2018] (see Figure 3.4). We label the corners of these shapes as junctions and edges as line segments. Gaussian noise, salt and pepper noise, and random shades are added to the rendered images as data augmentation. We follow the same process as in SuperPoint [DeTone et al., 2018] to generate the random homographies. They are generated as a composition of simple transformations with pre-defined ranges: scaling (normal distribution $\mathcal{N}(1., 0.1)$), translation (uniform distribution within the image boundaries), rotation (uniformly in $[-90°, +90°]$), and perspective transform. We generate $20,000$ training samples and 400 testing samples.

For the training with real images, we use the Wireframe dataset [Huang et al., 2018], for a fair comparison with the current state of the art also trained on these images. We follow the split policy in LCNN [Zhou et al., 2019a]: $5,000$ images for training and 462 images for testing, but only use the images and ignore the ground truth lines provided by the dataset.

3.5 Experiments

3.5.1 Line Segment Detection Evaluation

To evaluate our line segment detection, we use the test split of the Wireframe dataset [Huang et al., 2018] and the YorkUrban dataset [Denis et al., 2008], which contains 102 outdoor images. For both datasets, we generate a fixed set of random homographies and warp each

image to get a pair of matching images.

Line segment distance metrics. A line distance metric needs to be defined to evaluate the accuracy of a line detection. We use the two following metrics:

Structural distance ($\mathbf{d_s}$): The structural distance of two line segments l_1 and l_2 is defined as:

$$\begin{aligned} \mathrm{d_s}(l_1, l_2) = \min(&\left\|e_1^1 - e_2^1\right\|_2 + \left\|e_1^2 - e_2^2\right\|_2 , \\ &\left\|e_1^1 - e_2^2\right\|_2 + \left\|e_1^2 - e_2^1\right\|_2) , \end{aligned} \tag{3.11}$$

where (e_1^1, e_1^2) and (e_2^1, e_2^2) are the endpoints of l_1 and l_2 respectively. Contrary to the formulation of recent wireframe parsing works [Zhou et al., 2019a, Xue et al., 2020], we do not use square norms to make it directly interpretable in terms of endpoints distance.

Orthogonal distance ($\mathbf{d_{orth}}$): The orthogonal distance of two line segments l_1 and l_2 is defined as the average of two asymmetrical distances d_a:

$$\mathrm{d_a}(l_i, l_j) = \left\|e_j^1 - p_{l_i}(e_j^1)\right\|_2 + \left\|e_j^2 - p_{l_i}(e_j^2)\right\|_2 , \tag{3.12}$$

$$\mathrm{d_{orth}}(l_1, l_2) = \frac{\mathrm{d_a}(l_1, l_2) + \mathrm{d_a}(l_2, l_1)}{2} , \tag{3.13}$$

where $p_{l_j}(.)$ denotes the orthogonal projection on line l_j. When searching the nearest line segment with this distance, we ignore the line segments with an overlap below 0.5. This definition allows line segments corresponding to the same 3D line but with different line lengths to be considered as close, which can be useful in localization tasks [Micusik and Wildenauer, 2014].

Line segment detection metrics. Since the main objective of our line segment detection method is to extract repeatable and reliable line segments from images, evaluating it on the manually labeled lines of the Wireframe dataset [Huang et al., 2018] is not suitable. We thus instead adapt the detector metrics proposed for SuperPoint [DeTone et al., 2018] to line segments using pairs of images: the repeatability and localization error. Both of these metrics are computed using pairs of images I_1 and I_2, where I_2 is a warped version of I_1 under a homography $\mathcal{H}$. Each image is associated with a set of line segments $L_1 = \{l_m^1\}_{m=1}^{M_1}$ and $L_2 = \{l_m^2\}_{m=1}^{M_2}$, and d refers to one of the two line distances $\mathrm{d_s}$ or $\mathrm{d_{orth}}$.

Repeatability: The repeatability measures how often a line can be re-detected in different views. It is the average percentage of lines in the first image that have a matching line when reprojected in the second image. Two lines are considered to be matched when their distance is lower than a threshold ϵ. This metric is computed symmetrically across the two images and averaged. We define it as:

$$\forall l \in L_1, \mathrm{C}_{L_2}(l) = \begin{cases} 1 & \text{if}(\min_{l_j^2 \in L_2} \mathrm{d}(l, l_j^2)) \leq \epsilon, \\ 0 & \text{otherwise} , \end{cases} \tag{3.14}$$

$$\text{Rep-}\epsilon = \frac{\sum_{i=1}^{M_1} \mathrm{C}_{L_2}(l_i^1) + \sum_{j=1}^{M_2} \mathrm{C}_{L_1}(l_j^2)}{M_1 + M_2} . \tag{3.15}$$

Localization error: The localization error with tolerance ϵ is the average line distance between a line and its re-detection in the second image, only considering the matched lines.:

$$\text{LE-}\epsilon = \frac{\sum_{j \in Corr} \min_{l_i^1 \in L1} \mathrm{d}(l_i^1, l_j^2)}{|Corr|} \ , \tag{3.16}$$

$$Corr = \{j \mid C_{L_1}(l_j^2) = 1,\ l_j^2 \in L_2\} \ , \tag{3.17}$$

where $|\cdot|$ measures the cardinality of a set.

Evaluation on the Wireframe and YorkUrban datasets. We compare in Tables 3.1 and 3.2 our line segment detection method with 5 baselines including the handcrafted Line Segment Detector (LSD) [Von Gioi et al., 2008], wireframe parsing methods such as LCNN [Zhou et al., 2019a], the Holistically Attracted Wireframe Parsing (HAWP) [Xue et al., 2020], the Tri-Point Line Segment Detector (TP-LSD) [Huang et al., 2020], and Deep Hough-transform Line Priors (DeepHough) [Lin et al., 2020]. LSD is used with a minimum segment length of 15 pixels. For LCNN, HAWP, and DeepHough, we chose thresholds (0.98, 0.97, and 0.9 respectively) on the line scores to maximize their performances. We show two TP-LSD variants: HG using the same HourGlass backbone [Newell et al., 2016] as the other wireframe parsing baselines and our method, and TP512 that uses a ResNet34 [He et al., 2016] backbone.

Overall, our method achieves the best performance in terms of repeatability and localization error on both datasets. We also include our method with Non-Maximum Suppression (NMS), which removes the segments having other junctions between the two endpoints to avoid overlapping segments in the predictions $\hat{L}$. Without overlapping segments, the performance slightly decreases but we get fewer segments and faster inference speed.

	$\mathrm{d_s}$		$\mathrm{d_{orth}}$		Time ↓	# lines / image
	Rep-5 ↑	LE-5 ↓	Rep-5 ↑	LE-5 ↓		
LSD	0.358	2.079	0.707	0.825	**0.026**	228
LCNN @0.98	0.434	2.589	0.570	1.725	0.120	76
HAWP @0.97	0.451	2.625	0.537	1.738	0.035	47
DeepHough @0.9	0.419	2.576	0.618	1.720	0.289	135
TP-LSD HG	0.358	3.220	0.647	2.212	0.038	72
TP-LSD TP512	0.563	2.467	0.746	1.450	0.097	81
Ours w/ NMS	0.557	**1.995**	0.801	1.119	0.042	116
Ours	**0.616**	2.019	**0.914**	**0.816**	0.074	447

Table 3.1: Line detection evaluation on the Wireframe dataset [Huang et al., 2018]. We compare repeatability and localization error for an error threshold of 5 pixels in structural and orthogonal distances. Our approach provides the most repeatable and accurate line detections compared to the other baselines.

	d_s		d_{orth}		Time ↓	# lines / image
	Rep-5 ↑	LE-5 ↓	Rep-5 ↑	LE-5 ↓		
LSD	0.357	2.116	0.704	0.876	**0.031**	359
LCNN @0.98	0.318	2.662	0.449	1.784	0.206	103
HAWP @0.97	0.295	2.566	0.368	1.757	0.045	59
DeepHough @0.9	0.315	2.695	0.535	1.751	0.519	206
TP-LSD HG	0.233	3.357	0.524	2.395	0.038	113
TP-LSD TP512	0.433	2.612	0.633	1.555	0.099	125
Ours w/ NMS	0.528	**1.902**	0.787	1.107	0.064	222
Ours	**0.582**	1.932	**0.913**	**0.713**	0.093	1085

Table 3.2: Line detection evaluation on the YorkUrban dataset [Denis et al., 2008]. We compare repeatability and localization error for an error threshold of 5 pixels in structural and orthogonal distances. Our approach provides the most repeatable and accurate line detections compared to the other baselines.

3.5.2 Line Segment Description Evaluation

Line descriptor metrics. Our line descriptor is evaluated on several matching metrics, both on hand-labeled line segments and on detected line segments (LSD or our predicted lines). When using ground truth lines, there is an exact one-to-one line correspondence. For predicted lines, ground truth matches are computed with a threshold ϵ similarly as for the detector metrics. When depth is available, the lines are projected to 3D and directly compared in 3D space. Only lines with a valid reprojection in the other image are considered.
Accuracy: Percentage of correctly matched lines given a set of ground truth line matches.
Receiver Operating Characteristic (ROC) curve: Given a set of matching lines, we compute the SIFT [Lowe, 2004] descriptor of each endpoint, average the SIFT distances between each pair of lines, and use the second nearest neighboring line as negative match. The ROC curve is then the True Positive Rate (TPR) plotted against the False Positive Rate (FPR). The curve is obtained by varying the descriptor similarity threshold defining a positive match.
Precision: Ratio of true positive matches over the total number of predicted matches.
Recall: Ratio of true positive matches over the total number of ground truth matches.

Descriptor evaluation on ground truth lines. Our first experiment aims at evaluating our approach on a perfect set of lines with a one-to-one matching. We thus use the Wireframe test set with its ground truth lines. We compare our line matcher against 3 competing baselines: the handcrafted LBD [Zhang and Koch, 2013], the LLD [Vakhitov and Lempitsky, 2019] and the WLD [Lange et al., 2020], an improved version of the DLD [Lange et al., 2019b]. The results are shown in Figure 3.5.

Since LLD was trained on consecutive video frames with nearly no rotation between the images, it is not rotation invariant, hence its poor performance on the rotated images of our dataset. WLD showed that they were able to surpass the handcrafted LBD, and our descriptor gets a slight improvement over WLD by 5%.

Robustness to occlusion experiment. In real-world applications, the detected lines across multiple views are rarely exactly the same, and some may be partially occluded or with different endpoints. To evaluate the robustness of our descriptor to these challenges, we modify the Wireframe test set to include artificial occluders. We overlay ellipses with random parameters and synthetic textures on the warped image of each pair, until at most s% of

Method	Accuracy ↑
LBD	0.610
LLD	0.265
WLD	0.933
SOLD2 (Ours)	**0.978**

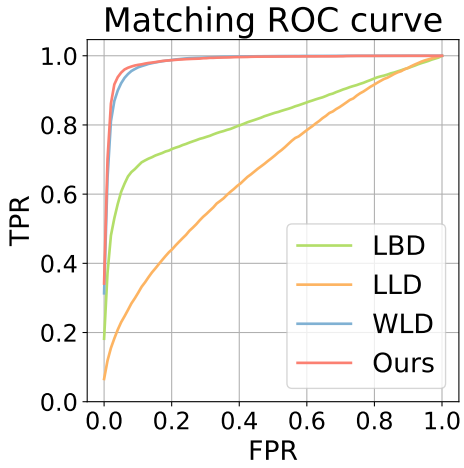


Figure 3.5: Descriptor evaluation on the Wireframe dataset [Huang et al., 2018] with ground truth lines. Matching the exact same lines yields a very high score for WLD and our method.

the lines are covered. We also shorten the line segments accordingly, so that each line stops at the occluders boundary. We compare line matches for various values of s and show the results in Figure 3.6.

While all methods show a decrease in performance with a larger occlusion, SOLD2 outperforms the other baselines by a large margin for all degrees of occlusion. Note the significant drop for the learned baseline WLD, which operates on line patches and is thus severely affected by occlusions. This experiment thus validates the robustness of our method to occlusion and unstable line endpoints.

Descriptor evaluation on predicted lines. To assess the performance of our proposed line description and matching, we also compute the matching metrics on predicted line segments instead of using hand-labeled lines. We perform two sets of experiments, on the Wireframe test set and on the ETH3D [Schöps et al., 2017] images which offer real world camera motions and can contain more challenging viewpoint changes than homographic warps.

The ETH3D dataset [Schöps et al., 2017] is composed of 13 scenes taken in indoor as well as outdoor environments. Each image comes with the corresponding camera intrinsics and depth map, and a 3D model of each scene built with Colmap [Schönberger and Frahm, 2016] is provided as well. We use the undistorted image downsampled by a factor of 8 to run the line detection and description and we select all pairs of images that share at least 500 covisible 3D points in the provided 3D models. We then use the depth maps and camera intrinsics to reproject the lines in 3D and compute the descriptor metrics in 3D space. While the depth maps have been obtained from a high-precision laser scanner, they contain some holes, in particular close to depth discontinuities. Since these discontinuities are actually where lines are often located, we inpaint the depth in all of the invalid areas at up to 10 pixels from a valid depth pixel. We used the Non-Local Spatial Propagation Network (NLSPN) [Park et al., 2020], the current state of the art in deep depth inpainting guided with RGB images.

In both experiments, we run the LSD detector and compute all the line descriptor methods

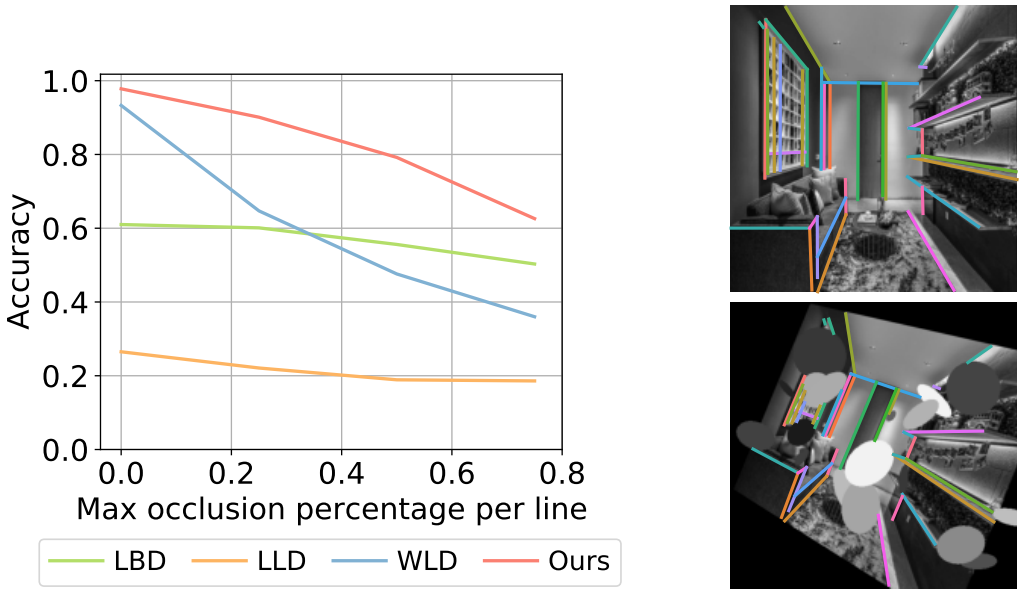


Figure 3.6: Robustness to occlusion. Left: When evaluated on the Wireframe dataset with ground truth lines and random occluders, our method shows a higher robustness to occlusion compared to other methods. **Right:** Example of matches in the presence of occluders.

		Wireframe		ETH3D	
Lines	Desc	Precision ↑	Recall ↑	Precision ↑	Recall ↑
LSD	LBD	0.496	0.597	0.132	0.376
	LLD	0.123	0.116	0.085	0.230
	WLD	0.528	0.804	0.127	0.398
	SOLD2 (Ours)	**0.591**	**0.889**	**0.159**	**0.525**
Ours	SOLD2 (Ours)	**0.882**	0.688	**0.196**	0.538
Ours w/ NMS	SOLD2 (Ours)	0.777	**0.949**	0.190	**0.688**

Table 3.3: Matching precision and recall using LSD [Von Gioi et al., 2008] and our lines. We use a threshold of 5 pixels in structural distance for the Wireframe [Huang et al., 2018] images and of 5cm for the ETH3D [Schöps et al., 2017] images to define the ground truth matches.

on them and also compare it with our full line prediction and description. Table 3.3 and Figure 3.7 evaluate the precision and recall of all methods.

Whether it is on synthetically warped images, or with real camera changes, SOLD2 outperforms all the descriptor baselines both in terms of matching precision and recall when compared on LSD lines. Using our own lines also improves the metrics, but the best performance is achieved when we apply a line NMS to remove overlapping segments. Having no overlap makes it indeed easier for the descriptor to discriminate the closest matching line.

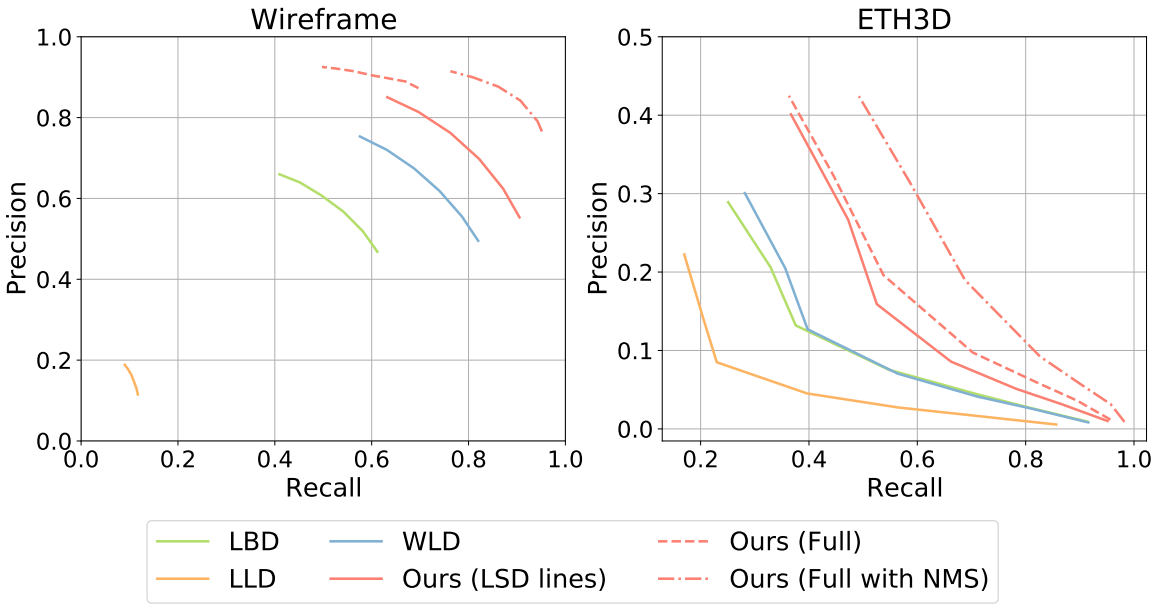


Figure 3.7: Precision-Recall curves on predicted lines. Our descriptor outperforms the other baselines when compared on LSD lines and the best performance is achieved for our full approach with our lines and descriptors.

3.5.3 Application: Homography Estimation

To validate the real-world applications of our method, we used the line segment detections and descriptors to match segments across pairs of images of the Wireframe dataset [Huang et al., 2018] related by a homography and estimate the homography with RANSAC [Fischler and Bolles, 1981b]. We sample minimal sets of 4 lines to fit a homography and run up to $1,000,000$ iterations with the Locally Optimized RANSAC (LO-RANSAC) [Lebeda et al., 2012] implementation of Sattler *et al.*[1]. The reprojection error is computed with the orthogonal line distance. We compute the accuracy of the homography estimation similarly as in SuperPoint [DeTone et al., 2018] by warping the four corners of the image with the estimated homography, warping them back to the initial image with the ground truth homography and computing the reprojection error of the corners. We consider the estimated homography to be correct if the average reprojection error is less than 3 pixels. The results are listed in Table 3.4.

		Homography estimation		
Lines	Desc	Accuracy ↑	# inliers	Reproj. error ↓
LSD	LBD	0.781	80	0.791
	LLD	0.201	21	0.927
	WLD	0.920	116	0.868
	Ours	**0.948**	116	0.863
Ours	Ours	0.935	200	**0.780**
SuperPoint		0.582	173	0.996

Table 3.4: Evaluation results of homography estimation. The homography between images of the Wireframe dataset [Huang et al., 2018] is estimated from line matches using RANSAC. We use a threshold of 5 pixels in orthogonal line distance to consider a match to be an inlier.

[1] https://github.com/tsattler/RansacLib

When compared on LSD line [Von Gioi et al., 2008], our descriptors provide the highest accuracy among all baselines, and our full pipeline achieves a similar performance. When using our lines, we use a similar refinement of the junctions as in LSD [Von Gioi et al., 2008]: we sample small perturbations of the endpoints by a quarter of a pixel and keep the perturbed endpoints maximizing the line average score. Similarly to feature point methods [DeTone et al., 2018, Dusmanu et al., 2019], this experiment shows that learned features are still on par or slightly worse than handcrafted detections in terms of localization error.

We also add to the comparison the results of homography estimation for a learned feature point detector and descriptor, SuperPoint [DeTone et al., 2018]. The point-based approach performs significantly worse than our method, due to the numerous textureless scenes and repeated structures present in the Wireframe dataset. We also found that SuperPoint is not robust to rotations above 45 degrees, while our line descriptor can leverage its ordered sequence of descriptors to achieve a better invariance to rotation.

3.5.4 Ablation Study

To validate the design choices of our approach, we perform an ablation study on the descriptor. *SIFT endpoints* computes a SIFT descriptor [Lowe, 2004] for both endpoints using the line direction as keypoint orientation, and averages the endpoints descriptor distance of each line candidate pair to get the line match scores. *Average descriptor* computes a line descriptor by averaging the descriptors of all the points sampled along each line. *NN average* computes the descriptor similarity of each line point with its nearest neighbor in the other line and averages all the similarities to get a line match score. *D2-Net sampling* and *ASLFeat sampling* refer to our proposed matching method where the points are sampled along the lines according to the saliency score introduced in D2-Net [Dusmanu et al., 2019] and ASLFeat [Luo et al., 2020], respectively. Finally, we test our method with various numbers of points sampled along each line. Table 3.5 compares the accuracy of all these methods on the Wireframe dataset with ground truth lines both with and without occluders.

Results show that simply matching the line endpoints with a point descriptor such as SIFT is quickly limited and confirm the necessity of having a specific descriptor for lines. The small drop in matching accuracy for *Average descriptor* and *NN average* highlights the

	Matching accuracy ↑	
Method	GT lines	GT lines w/ occlusion
SIFT endpoints	0.532	0.403
Average descriptor	0.944	0.754
Nearest Neighbor (NN) average	0.972	0.803
D2-Net sampling	0.969	0.825
ASLFeat sampling	0.963	0.812
Ours (3 samples)	**0.979**	0.813
Ours (5 samples)	0.978	**0.846**
Ours (10 samples)	0.972	0.836

Table 3.5: Ablation study on the Wireframe [Huang et al., 2018] dataset. We compare to various line matching and sampling methods along each line. Ground Truth (GT) lines are used, both without occlusion and with up to 50% occlusion.

importance of keeping ordered points in NW matching. Surprisingly, smarter selections of points along each line such as *D2-Net* and *ASLFeat* sampling perform slightly worse than a regular sampling of points. Finally, there is a trade-off on the number of samples along each line: the NW algorithm loses its benefit when used with few points and the line descriptor becomes less robust to occlusions. On the other hand, many points along the line may produce descriptors that are too close from each other, which makes it harder to correctly discriminate between them. We found that 5 samples is a good trade-off overall, as was also the case for LLD [Vakhitov and Lempitsky, 2019].

3.5.5 Additional Insights

Multi-task learning. The tasks of detecting lines, their junctions, and describing them are diverse, and we assume them to have a different homoscedastic aleatoric uncertainty. Additionally, they can have different orders of magnitude and their relative values are changing during training, in particular when the descriptor branch is added to the pre-trained detector network. Therefore, we chose to use the multi-task loss introduced by Kendall *et al.* [Kendall et al., 2018] and successfully used in other geometrical tasks [Kendall and Cipolla, 2017, Sarlin et al., 2019], to automatically adjust the weights of the losses during training.

The final weights of Equation (8) gracefully converged towards the inverse of each loss, such that the value of each loss multiplied by its weight is around 1. The final weight values are the following: $e^{-w_{junc}} = 7.2$, $e^{-w_{line}} = 16.3$ and $e^{-w_{desc}} = 8.2$. To show the effectiveness of the dynamic weighting, we tried two variants: (1) all loss weights are 1, and (2) we used the final values from the dynamic weighting as static loss weights. In the first case, the detection and description results are worse by at least 10% and 5.5%, respectively. In the second case, the detection and description results are worse by at least 6.7% and 76.2%, respectively.

Visualization of line detection and matches. We provide some visualizations of the line matching in Figure 3.8 and of the line segment detection results in Figure 3.9. Figure 3.10 also offers a comparison of line matches with point matches in challenging images with low texture, and repeated structures. Our method is able to match enough lines to obtain an accurate pose estimation, while point-based methods such as SuperPoint [DeTone et al., 2018] fail in such scenarios.

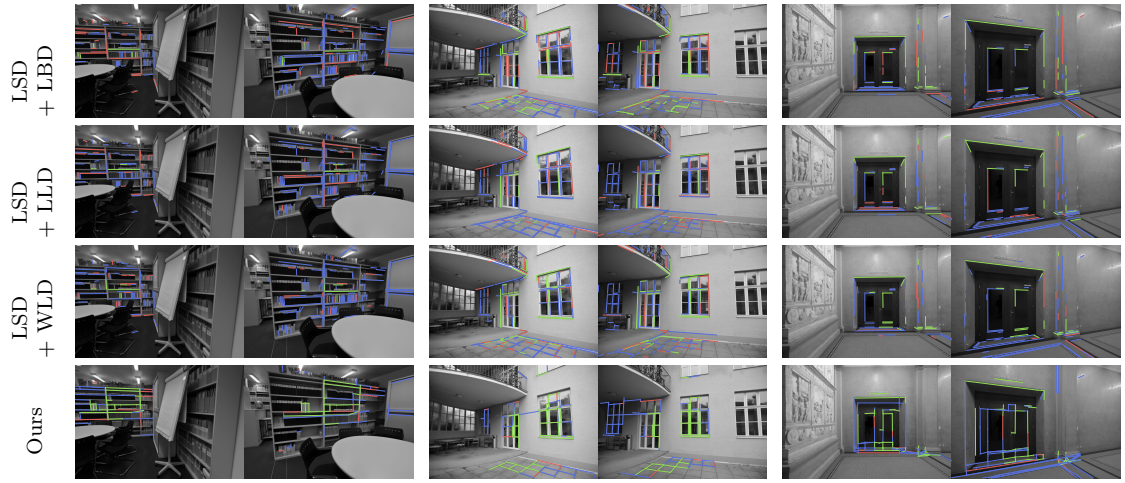


Figure 3.8: Qualitative results of line segment matching. We display line segment matches on the ETH3D dataset [Schöps et al., 2017] with correct matches, wrong matches and unmatched lines. Only lines shared between the two views are shown. Our full pipeline is compared to three line descriptor baselines computed on LSD lines [Von Gioi et al., 2008]: LBD [Zhang and Koch, 2013], LLD [Vakhitov and Lempitsky, 2019] and WLD [Lange et al., 2020].

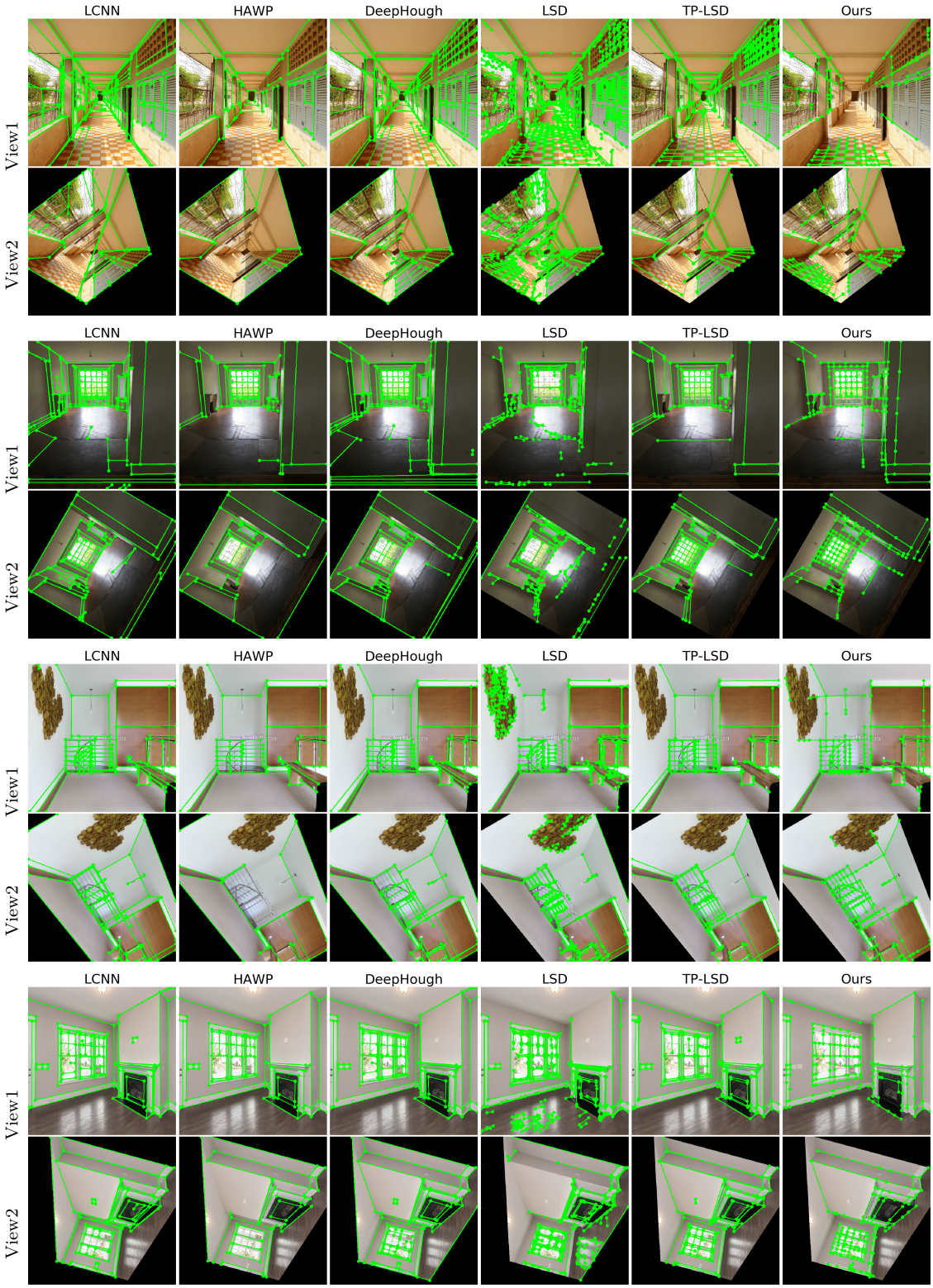


Figure 3.9: Qualitative results of line segment detections. We show examples of line detections on the Wireframe dataset [Huang et al., 2018] for the following methods: LCNN [Zhou et al., 2019a], HAWP [Xue et al., 2020], DeepHough [Lin et al., 2020], LSD [Von Gioi et al., 2008], TP-LSD [Huang et al., 2020] and ours.

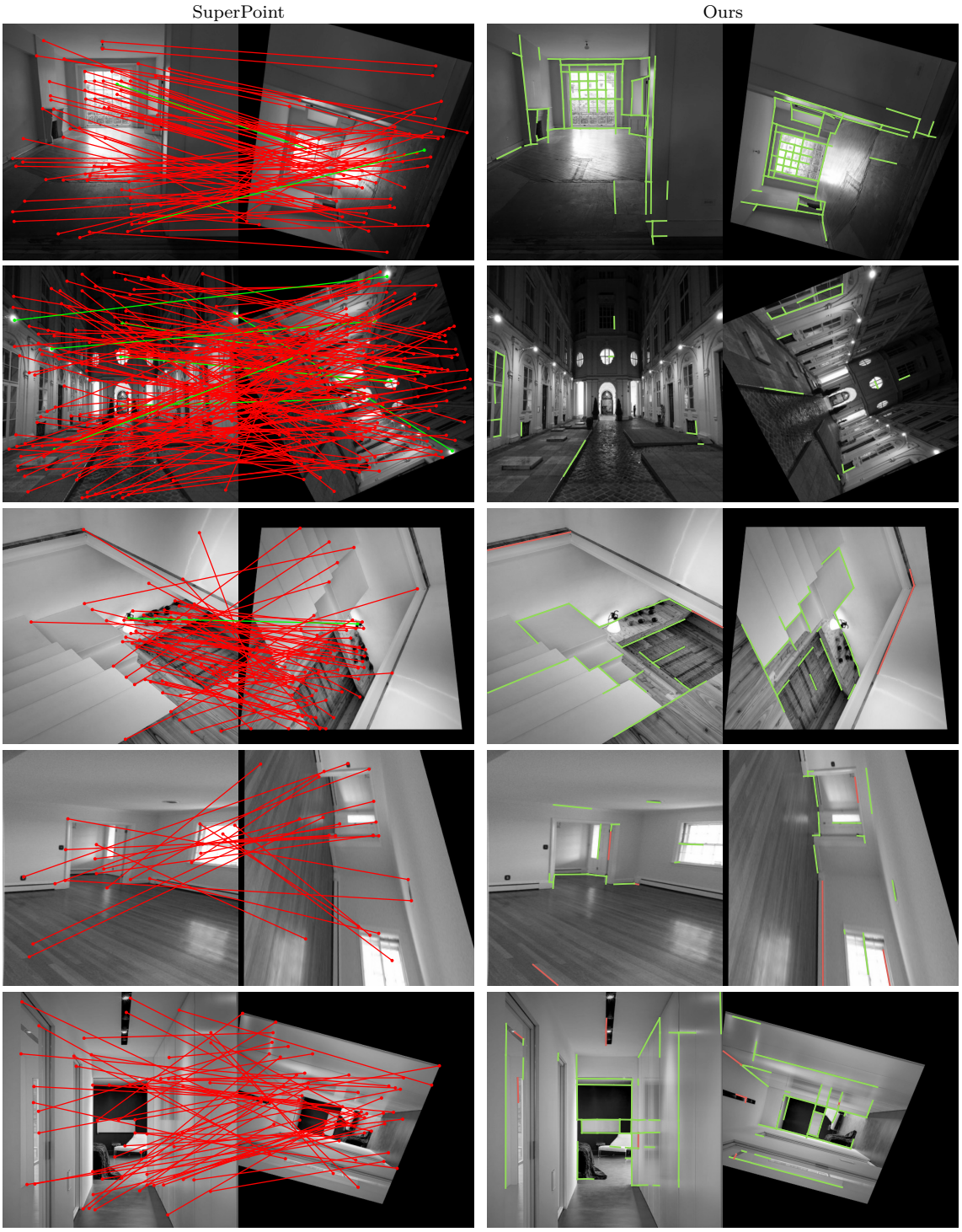


Figure 3.10: Benefits of lines compared to feature points. We compare our method with point matching from SuperPoint [DeTone et al., 2018] on challenging images of the Wireframe dataset [Huang et al., 2018] with correct and wrong matches. We use a distance threshold of 5 pixels to determine if a match is correct, using the orthogonal line distance in the case of lines. Lines can be matched even in the presence of textureless areas, as well as repeated and symmetrical structures.

3.6 Discussion

Summary. We presented the first deep learning pipeline for joint detection and description of line segments in images. Thanks to a self-supervised training scheme, our method can be applied to most image datasets, in contrast with the current learned line detectors limited to hand-labeled wireframe images. Our descriptor and matching procedure addresses common issues in line description by handling partial occlusions and poorly localized line endpoints, while benefiting from the discriminative power of deep feature descriptors. By evaluating our method on a range of indoor and outdoor datasets, we demonstrate an improved repeatability, localization accuracy and matching performance compared to previous baselines.

Limitations and future works. While our line segment predictions are designed to be generic, further work is needed to tune them for specific applications. For instance, line-based localization may prefer short and stable lines, while 3D reconstruction and wireframe parsing may favor longer lines to get a better estimate of the dimensions of the scene. Thanks to our flexible line segment definition, a tuning of the line parameters allows to steer the output segments in one direction or another.
Learning the endpoint position with a network also suffers from the same issue as with feature points, namely a lack of localization accuracy compared to handcrafted methods. Furthermore, the multi-stage training pipeline proposed in this chapter can be cumbersome and could be simplified in the future. We propose in the next chapter another approach to deep line detection to mitigate these challenges.

Chapter 4

DeepLSD: Line Segment Detection and Refinement with Deep Image Gradients

Deep line detectors can inherit the benefits of learned feature points with self-supervision, higher robustness to image changes, and repeatability, but they also inherit their limitations at the same time: poor localization of the predicted endpoints, heavy training pipelines, and a bias towards wireframe lines. Yet, traditional line detectors based on the image gradient were generic, extremely fast, and very accurate, in spite of a lack of robustness in noisy images and challenging conditions. We propose in this chapter to combine traditional and learned approaches to get the best of both worlds: an accurate and robust line detector that can be trained in the wild without ground truth lines. Our new Deep Line Segment Detector (DeepLSD) processes images with a deep network to generate a line attraction field, before converting it to a surrogate image gradient magnitude and angle, which is then fed to any existing handcrafted line detector. Additionally, we propose a new optimization tool to refine line segments based on the attraction field and vanishing points. This refinement improves the accuracy of current deep detectors by a large margin. We demonstrate the performance of our method on low-level line detection metrics, as well as on several downstream tasks using multiple challenging datasets. This chapter originates from the following publication: [Pautrat et al., 2023a], and the corresponding code is available at `https://github.com/cvg/DeepLSD`.

4.1 Motivation

The previous chapter showed that line features can benefit from the recent advances in deep local feature points, through a self-supervised training process, an increased repeatability, and a higher robustness to image changes. Adding line segments to the existing point-based pipelines can then mitigate the lack of structure of feature points and improve the robustness of these pipelines in texture-less areas. However, these benefits brought by lines are hindered by the other drawbacks of feature points: poor localization of the keypoints predicted by deep networks, and cumbersome training pipelines making it hard to apply these new line detectors to new datasets.

In this chapter, we propose to revisit the task of line segment detection in images after learning the lessons of the previous chapters and of previous line detectors. On the one hand, traditional line detectors based on the image gradient are extremely fast and accurate, but lack robustness in noisy images and challenging conditions. On the other hand, their learned counterparts are more repeatable and can handle challenging images, but at the cost of a lower accuracy and a bias towards wireframe lines. We propose here to fuse traditional and learned approaches into a single pipeline to combine the advantages of both approaches. Our method first processes an input image by a deep network supervised by bootstrapping existing line detectors, and then leverage handcrafted line detectors to obtain the line segments. The first step ensures a high robustness to noise, while the second one retains the accuracy of gradient-based detectors. Furthermore, we show that the intermediate output of this first step can be used to refine our predicted lines or any existing line segments with an energy-based optimization. Our final predicted lines are both generic, highly accurate, and robust to image changes, making them suitable for any geometrical tasks. We demonstrate in the following state-of-the-art results for a wide range of applications, such as homography estimation, 3D line reconstruction, point-line visual localization, and vanishing point estimation.

The rest of this chapter consists in an introduction to the problem (Section 4.2), a brief review of similar works (Section 4.3), a description of our new method (Section 4.4), its evaluation against previous line detectors (Section 4.5), and finally a conclusion about the current state of the field of line detection (Section 4.6).

4.2 Introduction

Line segments are ubiquitous in human-made environments and encode the underlying scene structure in a compact way. As such, line features have been used in multiple vision tasks: 3D reconstruction and Structure-from-Motion (SfM) [Hofer et al., 2017, Micusik and Wildenauer, 2017, Mateus et al., 2022, Liu et al., 2023], Simultaneous Localization and Mapping [Gomez-Ojeda et al., 2019, Pumarola et al., 2017, Lange et al., 2019a, Fu et al., 2020a, Zuo et al., 2017], visual localization [Gao et al., 2021], tracking [Quan et al., 2021], vanishing point estimation [Tardif, 2009], etc. Thanks to their spatial extent and presence even in texture-less areas, they offer a good complement to feature points [Gao et al., 2021, Gomez-Ojeda et al., 2019, Pumarola et al., 2017].

All these applications require a robust and accurate detector to extract line features from images. Traditionally, line segments are extracted from the image gradient using handcrafted heuristics, such as in the Line Segment Detector (LSD) [Von Gioi et al., 2008]. These methods are fast and very accurate since they rely on low-level details of the image. However, they can suffer from a lack of robustness in challenging conditions such as in low illumination, where the image gradient is noisy. They also miss global knowledge from the scene and will detect any set of pixels with the same gradient orientation, including uninteresting and noisy lines.

Recently, deep networks offer new possibilities to tackle these drawbacks. This resurgence of line detection methods was initiated by the deep wireframe methods aiming at inferring the line structure of indoor scenes [Huang et al., 2018, Zhou et al., 2019a, Xue et al., 2019, Xue et al., 2020, Lin et al., 2020]. Since then, more generic deep line segment detectors have been proposed [Huang et al., 2020, Li et al., 2021, Dai et al., 2021, Gu et al., 2022, Teplyakov et al., 2022], including joint line detectors and descriptors [Pautrat et al., 2021, Zhang et al., 2021a, Abdellali et al., 2021]. These methods can, in theory, be trained on challenging images and, thus, gain robustness where classical methods fail. As they require a large receptive field to be able to handle the extent of line segments in an image, they can also encode some image context and can distinguish between noisy and relevant lines. On the other hand, most of these methods are *fully* supervised and there exists currently only a single dataset with ground truth lines, the Wireframe dataset [Huang et al., 2018]. Initially designed for wireframe parsing, this dataset is biased towards structural lines and is limited to indoor scenes. Therefore, it is not a suitable training set for generic line detectors, as illustrated in Figure 4.1. Additionally, similarly as with feature points [Sarlin et al., 2021, Lindenberger et al., 2021], current deep detectors are lacking accuracy and are still outperformed by handcrafted methods on easy images. The exact localization of line endpoints is often hard to obtain, as lines can be fragmented and suffer from partial occlusion. Many applications using lines consequently consider infinite lines and ignore the endpoints [Micusik and Wildenauer, 2017].

Based on this assessment, we propose in this chapter to keep the best of both worlds: use deep learning to process the image and discard unnecessary details, then use handcrafted methods to detect the line segments. We thus retain the benefits of deep learning, namely, to abstract the image and gain more robustness to illumination and noise, while at the same time retaining the accuracy of classical methods. We achieve this goal by following the tracks of two previous methods that used a dual representation of line segments with attraction fields [Xue et al., 2019, Xue et al., 2020]. The latter are continuous representations that are well-suited for deep learning, and we show how to leverage them as input to the traditional line detectors. Contrary to these two previous methods, we do not rely on ground truth lines to train our line attraction field, but propose instead to bootstrap existing methods to create a high-quality pseudo ground truth. Thus, our network can be trained on any dataset and be specialized towards specific applications, which we show in our experiments.

We additionally propose a novel optimization procedure to refine the detected line segments. This refinement is based on the attraction field output by the proposed network, as well as on vanishing points, optimized together with the segments. Not only can this optimization

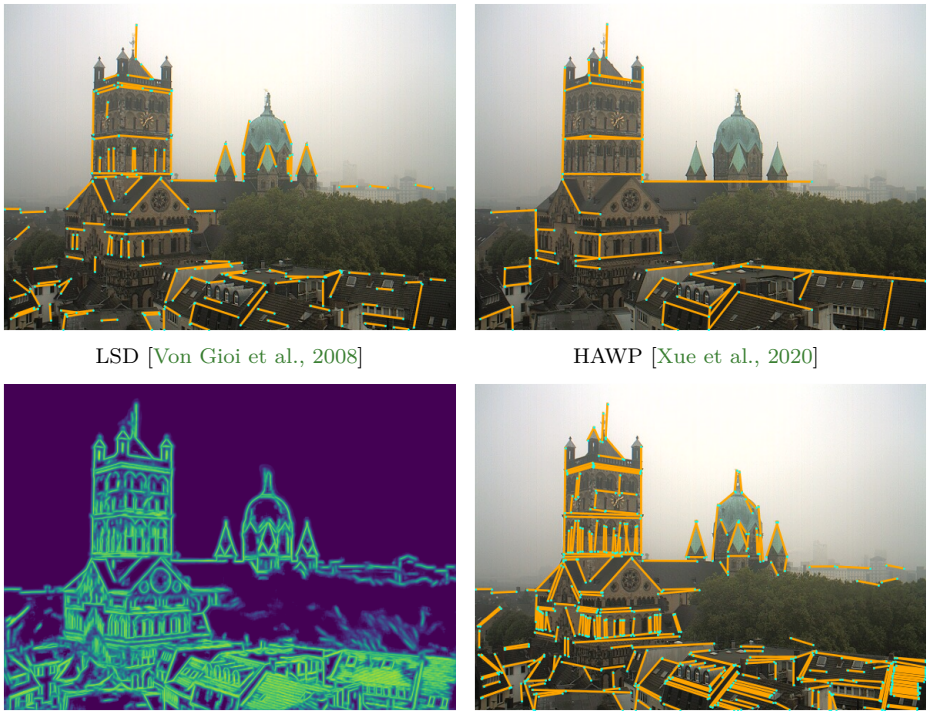


Figure 4.1: Line detection in the wild. Top row: on challenging images, handcrafted methods such as LSD [Von Gioi et al., 2008] suffer from noisy image gradients, while current learned methods like HAWP [Xue et al., 2020] were trained on wireframe images and generalize poorly. **Bottom row:** we combine deep learning to regress a line attraction field and a handcrafted detector to get both accurate and robust lines.

be used to effectively improve the accuracy of our prediction, but it can also be applied to other deep line detectors.

In summary, we propose the following contributions:

- We propose a method **bootstrapping current detectors to create ground truth line attraction fields** on any image.
- We introduce an optimization procedure that can simultaneously **refine line segments and vanishing points**. This optimization can be used as a stand-alone refinement to improve the accuracy of any existing deep line detector.
- We set a new record in several downstream tasks requiring line segments by combining the **robustness of deep learning approaches** with the **precision of handcrafted methods** in a single pipeline.

4.3 Related Work

Handcrafted and learned line detectors. Line detectors in the literature have evolved from handcrafted heuristics based on the image gradient, to deep predictors. An overview of such previous works is available in Section 1.2.2.

Attraction Fields. This chapter proposes to combine deep learning methods with classical line extractors. The key component for this is to use a dual representation of lines through an

attraction field. This representation was first introduced by Xue et al. [Xue et al., 2019] for the wireframe task, and later improved with HAWP [Xue et al., 2020, Xue et al., 2022]. They represent the set of discrete lines of an image with a continuous 2D vector field, suitable for deep networks. We adopt a similar approach, with small modifications to make the prediction more accurate. While not exactly an attraction field, Teplyakov et al. [Teplyakov et al., 2022] also proposed to predict a line mask and line angle field with a network, then used LSD [Von Gioi et al., 2008] to get line segments. Our method obtains better accuracy by predicting a distance field instead of a simple binary mask. Attraction fields have also been leveraged for keypoint detection [Huang et al., 2021], where 2D vectors are voting for the closest keypoint in the image. These detections-by-voting offer a convenient way to represent discrete quantities through continuous ones, and are also a key aspect of our approach when it comes to generating a reliable ground truth for line detection.

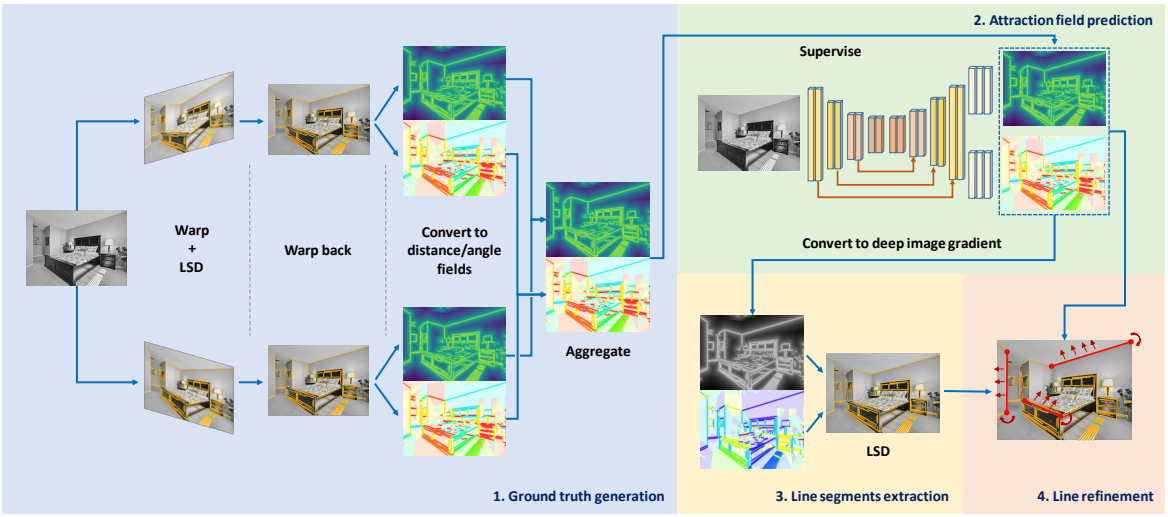


Figure 4.2: Overview of the method. (1) We generate ground truth line Distance Field (DF) and Angle Field (AF) by bootstrapping LSD [Von Gioi et al., 2008]. (2) A deep network is trained to predict the DF/AF, which is then converted to a surrogate image gradient. (3) Line segments are extracted with LSD and (4) refined based on the DF/AF.

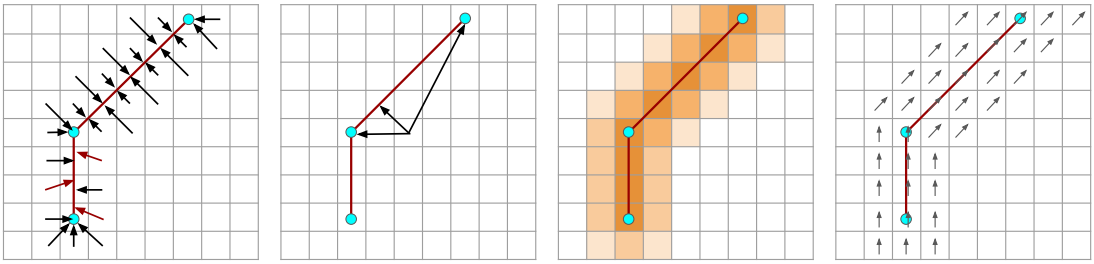
4.4 Hybrid Line Detector

We demonstrate how to combine the robustness of deep networks together with the accuracy of handcrafted line detectors. We train a deep network to predict a line attraction field, convert it to a surrogate image gradient, and feed it to a handcrafted line detector to obtain the segments. Finally, an optimization based on the attraction field is used to refine the lines, as depicted in Figure 4.2.

4.4.1 Line Attraction Field

Representing line segments through an attraction field was first proposed by Xue et al. [Xue et al., 2019]. They initially proposed to regress a 2D vector field for each pixel of an image, indicating the relative position of the closest point on a line. This approach allows to represent discrete quantities (the line segments) as a smooth 2-channel image well suited for deep learning. In [Xue et al., 2020], the authors enriched the attraction field by adding two angles pointing at the endpoints of the closest line. Recovering the original segments from the attraction field is then straightforward.

However, this representation is not optimal to obtain accurate line segments, as illustrated in Figure 4.3. Directly predicting the position of the endpoints as done in HAWP [Xue et al., 2020] requires a larger receptive field to be able to get information from far-away endpoints, so that the network will focus on higher-level details instead of low-level ones. Additionally, deep networks are still struggling to yield accurate keypoint detections [Sarlin et al., 2021, Lindenberger et al., 2021], which holds even more for line endpoints, which are notoriously noisy and unstable. On the contrary, handcrafted methods such as LSD [Von Gioi et al., 2008] are very low-level and gradually grow a line, so that endpoints are recovered only



(a) AFM (b) HAWP (c) Distance field (d) Angle field

Figure 4.3: Attraction field parametrizations. (a) Parametrizing with 2D vectors may produce noisy angles for small vector norms. (b) Adding offsets to the endpoints requires long-range information and is not robust to noisy endpoints. We propose to decouple the distance field (c) and line orientation field (d).

at the end of the region growing process. In this work, we propose to restrict our network to a smaller receptive field and to let the traditional heuristics determine the endpoints.

We adopt a similar attraction field representation as HAWP [Xue et al., 2020] but without the additional two angles pointing at endpoints, yielding only a *line distance field* (DF) and a *line angle field* (AF). For every pixel in these two images, the line distance field $\mathcal{D}$ gives the distance from the current pixel to the closest point on a line, and the line angle field $\mathcal{A}$ returns the orientation of the closest line. These two quantities can be easily obtained from the 2D offset field $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{H \times W} \times \mathbb{R}^{H \times W}$ pointing at the closest point on a line, where (H, W) are the dimension of the image:

$$\mathcal{D} = \sqrt{\mathbf{x}^2 + \mathbf{y}^2}, \qquad \mathcal{A} = \arctan \frac{\mathbf{y}}{\mathbf{x}} + \pi/2 \mod \pi \ . \tag{4.1}$$

We define here the line angle modulo π so that a pixel above or below a line would have the same angle. Adopting this parametrization has the advantage of separating the norm from the angle of the 2D offset. Traditional detectors are leveraging the image gradient magnitude and angle, so we adopt a similar representation. Furthermore, both quantities are continuous close to line segments, and the line angle is even constant close enough to a line.

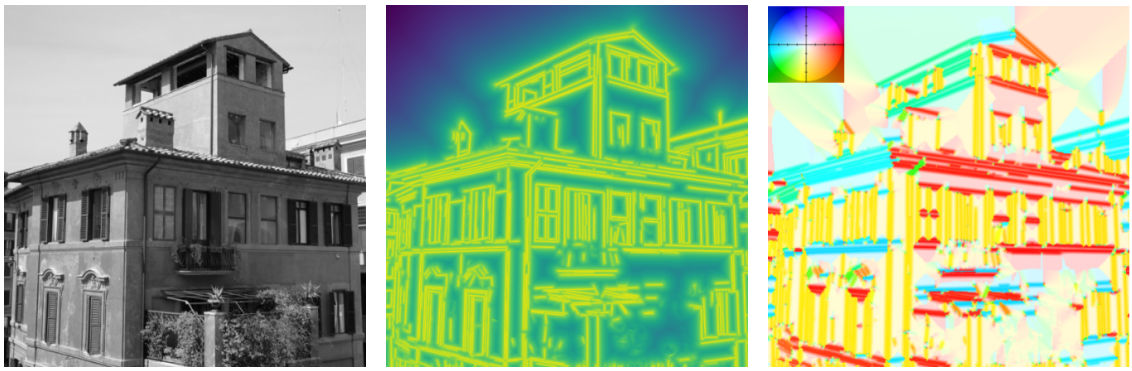
4.4.2 Ground Truth Generation

To learn the attraction field, a ground truth is needed. Both the Attraction Field Map (AFM) [Xue et al., 2019] and HAWP [Xue et al., 2020] are supervised with the ground truth lines of the Wireframe dataset [Huang et al., 2018]. We explore a novel method to acquire our ground truth, by bootstrapping previous line detectors. Inspired by SuperPoint [DeTone et al., 2018] and SOLD2 [Pautrat et al., 2021], we propose to generate the ground truth attraction field through *homography adaptation*. Given a single input image I, we warp it with N random homographies H_i, detect line segments in all the warped images I_i using any existing line detector, and then warp back the segments into I to get a set L_i of lines. We use LSD [Von Gioi et al., 2008] to extract lines as it is currently among the most accurate existing line detector. The next step is to aggregate all the detections together, however, aggregating discrete quantities such as lines is non trivial. In our previous chapter introducing SOLD2 [Pautrat et al., 2021], we proposed to aggregate the endpoints and line heatmaps,

and recover the segments afterwards. Instead, we propose here to convert the sets of lines L_i into a distance field $\mathcal{D}_i$ and angle field $\mathcal{A}_i$, and to aggregate them by taking the median value of each pixel (u, v) across all images:

$$\begin{cases} \mathcal{D}(u,v) = \text{median}_{i\in[1,N]}\mathcal{D}_i(u,v) \\ \mathcal{A}(u,v) = \text{median}_{i\in[1,N]}\mathcal{A}_i(u,v) \end{cases} . \tag{4.2}$$

By taking the median, we remove the noisy lines that were detected in only a few images, as shown in Figure 4.4.



Input image | Distance field | Angle field

Figure 4.4: Pseudo GT visualization. Given an input image, we generate a line distance and angle fields (color coded [Baker et al., 2007]) and use them to supervise a deep network. Noisy lines, such as the ones in the bush at the bottom, are averaged out and ignored.

4.4.3 Learning the Line Attraction Field

To regress our line distance and angle fields, we leverage a UNet-like neural network architecture [Ronneberger et al., 2015]. The input image of size (H, W) is processed by several convolutional layers and gradually downsampled up to a factor of 8 through 3 successive average pooling operations. The features are then upscaled back to the original resolution through another series of convolutional layers and bilinear interpolation. The resulting deep features are then split into two branches, one outputting the distance field $\hat{\mathcal{D}} \in \mathbb{R}^{H\times W}$ and the other one the angle field $\hat{\mathcal{A}} \in \mathbb{R}^{H\times W}$. Figure 4.5 provides an overview of the network.

While all convolutions are followed by ReLU [Agarap, 2018] and Batch Normalization [Ioffe and Szegedy, 2015], the last two outputs have different activations. The angle field is obtained through a sigmoid activation and is multiplied by π to get an angle within $]0, \pi[$. Since the distance field can get very small values close to lines, where we also want the highest accuracy, we adopt a special normalization. The distance field branch ends with a ReLU activation and outputs a normalized distance field $\hat{\mathcal{D}}_n \in (\mathbb{R}^+)^{H\times W}$. The final distance field is obtained through the following denormalization:

$$\hat{\mathcal{D}} = r \cdot e^{-\hat{\mathcal{D}}_n} , \tag{4.3}$$

where r is a parameter in pixels that defines a region around each line. Since handcrafted

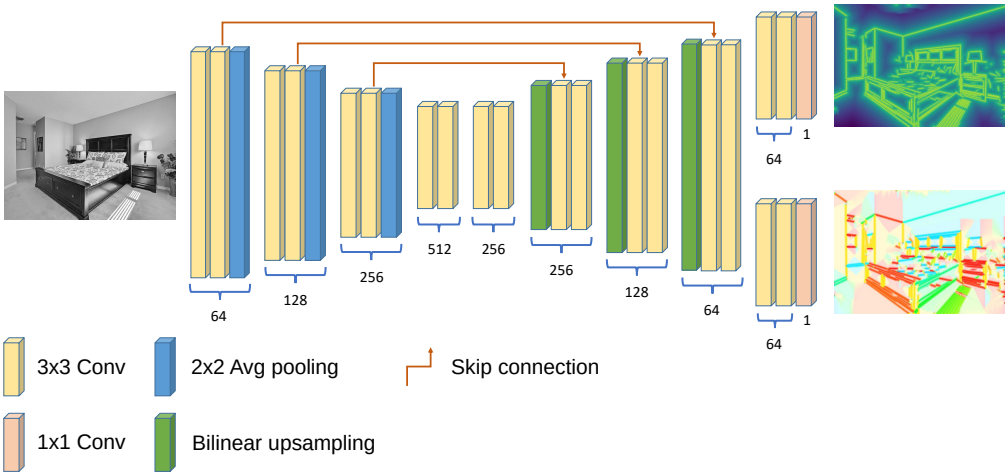


Figure 4.5: Network architecture. We use a standard UNet [Ronneberger et al., 2015] architecture to predict the distance and angle fields.

methods mainly need gradient information close to line segments, we supervise our network only on pixels at a distance of less than r pixels from a line. By selecting a small value for r, large portions of the image may not have any supervision, including areas where the pseudo ground truth was not able to detect real lines, e.g. lines with small contrast. Enforcing these lines to be in the background during training, i.e. with high distance field, provides a detrimental training signal and decreases the recall of the prediction. On the contrary, with our loose supervision, these low contrast lines are not penalized during training and our trained model can detect them, thus yielding a more complete prediction than the ground truth.

We compute the training loss by comparing with a normalized version of the ground truth: $\mathcal{D}_n = -\log\left(\frac{\mathcal{D}}{r}\right)$. Note that since we only supervise pixels with a distance field below r, $\frac{\mathcal{D}}{r} \in [0,1]$ and so $\mathcal{D}_n \in \mathbb{R}^+$. We compute the total loss as the sum of the losses for the distance field and the angular field:

$$\mathcal{L} = \mathcal{L}_D + \mathcal{L}_A \ , \tag{4.4}$$

where $\mathcal{L}_D$ is an L1 loss between the normalized distance fields and $\mathcal{L}_A$ is an L2 angular loss that takes the circularity of the angles into account for the given predicted and ground truth angle fields $\hat{\mathcal{A}}, \mathcal{A} \in [0,\pi]^{H\times W}$:

$$\begin{aligned} \mathcal{L}_D &= ||\hat{\mathcal{D}}_n - \mathcal{D}_n||_1 \ , \\ \mathcal{L}_A &= \min(||\hat{\mathcal{A}} - \mathcal{A}||_2, ||\pi - |\hat{\mathcal{A}} - \mathcal{A}|||_2) \ . \end{aligned} \tag{4.5}$$

4.4.4 Extracting Line Segments

Since handcrafted detectors are based on the image gradient, we propose to convert our distance and angle fields into a surrogate image gradient magnitude $\mathbf{M}$ and angle θ:

$$\begin{cases} \mathbf{M} & = r - \hat{\mathcal{D}} \\ \theta & = \hat{\mathcal{A}} - \frac{\pi}{2} \end{cases} . \tag{4.6}$$

Our predicted angle follows the directions of the lines and is perpendicular to the image gradient, so we rotate it by $\frac{\pi}{2}$. The maximal magnitude of a pixel on a line is r.

An important difference between the approaches of AFM and LSD is the gradient orientation. For an edge separating a dark from a bright area, LSD keeps track of the dark-to-bright gradient direction, while AFM does not. This becomes important when several parallel lines occur next to each other in a dark-bright-dark or bright-dark-bright pattern, as illustrated in Figure 4.6.

(a) A double edge

(b) HAWP [Xue et al., 2020]

(c) Ours

Figure 4.6: Distinguishing double edges. (a) An example of a bright-dark-bright edge and the oriented angle field. (b) Wireframe methods treat it as a single line. (c) We detect it as two lines for better accuracy.

For better accuracy and scale-invariance, we advocate to detect these double edges and make our predicted angle *oriented*, based on the sign of the image gradient angle θ_I:

$$\theta_o = \begin{cases} \theta & \text{if } d(\theta, \theta_I) < d(\theta - \pi, \theta_I) \\ \theta - \pi & \text{otherwise} \end{cases} , \tag{4.7}$$

where $d(\cdot,\cdot)$ is a circular distance between two angles. Now equipped with an oriented angle θ_o and magnitude $\mathbf{M}$, we can directly apply any existing classical line segment detector. Unless stated otherwise, we always use the LSD [Von Gioi et al., 2008] approach in the following, due to its high accuracy. In summary, the purpose of the deep net is to suppress image noise and detect low-contrast lines, while the line segments are accurately extracted by LSD afterwards.

We also add a filtering step, leveraging the DF and AF. We sample n_f points along each line, and compute the fraction p of samples whose distance function is below η_{DF} and angle is close enough to the line orientation with tolerance η_θ. Only the segments with enough inliers are kept.

4.4.5 Line Segment Refinement with Optimization

To make lines even more accurate, we propose an optimization step to refine them by leveraging the predicted DF and AF. This refinement can also be used to enhance the lines of any other detector, and we show in Section 4.5.5 how it can make current deep detectors much more precise.

While lines are detected independently, they usually appear in highly structured configurations in the image. In particular, lines that are parallel in 3D will share vanishing points. We propose to integrate this as soft constraints into our refinement, effectively reducing the degrees of freedom.

We first compute a set of Vanishing Points (VPs) associated with the predicted line segments, using the multi-model fitting algorithm Progressive-X [Barath and Matas, 2019]. We use a strict inlier threshold to be sure to associate only relevant lines to a VP. The optimization is then performed independently for each line and is a weighted unconstrained least square minimization of three different costs:

$$\mathcal{C} = \lambda_A \mathcal{C}_A + \lambda_D \mathcal{C}_D + \lambda_V \mathcal{C}_V \ . \tag{4.8}$$

Given a set P of n_{opt} points uniformly sampled along a line segment l, we denote each point by p_i, the orientation angle of the line as θ_l, and the VP associated with the line as $\mathbf{v}_l$. We use the following three costs:

$$\begin{aligned} \mathcal{C}_A &= \frac{1}{n_{opt}} \sum_{p_i \in P} \left(1 - \left(\cos(\hat{\mathcal{A}}(p_i) - \theta_l)\right)\right) \ , \\ \mathcal{C}_D &= \frac{1}{n_{opt}} \sum_{p_i \in P} \hat{\mathcal{D}}(p_i) \ , \qquad\qquad \mathcal{C}_V = d_{\text{VP}}(l, \mathbf{v}_l) \ , \end{aligned} \tag{4.9}$$

where d_{VP} is a distance measure between a line and a VP. We adopt the perpendicular distance of the line endpoints, projected onto the infinite line passing through the center of the line and the VP, as in [Tardif, 2009]. These objectives are thus minimizing the difference between the sampled angle $\hat{\mathcal{A}}(p_i)$ and the line orientation angle, minimizing the average distance field value over the line, and minimizing the distance between the line and its VP. In case the closest VP is farther away from the line than a threshold t_{VP}, we drop the VP

constraint as it would push the line towards a wrong VP. To avoid lines drifting or collapsing to a single point, we keep the length of the line fixed, and we only optimize the lines over two degrees of freedom: the orientation angle of the line θ_l, and a translation of the middle point in the perpendicular direction of the line.

Since the VPs are already computed, we can even optimize the VPs as well, as a by-product of our approach. Jointly optimizing lines and VPs empirically led to inferior results, mainly because some lines require more refinement than others, so that a global refinement performs worse than independently optimizing the lines. We alternate, instead, between refining the lines and refining the VPs, for a fixed number of iterations k. The VP refinement is performed through a least square minimization of the distance d_{VP} between the VP and all associated lines, and the line-VP association is recomputed after each iteration.

4.4.6 Implementation Details

We train two versions of our network, one indoors on the Wireframe dataset [Huang et al., 2018], but without using the ground truth lines, and one outdoors on MegaDepth [Li and Snavely, 2018]. Given the large size of MegaDepth, we keep 150 scenes for training and 17 for validation, and only sample 50 images from each scene. We use the Adam optimizer [Kingma and Ba, 2014] and an initial learning rate of $1e^{-3}$, which is divided by 10 each time the validation loss reaches a plateau. The training takes roughly 12 hours on a single NVIDIA RTX 2080 GPU. For the line detection, we set the line region r to 5 pixels and ignore magnitudes in $\mathbf{M}$ below 3 when applying LSD. We use $n_f = 50$ samples in the filtering step, $\eta_{\text{DF}} = 1.5$, $\eta_\theta = \frac{\pi}{9}$ and accept lines with more than 50% inliers. The parameters for VP estimation are tuned for each method on a validation set, but the usual threshold t_{VP} ranges from 1 to 2 pixels. The optimization weights are empirically chosen as $\lambda_D = 1$, $\lambda_A = 1$, and $\lambda_V = 0.2$. We adopt $n_{opt} = 10$ samples, perform a fixed set of $k = 5$ alternating iterations, and optimize with Ceres [Agarwal and Mierle, 2012].

4.5 Experiments

To evaluate the performance of our method, we once again cannot use labeled lines as the existing ones are usually biased towards wireframes. We are more interested in evaluating the potential to use these lines for downstream applications, such as homography estimation, 3D line reconstruction, and visual localization.

4.5.1 Evaluation on Low-Level Metrics

We first evaluate our line detection on two challenging datasets to test the robustness of the methods. First, the HPatches dataset [Balntas et al., 2017], consisting of 580 pairs of images with ground truth homographies relating them and varying illumination and viewpoint changes. Second, the RDNIM dataset [Pautrat et al., 2020] (see Appendix A), also with image pairs related by a homography and with challenging day-night variations. We use the night reference in our experiments to get more challenging pairs.

Similarly as in the previous chapter, we assess the *repeatability* and *localization error* metrics. For both metrics, we compute a one-to-one matching of the detected line segments between the two images of a pair using the ground truth homography. For each match, one can then compute the distance between the line in the reference image and the line of the warped image reprojected into the reference frame. We consider two line distance measures: the structural distance evaluating the average distance between the endpoints, and the orthogonal distance measuring the average distance of each endpoint of one line to their orthogonal projection to the other line. Repeatability (Rep) measures the ratio of lines whose match has an error below 3 pixels, and the Localization Error (LE) returns the average distance of the 50 most accurate matches.

We also compute a *homography estimation score*, similarly as in [DeTone et al., 2018]. We first match line segments between the two images, using the Line Band Descriptor (LBD) [Zhang and Koch, 2013]. To estimate the homography, we sample minimal sets of 4 line matches and run LO-RANSAC [Lebeda et al., 2012] for up to 1M iterations, using the orthogonal line distance as reprojection error.

We compare in Tables 4.1 and 4.2 our method to two classical detectors: LSD [Von Gioi et al., 2008] and the ELSED [Suárez et al., 2022]; the best methods using attraction fields: HAWP [Xue et al., 2020], its recent update HAWPv3 trained in a self-supervised way [Xue et al., 2022], and the Line Segment Detector Network (LSDNet) [Teplyakov et al., 2022]: a similar approach as ours combining LSD and a deep network; and two generic deep line detectors: TP-LSD [Huang et al., 2020] and SOLD2 [Pautrat et al., 2021]. We use the implementation of the authors with the biggest model available and default parameters, except for HAWP where we use a threshold of 0.9, as it was not detecting enough lines otherwise. HAWPv3 was trained on ImageNet [Deng et al., 2009]. For LSD, we use the implementation of Rafael Grompone[1] instead of the OpenCV one as it gets much better results. Our method is given without the final optimization in the following, unless otherwise specified.

[1] http://www.ipol.im/pub/art/2012/gjmr-lsd/

		Traditional		Learned				Hybrid	
		LSD	ELSED	HAWP	HAWPv3	TP-LSD	SOLD2	LSDNet	DeepLSD
Struct	Rep ↑	0.314	0.240	0.330	0.272	**0.413**	0.308	0.108	<u>0.367</u>
	LE ↓	<u>1.309</u>	1.551	2.019	2.132	1.500	1.741	2.860	**1.235**
Orth	Rep ↑	<u>0.468</u>	0.465	0.337	0.309	0.444	0.395	0.200	**0.485**
	LE ↓	**0.793**	0.845	1.905	1.937	1.305	1.362	2.285	<u>0.818</u>
H estimation ↑		<u>0.697</u>	0.617	0.260	0.231	0.388	0.421	0.316	**0.705**
# lines / img		492.6	425.4	53.6	82.0	88.6	122.9	172.1	486.2
Time [ms] ↓		104	**10**	61	51	179	334	<u>48</u>	271

Table 4.1: Line detection evaluation on the HPatches dataset [Balntas et al., 2017]. We compare the Repeatability (Rep) and Localization Error (LE) in structural and orthogonal distances, together with homography estimation. We get the best score on homography estimation and a good trade-off between classical and learned methods for the all metrics. The best score is in bold and the second best is underlined.

		Traditional		Learned				Hybrid	
		LSD	ELSED	HAWP	HAWPv3	TP-LSD	SOLD2	LSDNet	DeepLSD
Struct	Rep ↑	0.283	0.209	0.284	<u>0.320</u>	**0.344**	0.307	0.047	0.285
	LE ↓	2.039	2.303	2.206	1.939	<u>1.779</u>	1.879	3.331	**1.733**
Orth	Rep ↑	**0.403**	0.392	0.284	0.354	0.377	0.386	0.130	<u>0.394</u>
	LE ↓	1.369	<u>1.248</u>	2.215	1.704	1.625	1.449	2.752	**1.098**
H estimation ↑		<u>0.468</u>	0.200	0.006	0.026	0.030	0.182	0.027	**0.591**
# lines / img		191.4	112.0	31.6	23.8	24.1	138.2	109.1	400.0
Time [ms] ↓		<u>34</u>	**3**	42	47	75	199	44	96

Table 4.2: Line detection evaluation on the RDNIM dataset [Pautrat et al., 2020]. We compare the Repeatability (Rep) and Localization Error (LE) in structural and orthogonal distances, together with homography estimation. We get the best score on homography estimation and a good trade-off between classical and learned methods for the all metrics. The best score is in bold and the second best is underlined.

From the results, the learned methods, led by TP-LSD [Huang et al., 2020], offer good repeatability, but suffer from a low localization error and inaccurate homography estimation. Handcrafted methods and our method are much more accurate, due to the fact that they do not directly regress the endpoints, but gradually grow the line segments using very low-level details. DeepLSD displays the best improvement over LSD when the changes become the most challenging, i.e. on RDNIM with strong day-night changes. It can significantly improve the localization error and homography estimation score. In spite of having a similar approach as ours, LSDNet [Teplyakov et al., 2022] performs poorly for multiple reasons: they lose accuracy by rescaling images to a fixed low resolution, their line mask is less precise than our distance field, and their training is limited to the Wireframe dataset, while ours can be trained on more diverse images. Overall, our method offers the best trade-off between handcrafted and learned methods and consistently ranks first in the downstream task of homography estimation.

4.5.2 3D Line Reconstruction

The aim of this work is to provide general-purpose lines and as such, the lines generated by DeepLSD should be suitable for 3D reconstruction. We leverage Line3D++ [Hofer et al., 2017] that takes a collection of images with known poses and the associated 2D line segments, and outputs a 3D reconstruction of lines. We propose to compare our method with a few baselines on the first 4 scenes of the Hypersim dataset [Roberts et al., 2021]. This synthetic - but highly realistic - dataset has the advantage of offering a ground truth mesh and 3D model, making it suitable for a quantitative evaluation. Given the ground truth mesh of the scene, we can compute the recall and precision of the 3D lines. Recall is the length in meters of all the portions of lines that are within 5 millimeters from the mesh. High values mean that many lines have been reconstructed. Precision is the percentage of predicted lines that are within 5 millimeters from the mesh. High values indicate that most of the predicted lines are on a real 3D surface.

The results can be seen in Table 4.3. DeepLSD obtains the best recall overall, and second best precision. While TP-LSD [Huang et al., 2020] ranks first in precision, it is able to recover very few lines, as shows its average recall, which is 71% smaller than the one of DeepLSD. Note that DeepLSD is able to reconstruct more lines and with a higher precision than LSD [Von Gioi et al., 2008], the detector that is the most commonly used for line reconstruction [Hofer et al., 2017].

	ai_001_001		ai_001_002		ai_001_003		ai_001_004		Average	
	R	P	R	P	R	P	R	P	R	P
LSD	183.6	95.8	61.8	95.3	**385.0**	88.9	225.3	91.5	213.9	92.9
SOLD2	109.9	94.7	89.3	92.8	62.0	89.0	58.6	89.1	80.0	91.4
HAWPv3	15.8	79.9	15.6	81.0	24.4	68.4	18.5	77.3	18.6	76.7
TP-LSD	68.8	95.3	38.9	94.7	50.7	**98.2**	102.7	**94.3**	65.3	**95.6**
DeepLSD	**204.8**	**96.5**	**89.5**	**98.1**	378.8	88.0	**231.1**	91.9	**226.1**	93.6

Table 4.3: Line 3D reconstruction evaluation. We reconstruct lines in 3D with Line3D++ [Hofer et al., 2017] and evaluate the line length recall in m (R ↑) and precision (P ↑) on the first 4 scenes of Hypersim [Roberts et al., 2021].

We show in Figure 4.7 a qualitative comparison of the 3D line reconstructions of our lines and some baselines for the first 4 scenes of the Hypersim dataset [Roberts et al., 2021]. TP-LSD [Huang et al., 2020] can reconstruct fewer lines as it is trained on wireframe lines only and cannot recover subtle details of the scene. While LSD [Von Gioi et al., 2008] is usually the traditional detector being used for 3D reconstruction [Hofer et al., 2017], the reconstructions produced by DeepLSD are overall more complete and the lines are cleaner compared to the LSD reconstruction. In addition, LSD has a tendency to break segments on higher resolution images, while DeepLSD will detect longer and cleaner lines. Thus, it is easy to merge all lines of a track into a long 3D line for DeepLSD, while LSD will generate a collection of dissociated small segments along the 3D line.

4.5.3 Visual Localization

The 7Scenes dataset [Shotton et al., 2013] is a well-known RGB-D dataset for visual localization, displaying 7 indoor scenes with GT poses and depth. While most scenes are already saturated for point-based localization, the Stairs scene remains very challenging for feature points. Due to the lack of texture and repeated patterns of the stairs, current point-based methods are still struggling on this scene [Brachmann and Rother, 2022]. We thus propose to evaluate our method and previous works on this particular scene, by following the pipeline of hloc [Sarlin et al., 2019, Sarlin, 2020], enriched with line features. As points remain important features, we still detect SuperPoint features [DeTone et al., 2018] and match them with SuperGlue [Sarlin et al., 2020a]. We detect lines with different detectors, and match them between database and query images with the SOLD2 descriptor [Pautrat et al., 2021]. Since depth is available on 7Scenes, we can directly back-project lines in 3D and do not rely on line mapping. In practice, we sample points along each line, un-project them to 3D, and re-fit a line in 3D to these un-projected points. We use the solvers of [Kukelova et al., 2016, Zhou et al., 2018, Larsson, 2020] to generate poses from a minimal set of 3 features (3 points, 2 points and 1 line, 1 point and 2 lines, or 3 lines), then combine them in a hybrid RANSAC implementation [Sattler et al., 2019, Camposeco et al., 2018] to robustly recover the query camera poses. We report the median translation and rotation error, as well as the percentage of successfully recovered poses under various thresholds. This evaluation was implemented with the visual localization pipeline of LIMAP [Liu et al., 2023].

Figure 4.8 shows that DeepLSD obtains the best performance on the challenging scene Stairs. One can highlight the large boost of performance brought by line features compared to using points only. Lines are indeed still present and well localized in indoor environments such as in this scene, and can be matched even when in low-textured scenes. We also provide the results on the full 7Scenes dataset [Shotton et al., 2013] in Table 4.4. Though most scenes are already saturated, DeepLSD remains the best method overall.

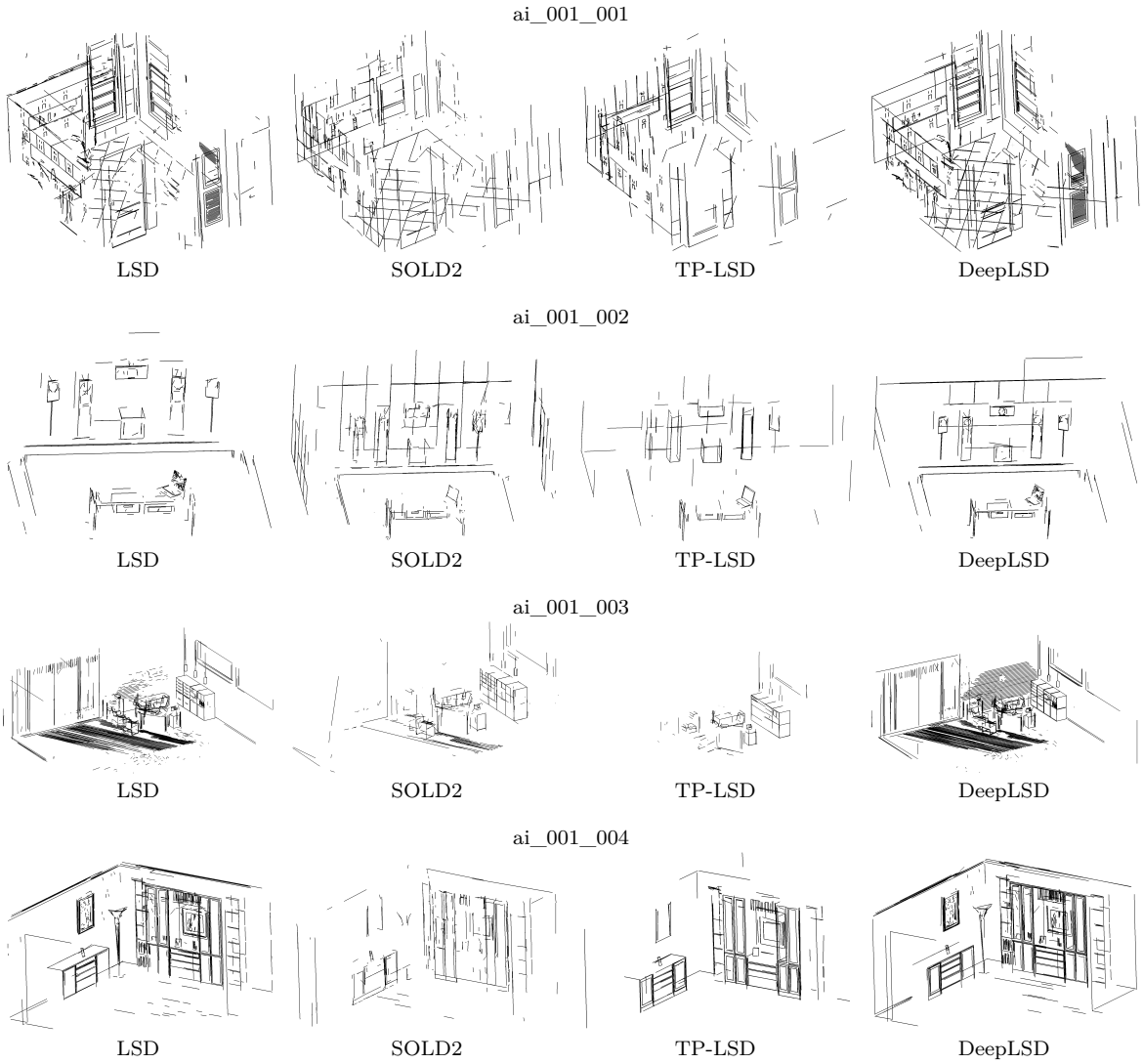
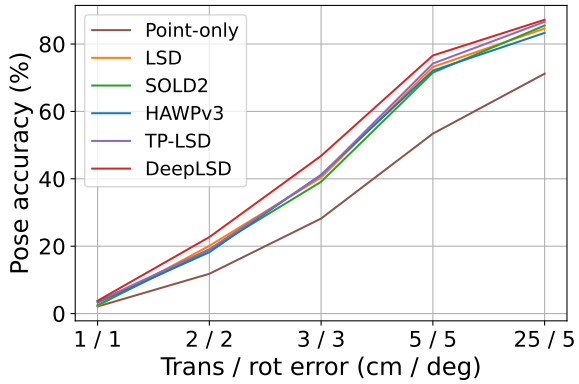


Figure 4.7: Line 3D reconstruction on Hypersim [Roberts et al., 2021]. We leverage the line 3D mapping software Line3D++ [Hofer et al., 2017] on the first 4 scenes of Hypersim [Roberts et al., 2021]. DeepLSD produces more complete and accurate reconstructions than all baselines.

	Point-only SP + SG	LSD	SOLD2	TP-LSD	HAWPv3	DeepLSD
Chess	**2.4** / 0.81 / **94.5**	**2.4** / 0.82 / 94.4	**2.4** / 0.81 / 94.0	**2.4** / **0.80** / 94.4	**2.4** / **0.80** / **94.5**	**2.4** / 0.82 / **94.5**
Fire	1.9 / 0.76 / 96.4	**1.7** / 0.73 / 96.5	1.8 / 0.76 / 95.9	1.8 / 0.76 / 95.8	1.9 / 0.77 / **97.1**	**1.7** / **0.70** / 96.7
Heads	1.1 / 0.74 / 99.0	1.1 / 0.74 / 99.4	1.1 / 0.76 / 99.3	1.1 / **0.73** / **99.5**	1.1 / 0.80 / 99.2	**1.0** / **0.73** / **99.5**
Office	2.7 / 0.83 / 83.9	**2.6** / **0.79** / 84.7	2.7 / 0.82 / 83.8	**2.6** / 0.81 / 84.1	2.7 / 0.82 / 83.8	**2.6** / 0.80 / **85.0**
Pumpkin	4.0 / 1.05 / 62.0	4.0 / 1.04 / 62.1	4.1 / 1.07 / 60.7	4.0 / 1.04 / **62.6**	4.0 / 1.04 / 62.3	**3.9** / **1.02** / 62.2
Redkitchen	3.3 / **1.12** / 72.5	**3.2** / 1.14 / 73.2	**3.2** / **1.12** / **73.5**	**3.2** / **1.12** / 73.2	3.3 / 1.13 / 73.0	**3.2** / 1.13 / 73.4
Stairs	4.7 / 1.25 / 53.4	3.4 / 0.94 / 73.2	3.5 / 0.96 / 71.5	3.4 / 0.93 / 72.1	3.4 / 0.98 / 74.2	**3.1** / **0.85** / **76.6**
Total	2.9 / 0.94 / 80.2	**2.6** / 0.89 / 83.4	2.7 / 0.90 / 82.7	**2.6** / 0.88 / 83.1	2.7 / 0.91 / 83.4	**2.6** / **0.86** / **84.0**

Table 4.4: Visual localization on the full 7Scenes dataset. We report the translation error (in cm) / rotation error (in deg) / pose accuracy at a 5 cm / 5 deg threshold (in %) for the 7 scenes and the average score across all scenes.



	T / R err ↓	Acc ↑
Point-only	4.7 / 1.25	53.4
LSD	3.4 / 0.94	73.2
SOLD2	3.5 / 0.96	71.5
HAWPv3	3.4 / 0.93	72.1
TP-LSD	3.4 / 0.98	74.2
DeepLSD	**3.1 / 0.85**	**76.6**

Figure 4.8: Visual localization on 7Scenes stairs [Shotton et al., 2013]. We evaluate the median translation and rotation errors (cm / deg), the pose accuracy at a 5 cm / 5 deg threshold, and plot the pose accuracy curve for various thresholds.

4.5.4 Vanishing Point Estimation

Another common application for line segments is the Vanishing Point (VP) estimation task. Given the line segments extracted by all the baselines and our method, we apply multi-model fitting with Progressive-X [Barath and Matas, 2019] to find an unconstrained number of (not necessarily orthogonal) VPs. A minimal set of 2 lines provides a VP candidate, and its consistency with the other lines is evaluated under the d_{VP} metric [Tardif, 2009]. This distance is computed as the average orthogonal distance between the endpoints of a line segment and the infinite line going from the VP to the midpoint of the segment. Based on the inlier lines, we do a weighted least squares of the distance of all inliers to the VP, using the line length as weight. We tune the parameters of the model fitting algorithm for each method on a validation set.

We consider two benchmarks for vanishing point estimation. YorkUrbanDB [Denis et al., 2008] pictures 102 images (51 for validation and 51 for test) of urban scenes. It offers 2 or 3 ground truth VPs per image, ground truth lines, and the association between VPs and lines. Additionally, we consider the extended set of VPs proposed in YUD+ [Kluger et al., 2020], which labels up to 8 VPs per image. The second dataset is adapted from the NYU Depth dataset V2 [Nathan Silberman and Fergus, 2012] by [Kluger et al., 2020], consisting of 1449 images (we keep the last 49 for parameter tuning), each labelled with 1 to 8 VPs.

We consider three metrics. *VP consistency* counts the percentage of ground truth lines that are within a given threshold of the predicted VPs [Tardif, 2009]. Each set of ground truth lines is associated to a single predicted VP and each VP can be associated with at most one set of lines. We only show this metric for YorkUrbanDB as NYU does not have manually labelled lines. *VP error* measures how precise the estimated VPs are in 3D. It is the angular error between the directions in 3D of the ground truth VPs and the predicted ones. We perform again a 1:1 matching to optimally assign the predicted VPs to the ground truth ones. For each experiment, we run the VP detection algorithm 20 times and report the median results. *AUC* represents the Area Under the Curve (AUC) of the recall curve of the VPs, as described in [Kluger et al., 2020]. We show the average AUC and its standard deviation over 5 runs.

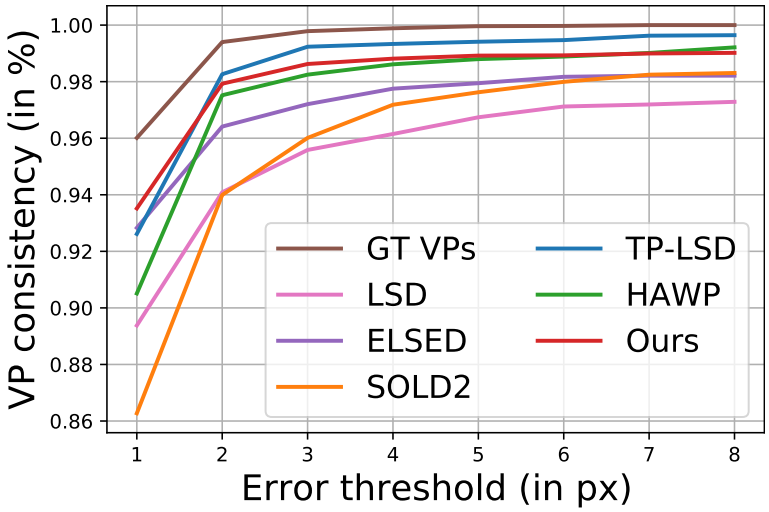


Figure 4.9: VP consistency on the York Urban dataset [Denis et al., 2008]. DeepLSD ranks first on the 1 pixel threshold of VP consistency, meaning that it leads to the largest number of highly accurate VPs.

Results are shown in Figure 4.9 and Table 4.5. The wireframe methods TP-LSD [Huang et al., 2020] and HAWP [Xue et al., 2020] are particularly good for vanishing point estimation, as they only detect structural lines, which are usually the only relevant ones for VP estimation. However, when evaluated on the more challenging and non-Manhattan scenes of NYU-VP, the handcrafted line detectors provide the best accuracy since they can detect all types of lines. Our proposed DeepLSD outperforms all baselines in terms of VP error and AUC, and obtains the most consistent lines with the GT VPs at small thresholds in Figure 4.9.

	YUD+		NYU-VP	
	VP error ↓	AUC ↑	VP error ↓	AUC ↑
LSD	2.05	82.9 (5.3)	3.29	68.6 (6.3)
ELSED	1.88	81.9 (6.0)	**3.24**	68.3 (6.6)
HAWP	1.76	84.2 (4.2)	3.35	68.0 (5.7)
TP-LSD	1.73	85.1 (5.0)	3.35	68.0 (4.5)
SOLD2	2.59	75.4 (6.4)	4.46	56.9 (7.6)
DeepLSD	**1.63**	**85.6** (3.6)	**3.24**	**69.1** (6.2)

Table 4.5: VP estimation on York Urban [Denis et al., 2008] and NYU-VP [Nathan Silberman and Fergus, 2012, Kluger et al., 2020]. We compare DeepLSD with other baselines in terms of median VP error and average recall AUC (and standard deviation). DeepLSD obtains the best performance overall.

4.5.5 Impact of the Line Refinement

We evaluate applying our proposed line refinement as a post-processing step for several learned detection methods. Classical detectors are usually already accurate enough, so that our refinement would not enhance them much. For each method, we compare the raw lines with the lines and VPs optimized by our line optimization. Table 4.6 shows results of line detectors on the 462 images of the test set of the Wireframe dataset [Huang et al., 2018]. The second image is obtained using a synthetic homographic warp of the first image. We use

		Struct		Orth		H estim	# lines / img	Time [ms] ↓
		Rep ↑	LE ↓	Rep ↑	LE ↓			
HAWP	Baseline	0.253	1.34	0.253	1.43	0.701	95.2	**40**
	Opt w/o VP	0.300	1.293	0.399	1.067	0.864		142
	Opt w/ VP	**0.318**	**1.245**	**0.431**	**0.967**	**0.892**		300
TP-LSD	Baseline	0.273	1.379	0.342	1.269	0.658	90.8	**46**
	Opt w/o VP	0.314	1.326	0.470	0.949	0.898		145
	Opt w/ VP	**0.331**	**1.277**	**0.512**	**0.861**	**0.913**		297
SOLD2	Baseline	**0.197**	**1.277**	0.333	0.894	0.848	166.7	**297**
	Opt w/o VP	0.172	1.388	0.339	0.814	**0.935**		426
	Opt w/ VP	0.185	1.330	**0.368**	**0.753**	0.920		697
DeepLSD	Baseline	0.318	0.941	0.489	0.574	0.991	168.8	**68**
	Opt w/o VP	0.314	0.938	0.482	0.575	**0.994**		154
	Opt w/ VP	**0.319**	**0.927**	**0.501**	**0.544**	0.981		542

Table 4.6: Line refinement on the Wireframe dataset [Huang et al., 2018]. We use an error threshold of 1 pixel for the repeatability metrics. The refinement can significantly improve the localization error and homography score of inaccurate methods.

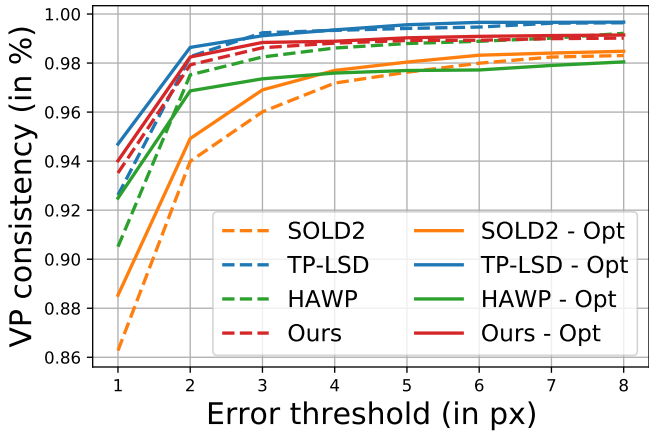
the Wireframe dataset as it has a lot of well-defined vanishing points, which can be leveraged during the optimization. We include results for our proposed optimization with and without the VP constraint to show the increased accuracy with VPs. As we want to highlight the gain in accuracy, we compute repeatability with an error threshold of only 1 pixel.

Results show that the refinement can significantly improve all metrics evaluating the accuracy of the lines, i.e. the localization error and homography estimation. This is particularly true for HAWP [Xue et al., 2020] and TP-LSD [Huang et al., 2020], with a decrease in localization error with orthogonal distance of up to 32% for both, and an improvement of homography score of 27% and 39%. The benefits brought by the refinement are lower for our method, as its raw predicted lines are already sub-pixel accurate and the optimization is limited by the resolution of the DF and AF. Nonetheless, it can slightly improve most metrics. A limitation of this refinement is the execution time, which grows linearly with the number of lines, and requires running two networks.

We additionally study the effect of refinement on the VP estimation task in Figure 4.10. We show again the difference in VP consistency on the YorkUrbanDB dataset [Denis et al., 2008] and VP error on YUD+ [Kluger et al., 2020], with the optimization objective including VPs. Except for HAWP, all methods benefit from the refinement, showing that our refinement can improve the lines as much as their associated VPs.

4.5.6 Ablation Studies

Alternative methods. We validate our design choices on the HPatches dataset [Balntas et al., 2017] with low-level detector metrics. We compare our proposed approach with the same model detecting single edges instead of double ones, our network trained without the DF normalization, and a version of the HAWP [Xue et al., 2020] backbone re-trained on our line GT on the MegaDepth dataset [Li and Snavely, 2018]. Our line GT refers here to the lines obtained by running LSD on our GT distance and angle field. The results of Table 4.7 emphasize the importance of each component. Note that re-training HAWP [Xue et al., 2020] on our lines yields poor results due to the high number of GT lines, and the fact that generic



	VP error ↓
HAWP	**1.76**
HAWP - Opt	1.78
TP-LSD	1.73
TP-LSD - Opt	**1.59**
SOLD2	2.59
SOLD2 - Opt	**2.28**
DeepLSD	1.63
DeepLSD - Opt	**1.59**

Figure 4.10: Effect of the line refinement on VP estimation on YorkUrbanDB [Denis et al., 2008, Kluger et al., 2020]. The line optimization improves the VP consistency and error of most deep methods.

lines have often noisy endpoints, so that predicting an angle to the two endpoints is noisy as well. Figure 4.11 shows two examples of lines detected by the original HAWP, the re-trained version using our GT lines, and DeepLSD.

	Struct		Orth		H estim	# lines / img
	Rep ↑	LE ↓	Rep ↑	LE ↓		
Single edge	0.241	2.121	0.328	1.686	0.434	130.8
No DF normalization	0.344	1.343	0.475	0.879	0.674	439.6
HAWP with our lines	0.209	2.138	0.239	1.840	0.245	98.0
DeepLSD (Ours)	**0.367**	**1.235**	**0.485**	**0.818**	**0.705**	**486.2**

Table 4.7: Ablation study on the HPatches dataset [Balntas et al., 2017]. We compare DeepLSD to alternatives detecting single edges, without DF normalization and with HAWP re-trained on our line GT.

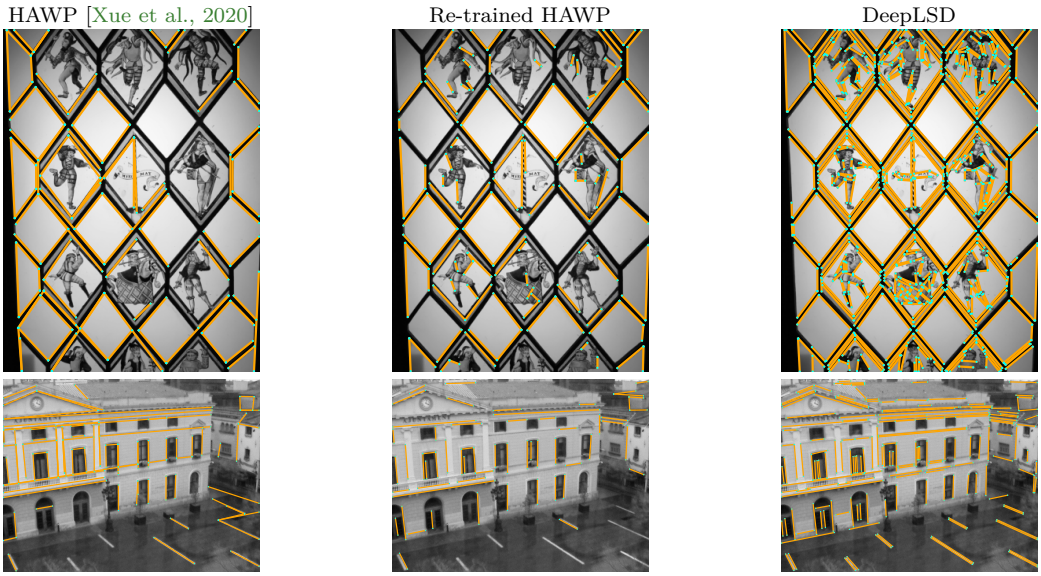


Figure 4.11: Re-training the HAWP detector [Xue et al., 2020] with the proposed pseudo ground truth lines. It yields unsatisfactory lines compared to the DeepLSD approach, mainly because HAWP is not suited to predict high densities of line segments.

			LSD		ELSED	
			Traditional	DeepLSD	Traditional	DeepELSED
HPatches	Struct	Rep ↑	0.314	**0.367**	0.240	**0.263**
		LE ↓	1.309	**1.235**	**1.551**	1.585
	Orth	Rep ↑	0.468	**0.485**	0.465	**0.478**
		LE ↓	**0.793**	0.818	0.845	**0.839**
	H estimation ↑		0.697	**0.705**	0.617	**0.624**
	# lines / img		492.6	486.2	425.4	419.4
	Time [ms] ↓		**104**	271	**10**	144
RDNIM	Struct	Rep ↑	0.283	**0.285**	0.209	**0.230**
		LE ↓	2.039	**1.733**	2.303	**2.258**
	Orth	Rep ↑	**0.403**	0.394	0.392	**0.407**
		LE ↓	1.369	**1.098**	**1.248**	1.361
	H estimation ↑		0.468	**0.591**	0.200	**0.221**
	# lines / img		191.4	400.0	112.0	162
	Time [ms] ↓		**34**	96	**3**	88

Table 4.8: Generalization to other traditional detectors. Our method is not limited to LSD [Von Gioi et al., 2008], but can also be applied to the ELSED [Suárez et al., 2022] line detector for example. We show the comparison between our approach and the original detectors on the HPatches [Balntas et al., 2017] and RDNIM [Pautrat et al., 2020] datasets. Results are given without the final line refinement.

Generalization to other traditional detectors. While DeepLSD is using LSD [Von Gioi et al., 2008] as its base line detector, our approach can be applied to any other traditional detector leveraging the image gradient. We show here the results of our method using ELSED [Suárez et al., 2022] as base detector (coined DeepELSED) and compare it to the original ELSED in Table 4.8. We give the results for the raw lines without any refinement on low-level line detection metrics on the HPatches [Balntas et al., 2017] and RDNIM [Pautrat et al., 2020] datasets. For both traditional detectors LSD and ELSED, our deep version can improve most metrics, thanks to the additional robustness brought by the learned processing of the image.

4.5.7 Visualizations

We provide a visual comparison of our method and the other baselines for line detection in Figure 4.12. We first show line detection examples from the YorkUrbanDB dataset [Denis et al., 2008], picturing indoor and outdoor urban scenes. DeepLSD offers more complete and accurate lines than its competitors. We also compare our method to the other line detectors on some images of the Day-Night Image Matching dataset [Zhou et al., 2016], where DeepLSD provides more lines than the other baselines in challenging scenarios such as night time, over-exposition and low image quality.

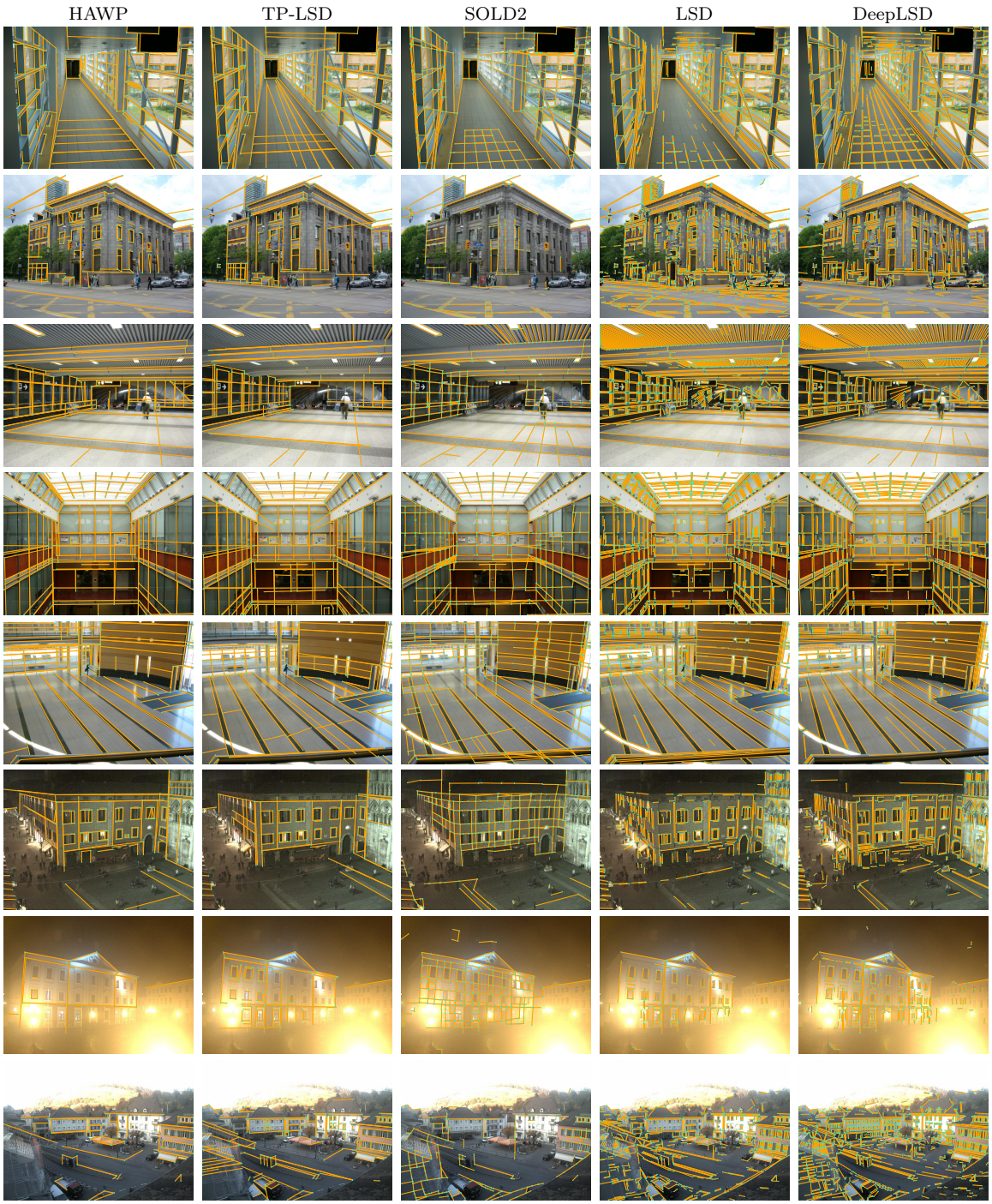


Figure 4.12: Visual comparison of line detectors. First five rows: the lines of DeepLSD (here, without line refinement) are more complete and accurate in urban scenarios (images from the YorkUrbanDB dataset [Denis et al., 2008]). **Last three rows:** when employed in challenging scenarios such as by night, over-exposition and low image quality, DeepLSD can detect more relevant lines than the other baselines (images from the Day-Night Image Matching (DNIM) dataset [Zhou et al., 2016]).

4.5.8 Limitations

Even though DeepLSD can produce repeatable and accurate lines by taking advantage of the benefits of both traditional and learned methods, it still suffers from a few limitations:

- The current approach of running a deep network, followed by handcrafted heuristics and line optimization is not fully differentiable. Making the full pipeline differentiable would mean making LSD differentiable, which is unclear how to do it. An end-to-end pipeline would certainly provide better training signals to the deep network processing the image.

- The generation of the pseudo ground truth lines is still limited by the performance of LSD [Von Gioi et al., 2008]. If a line is almost never detected by LSD during homography adaptation, it will most likely not be detected in the ground truth attraction field. Similarly, a noisy but repeatable line will be kept in the pseudo ground truth. One way to overcome this issue could be to leverage the trained DeepLSD to re-generate a new pseudo ground truth with less noise, as was done in SuperPoint [DeTone et al., 2018].

- In spite of our efforts to make the pseudo ground truth as clean as possible, there is always a trade-off between detecting all low-contrast lines and avoiding to detect noisy lines in the background. For example, DeepLSD misses some good lines at the bottom right of the image in the 5th row of Figure 4.12 and is also detecting some noisy lines in the sky of the image in the 7th row. We can influence this trade-off in two ways. First, by tuning the aggregation of the attraction field when generating the ground truth. We currently take the median value of the distance and angle fields, but one could also take a given percentile, to allow more or less outlier values. Second, one can enforce more or less constraints to the distance field for background areas. Enforcing a high distance field for pixels far away from the ground truth lines will reduce the number of noisy lines in the background, but will also ignore the lines with low contrast. The parameters proposed in this paper are the ones visually yielding the best trade-off between the two.

- Though the input image is processed through a deep network, there is still no proper semantic understanding of the detected lines, so that DeepLSD will detect any kind of lines. Depending on the application, one could imagine adding some semantic filtering in the ground truth generation to keep only a specific kind of lines (e.g. avoiding lines in the sky or on dynamic objects such as humans).

- The proposed line refinement is for now rather slow, especially when it is applied to other deep line detectors, as it requires running two networks. However, we believe that it is still valuable for applications that can run offline and that require high precision, such as for 3D reconstruction. Our current implementation can also certainly be optimized, and our network compressed to run on embedded devices, without sacrificing too much performance.

4.6 Discussion

Summary. We presented in this chapter a hybrid line segment detector combining the robustness of deep learning and the accuracy of handcrafted detectors, using a learned surrogate image gradient as intermediate representation. Without the requirement of ground truth lines, our method can be trained on any dataset and is suitable for most geometrical tasks including line segments. Finally, we proposed a line refinement able to improve the accuracy of our method and to bridge the gap in line localization between deep line detectors and handcrafted ones. We believe that our general-purpose lines will open new possibilities to use line segments in the wild.

Limitations and future works. Line segment detection can now be performed in an accurate and robust way, thanks to DeepLSD. However, this method is not perfect, and the previous section lists some of the limitations and paths to explore to further improve our approach. In particular, DeepLSD offers generic lines, that are designed to be suitable for as many applications as possible, but some tasks may require specific types of lines (e.g. wireframes, no texture lines, etc). Thus, processing the ground truth to only keep the relevant lines may be necessary.

While the use of lines in the wild is now possible, matching lines across frames remains a challenge. The previous chapter explored the issues related to line matching and proposed a first solution, but the current line matchers remain nevertheless inferior to the point matchers. Additionally, we saw in this chapter that points are still necessary to complement lines, for example in visual localization (Section 4.5.3), and in 3D reconstruction, where points can solve some of the degenerated configurations for line triangulation [Liu et al., 2023]. Therefore, we propose in the next chapter to combine point and line features in a single matcher, and we show how these combined features can significantly improve geometrical tasks.

Part III

Combining Point and Line Features

Chapter 5

GlueStick: Robust Image Matching by Sticking Points and Lines Together

While complementary to each other, point and line features are traditionally detected and matched independently. Yet, processing them together is more efficient and unlocks some synergies improving the overall performance. This chapter introduces a new matching paradigm, where points, lines, and their descriptors are unified into a single wireframe structure. We propose GlueStick, a deep matching Graph Neural Network (GNN) that takes two wireframes from different images and leverages the connectivity information between nodes to better glue them together. In addition to the increased efficiency brought by the joint matching, we also demonstrate a large boost of performance when leveraging the complementary nature of these two features in a single architecture. We show that our matching strategy outperforms the state-of-the-art approaches independently matching line segments and points for a wide variety of datasets and tasks. The content of this chapter stems from the publication [Pautrat et al., 2023c], and the code for GlueStick is available at `https://github.com/cvg/GlueStick`.

5.1 Motivation

As described in the previous two chapters, line segments are powerful features complementary to points. They offer structural cues, robust to drastic viewpoint and illumination changes, and can be present even in texture-less areas. However, describing and matching them is more challenging compared to points due to partial occlusions, lack of texture, or repetitiveness. Furthermore, points and lines are classically detected, described, and matched through a multitude of different algorithms, making the existing point-line pipelines inefficient and redundant. Both features indeed share similar characteristics, which could be acquired simultaneously.

In this chapter, we revisit the task of feature matching by jointly matching points and lines in a single network. In addition to being more efficient, this allows the matching network to reason about the inter-connectivity between points and lines, and to leverage additional information compared to previous matchers. On the one hand, connecting points with lines brings more structure and connectivity information to the point matching task. On the other hand, the latter can in turn guide the line matching to disambiguate the challenging cases where line descriptors are struggling because of partial occlusion or homogeneous areas. Furthermore, both features appear in different scenarios, such as richly textured areas for points, and low textured surfaces for lines, so that their combination can be applied to any given situation. We show in the following that points and lines can be combined in most tasks, such as homography and pose estimation, image stitching, and 3D reconstruction, and that their combination brings a new state of the art in local feature matching.

This chapter is structured as follows: it starts with an introduction to the topic (Section 5.2), reminds the reader with previous work related to it (Section 5.3), introduces our novel point-line matcher (Section 5.4), tests it against previous point and line matchers (Section 5.5), and finally closes with a discussion about what remains to be done (Section 5.6).

5.2 Introduction

As seen in the previous chapters, line segments are high-level geometric structures useful in a wide range of computer vision tasks such as SLAM [Gomez-Ojeda et al., 2019, Zuo et al., 2017, Pumarola et al., 2017], pose estimation [Xu et al., 2017], construction monitoring [Kropp et al., 2018, Asadi et al., 2019], and 3D reconstruction [Hofer et al., 2017, Zeng et al., 2020, Zhou et al., 2019b]. Lines are ubiquitous in structured scenes and offer stronger constraints than feature points. In particular, lines shine in low-textured scenes where point-based approaches struggle.

However, compared to keypoints, line segments are often poorly localized in the image and suffer from lower repeatability. Line segments are also more challenging to describe since they can cover a large spatial extent in the image and suffer from occlusions and perspective effects due to viewpoint changes. Furthermore, lines often appear as part of repetitive structures in human-made environments, making classical descriptor-based matching fail. For this reason, typical matching heuristics such as mutual nearest neighbor and Lowe's ratio test [Lowe,

2004] are often less effective for lines.

Recently, deep learning has ushered in a new paradigm for feature point matching using a Graph Neural Network (GNN) [Sarlin et al., 2020a, Sun et al., 2021]. This new approach bypasses the need for matching heuristics or even outlier removal techniques, thanks to the high precision of the predicted matches [Sarlin et al., 2020a]. A key component to achieve this is to leverage the positional encoding of keypoints directly in the network and to let it combine visual features with geometric information [Sarlin et al., 2020a, Sun et al., 2021, Truong et al., 2021, Jiang et al., 2021]. Letting the GNN reason with all features simultaneously brings in additional context and can disambiguate repetitive structures, as shown in Figure 5.1.

Even though recent advances leveraged similar ideas to enrich line descriptors [Yoon and Kim, 2021], directly transferring this GNN approach to line matching is not trivial. The large extent of lines and their lack of repeatability make it hard to find a good feature representation for them. In this paper, we take inspiration from SuperGlue [Sarlin et al., 2020a] and introduce GlueStick, to jointly match keypoints and line segments. Our goal is to leverage their complementary nature in the matching process. By processing them together in a single GNN, the network can learn to resolve ambiguous line matches by considering nearby distinctive keypoints, and vice versa. We propose to leverage the connectivity between points and lines via a unified wireframe structure, effectively replacing previous handcrafted heuristics for line matching [Schmid and Zisserman, 1997a, Zhang and Koch, 2013, Li et al., 2016c] by a data-driven approach.

Our network takes as input sparse keypoints, lines, and their descriptors extracted from an image pair, and outputs a set of locally discriminative descriptors enriched with the context from all features in both images, before establishing the final matches. Inside the network, keypoints and line endpoints are represented as nodes of a wireframe. The network is composed of self-attention layers between nodes, cross-attention layers exchanging information across the two images, and a new line message passing module enabling communication between neighboring nodes of the wireframe. After the GNN, points and lines are split into two separate matching matrices and a dual-softmax is used to find the final assignment of the features. Overall, our contributions are as follows:

- We replace heuristic geometric strategies for line matching with a **data-driven approach**, by **jointly** matching points and lines within a single network.
- We offer a novel architecture **exploiting the local connectivity** of the features within an image.
- We experimentally show **large improvements** of our method over previous state-of-the-art point and line matchers on a wide range of datasets and tasks.

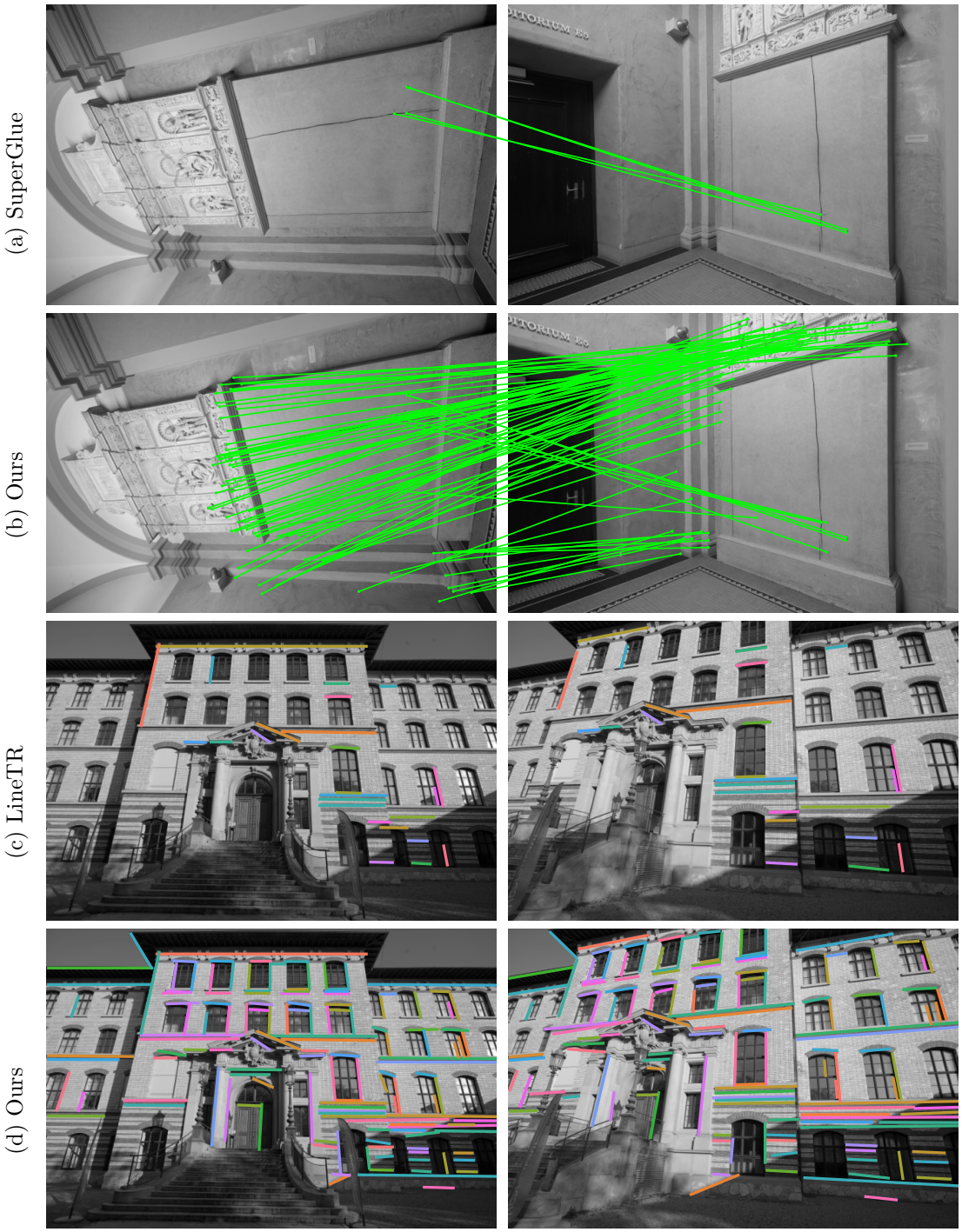


Figure 5.1: Joint matching of points and lines. Matching feature points often fails in textureless areas (a), while current line matching methods struggle with large viewpoint changes (c). We propose GlueStick, a network jointly matching points and lines. While none of the methods were trained on the rotations of (a)(b), line matches can guide GlueStick while SuperGlue [Sarlin et al., 2020a] fails, and vice-versa in (d), where points can complement the line matching. For clarity reasons, we show here only the correct matches.

5.3 Related Work

Line detection and description. We refer the reader to Section 1.2.2 for an overview of the existing line features.

Combination of points and lines. Section 1.2.3 reviews the existing applications combining point and line features.

Matching with transformers. SuperGlue [Sarlin et al., 2020a] uses a GNN to process keypoints and their descriptors from two input images, adding a positional encoding to better disambiguate repetitive patterns. Several variations of this method have been proposed later, with higher efficiency [Chen et al., 2021, Shi et al., 2022, Lindenberger et al., 2023] and with dense predictions [Sun et al., 2021, Truong et al., 2021, Jiang et al., 2021, Wang et al., 2022, Chen et al., 2022, Edstedt et al., 2023].

[Ma et al., 2021] combine a CNN and a GNN to match line segments, but without feature points. In the GNN, each line is represented with a single node, and the assignation is solved using a single Sinkhorn matrix. LineTR [Yoon and Kim, 2021] proposes to use attention inside points sampled for each line to deal with the line scale changes and occlusions. The Hybrid Descriptor for Points and Lines (HDPL) [Guo et al., 2021] mixes points and lines in the same GNN, each line being represented with a single node in the GNN. They only use a single Sinkhorn matrix, allowing point-line assignments.

In contrast to these methods, we model each line segment endpoint as a separate node in the GNN. The endpoints are, in most cases, consistent with the underlying epipolar geometry, allowing the network to leverage both points and lines to disambiguate the matching. In our ablation study, we show that matching points and line endpoints together already greatly improves the matching performance.

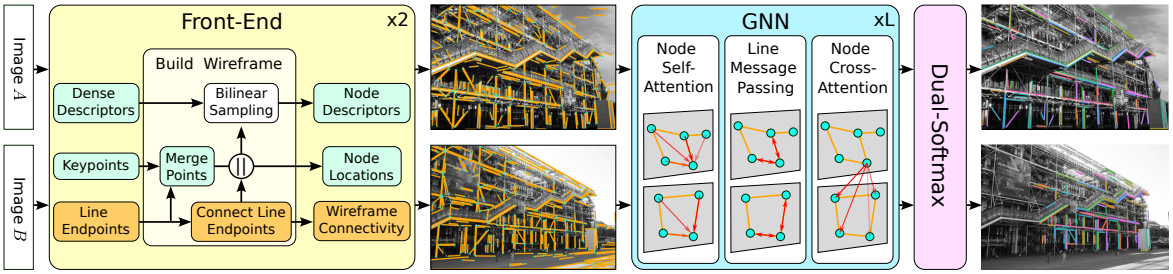


Figure 5.2: Overview of GlueStick. Keypoints, dense descriptors, and lines are extracted from two images, and unified into two wireframes (front-end). We then enrich the features of the nodes of both wireframes via self, line, and cross-attention inside a Graph Neural Network (GNN). Finally, points and lines are matched separately via two dual-softmax modules.

5.4 GlueStick: a Joint Point-Line Matcher

In this section, we show how to combine points and lines within the same network. The motivation for this is that each feature can leverage cues from the neighbouring features to improve the matching performance. For example, a line using the surrounding points or vice-versa. Furthermore, the network can automatically discover combinations of points and lines that are useful for matching, instead of heuristically mining them as in previous works [Li et al., 2016c]. Our architecture, displayed in Figure 5.2, consists in three blocks:

1. **Front-End**: We extract points, lines, and their descriptors with common feature detectors, then combine them into a single wireframe (Sec. 5.4.1).

2. **GNN**: The goal of this block, described in Sec. 5.4.2, is to combine the visual and spatial information of each feature, and to allow interaction between all features, regardless of their original receptive field. The output is a set of updated descriptors, enriched by the knowledge of relevant features within and across images, as well as within nodes connected by a line segment.

3. **Dual-Softmax**: The final assignation is solved separately for points and lines, using two independent dual-softmax modules [Rocco et al., 2018, Sun et al., 2021], as detailed in Sec. 5.4.3.

5.4.1 From Points and Lines to Wireframes

The input to our GNN is a set of points, their associated local descriptors, and a connectivity matrix indicating which points are connected by a line. The first step is to establish this connectivity and build the wireframe graph.

We use SuperPoint (SP) [DeTone et al., 2018] to predict keypoints and a dense descriptor map, and we detect segments with the general-purpose LSD [Von Gioi et al., 2008] detector. Keypoints located close to line endpoints are redundant, so we remove SP keypoints that are within a small distance d to existing line endpoints.

Furthermore, line segments generated by generic detectors such as LSD are usually disconnected. To give more structure to the input and to explicitly encourage the network to reason in terms of line connectivity, we merge close-by endpoints, again with a distance

threshold d. This process lifts the unstructured line cloud into an interconnected wireframe. After this step, each keypoint and line endpoint is represented as a node in the wireframe, with different connectivities for each node: 0 for an isolated keypoint, 2 for a corner, etc. We then interpolate the dense SP feature map at the node locations to equip them with a visual descriptor. Note that this endpoint merging is modifying the position of the endpoints but not the number of lines. For downstream tasks requiring high precision, we use the original position of the endpoints, to keep the sub-pixel accuracy of the original detector.

5.4.2 Attention-based Graph Neural Network

A key part of our method is the GNN, which aggregates visual and spatial information to predict a set of *enriched* feature descriptors, that are used to establish the final matches via descriptor similarity. Within the network, each node (either a keypoint, or a line endpoint) is associated with an initial descriptor that is based on the visual appearance as well as the position in the image.

Let A and B be a pair of images. For each image, the inputs of the network are: a set of nodes $\mathbf{p}$, with coordinates $(x_p,\, y_p)$, confidence score s_p and visual descriptors $\mathbf{d}^{vis} \in \mathbb{R}^D$; and a set of line segments $\mathbf{l}$ defined as a pair of nodes $(x_p,\, y_p)$ and $(x'_p,\, y'_p)$, and with a line score s_l. This line score can be any value returned by the line detector indicating the quality of the line, or simply the length of the line to put more emphasis on longer lines. The node score s_p is either coming from the keypoint detector, or is equal to s_l when it is a line endpoint.

Positional and Directional Encoding. The first step is to encode the spatial information of each feature. To this end, we learn two Positional Encoder (PE^p and PE^e) with Multi-Layer Perceptron (MLP) that generate a *spatial* descriptor $\mathbf{d}^p$ for each node and an *edge*-descriptor $\mathbf{d}^e$ for each line segment originating from this node. A node with connectivity 3 will for instance get assigned one $\mathbf{d}^p$ and 3 $\mathbf{d}^e$ (one for each outgoing line segment). The edge-positional encoding takes as additional information the offset to the other endpoint of its line segment, allowing it to have access to the angle and length of the line segment:

$$\begin{aligned} \mathbf{d}^p &= \mathrm{PE}^p([x_p, y_p, s_p]^\top) \ , \\ \mathbf{d}^e &= \mathrm{PE}^e([x_p, y_p, x'_p - x_p, y'_p - y_p, s_l]^\top) \ . \end{aligned} \tag{5.1}$$

The spatial-descriptor $\mathbf{d}^p$ is used to initialize the node features, while the edge-descriptors $\mathbf{d}^e$ are used in the line message passing (see below).

Network Architecture. Our GNN is a complete graph with three types of undirected edges (See Figure 5.3). Self-attention edges $\mathcal{E}_{\text{self}}$, connect nodes of one image with all the nodes of the same image. Line edges $\mathcal{E}_{\text{line}}$, connect nodes that are endpoints of the same line. Cross attention edges $\mathcal{E}_{\text{cross}}$, connect nodes of one image to the other image nodes.

A node i is initially assigned a feature descriptor fusing its spatial and visual information: ${}^{(0)}\mathbf{x}_i = \mathbf{d}_i^p + \mathbf{d}_i^{vis}$. This node descriptor is then iteratively enriched and refined with the context of all the other descriptors in L iterations of Self, Line, and Cross layers. Finally, the features of each node are linearly projected to obtain the output features. The next paragraphs detail each type of layer.

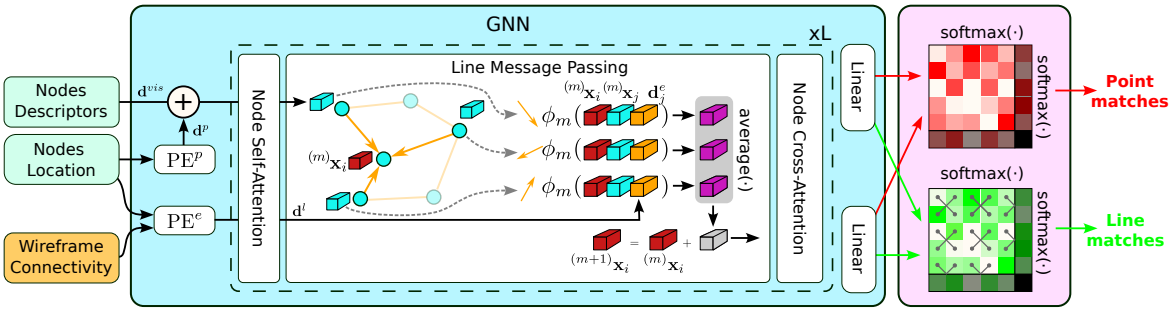


Figure 5.3: Graph Neural Network (GNN) architecture. Node features of the wireframe are enriched via several communication layers. Our Line Message Passing (LMP) exchanges information between neighboring nodes that are connected together.

Self and Cross Layers. $\mathcal{E}_{\text{self}}$ and $\mathcal{E}_{\text{cross}}$ edges are similarly defined as in [Sarlin et al., 2020a]. The m-th feature update is defined by a residual message passing:

$$ {}^{(m+1)}\mathbf{x}_i = {}^{(m)}\mathbf{x}_i + \psi_m \left(\left[{}^{(m)}\mathbf{x}_i || a_m({}^{(m)}\mathbf{x}_i; \mathcal{E}) \right] \right) \ , \tag{5.2} $$

where $||$ denotes concatenation, the function ψ_m is modeled with an MLP, and $a_m({}^{(m)}\mathbf{x}_i; \mathcal{E})$ is the Multi-Head Attention mechanism from [Vaswani et al., 2017] applied to the set of edges $\mathcal{E}$:

$$ a_m(\mathbf{x}_i; \mathcal{E}) = \sum_{j:(i,j)\in\mathcal{E}} \text{softmax}_j \left(\frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{D}} \right) \mathbf{v}_j \ , \tag{5.3} $$

where the keys $\mathbf{k}_j$, queries $\mathbf{q}_i$, and values $\mathbf{v}_j$ are computed as linear projections of the node features $\mathbf{x}_i$ and $\mathbf{x}_j$. In self-attention layers, $\mathbf{k}_j$ and $\mathbf{v}_j$ will come from the same image, whereas in cross-attention they will come from the other image. Self-attention allows the network to leverage the context of the full image, and to resolve repetitive structures. Cross-attention moves corresponding features closer in descriptor space and can search for similar node structures in the other image to fully leverage spatial information.

Line Message Passing. We describe here our novel LMP transferring information across the line edges $\mathcal{E}_{\text{line}}$. By connecting line segments in a wireframe structure, we allow the i-th node to leverage the local edge connectivity to the set $\mathcal{N}_i$ of neighboring nodes, and to look for the same type of connectivities in the other image. This mechanism is enabled by the m-th LMP update which aggregates the information contained in the two endpoint features ${}^{(m)}\mathbf{x}_i$ and ${}^{(m)}\mathbf{x}_j$ and the corresponding endpoint positional encoding $\mathbf{d}_j^e$:

$$ {}^{(m+1)}\mathbf{x}_i = {}^{(m)}\mathbf{x}_i + \sum_{j\in\mathcal{N}_i} \frac{\phi_m([{}^{(m)}\mathbf{x}_i || {}^{(m)}\mathbf{x}_j || \mathbf{d}_j^e])}{|\mathcal{N}_i|} \ , \tag{5.4} $$

where ϕ_m denotes again an MLP and $|\mathcal{N}_i|$ is the number of neighbors of node i. We use here a simple average across all neighbors. An attention mechanism could also have been applied, but we empirically found that it only increased the complexity of the model, for no gain in performance.

5.4.3 Dual-Softmax for Points and Lines

Recent works [Rocco et al., 2018, Sun et al., 2021] show that dual-softmax approach obtains similar or better results than the usual Sinkhorn algorithm [Sinkhorn and Knopp, 1967, Sarlin et al., 2020a], being also more efficient. We observed similar behaviour in our experiments and opted for the dual-softmax assignment. GlueStick provides both point and line matches in a single forward pass. We match nodes and lines separately through two independent dual-softmax assignments. On the one hand, all nodes (keypoints and line endpoints) are matched against each other using the final features output by the GNN: $\mathbf{f}_i^A \in \mathbb{R}^D$ for node i in image A and $\mathbf{f}_j^B \in \mathbb{R}^D$ for node j in image B. Each element of the assignment matrix $\mathbf{S}^p$ is formed by:

$$\mathbf{S}_{ij}^p = (\mathbf{f}_i^A)^\top \mathbf{f}_j^B \ . \tag{5.5}$$

We add a dustbin row and column at the end of $\mathbf{S}^p$, filled with a learnable parameter representing the threshold below which a node is considered unmatched, as [Sarlin et al., 2020a] also does. We then apply softmax on all rows and all columns, and compute their geometric mean:

$$\mathbf{S}_{\text{final}}^p = \sqrt{\text{softmax}_{\text{row}}(\mathbf{S}^p) \odot \text{softmax}_{\text{col}}(\mathbf{S}^p)} \ . \tag{5.6}$$

Where $\odot$ means the element-wise product. Given this final assignation matrix, we keep the mutual nearest neighbors that have a matching score above a given threshold η.

On the other hand, lines are matched in a similar way, except that each line is represented by its two endpoints features $\mathbf{f}_s \in \mathbb{R}^D$ and $\mathbf{f}_e \in \mathbb{R}^D$. To make the matching agnostic of the endpoint ordering, we take the maximum of the two configurations in the line assignation matrix (see Figure 5.4):

$$\begin{aligned} \mathbf{S}_{ij}^l = \max\Big(&\left(\mathbf{f}_s^A\right)^\top \mathbf{f}_s^B + \left(\mathbf{f}_e^A\right)^\top \mathbf{f}_e^B, \\ &\left(\mathbf{f}_s^A\right)^\top \mathbf{f}_e^B + \left(\mathbf{f}_e^A\right)^\top \mathbf{f}_s^B \Big) \ . \end{aligned} \tag{5.7}$$

Finally, we get $\mathbf{S}_{\text{final}}^l$ by applying the dual-softmax of Eq. 5.6 and match lines with mutual nearest neighbors.

5.4.4 Ground Truth Generation

A challenging task in line matching is to generate high-quality labels handling line fragmentation, assignation, and partial visibility. To obtain the Ground Truth (GT) point matches $\mathcal{M}^p$, we use the same methodology as in [Sarlin et al., 2020a]. In a nutshell, we leverage camera poses and depth to re-project keypoints from one image to another, and we add a new match whenever a re-projection falls within a small neighborhood of an existing keypoint.

For lines, we also leverage depth, but with a more complex setup. Let images A and B contain M and N line segments indexed by $\mathcal{A} := \{1, \dots, M\}$ and $\mathcal{B} := \{1, \dots, N\}$. We will denote the generated GT line matches $\mathcal{M}^l = \{(i, j)\} \subset \mathcal{A} \times \mathcal{B}$. For each segment $\mathbf{l}_i^A$ detected

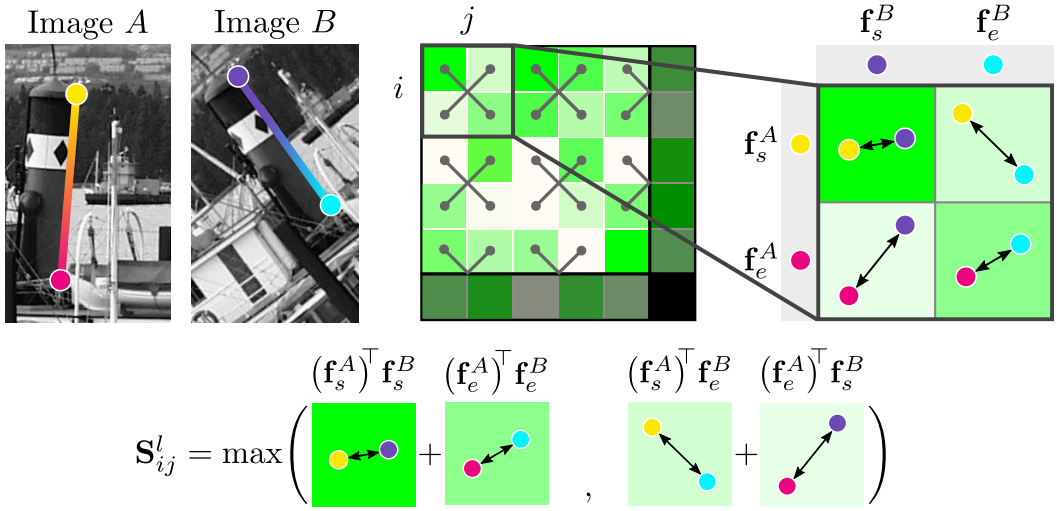


Figure 5.4: Line matching with order-agnostic endpoints. We consider the maximum score assignment between the two possible configurations of endpoint matching.

on image A, we sample K points $\left[\mathbf{x}_{i,1}^A, \ldots, \mathbf{x}_{i,K}^A\right]$ along it. A point is considered invalid if it has either no depth or its projection $\mathbf{x}_i^B$ in the other image has no depth. A point is also considered non-valid if it is occluded. We detect these cases by comparing the depth $d(\mathbf{X}_i)$ of the unprojection $\mathbf{X}_i$ in 3D of point $\mathbf{x}_i^A$ with its expected depth d^B in image B:

$$\text{Occluded} = \frac{|d(\mathbf{X}_i) - d^B|}{d^B} > T_{\text{occlusion}} \ , \tag{5.8}$$

where $T_{\text{occlusion}}$ defines the tolerance threshold of depth variations. Segments with more than 50% of invalid points are labeled as `IGNORE` and will not affect the loss function.

Next, we generate a closeness matrix $\mathbf{C}^B \in \mathbb{N}^{M \times N}$ keeping track of how many sampled points of line i in A are reprojected close to a line j in B:

$$\mathbf{C}_{i,j}^B = \sum_{k=1}^{K} \mathbb{1}\left(\text{valid}(\mathbf{x}_{i,k}^B) \wedge d_\perp\left(\mathbf{x}_{i,k}^B, \mathbf{l}_j^B\right) < T_{\text{dist}}\right) \ , \tag{5.9}$$

where $\mathbb{1}(\cdot)$ is the indicator function and $d_\perp(\cdot,\cdot)$ the perpendicular point-line distance. T_{dist} is a distance threshold in pixels that controls how demanding the GT is. $\mathbf{C}^A$ is defined analogously, and thus, we can define a cost matrix $\mathbf{C}$ with a minimum overlap threshold T_{overl}:

$$\mathbf{C}_{i,j} = \begin{cases} \infty, & \text{if } \mathbf{C}_{i,j}^A < T_{\text{overl}} \vee \mathbf{C}_{j,i}^B < T_{\text{overl}} \\ -\mathbf{C}_{i,j}^A \mathbf{C}_{j,i}^B, & \text{otherwise} \ . \end{cases} \tag{5.10}$$

Last, we solve the assignation problem defined by $\mathbf{C}$ with the Hungarian algorithm [Kuhn, 1955]. The resulting assignations $(i,j) \in \mathcal{M}^l$ are the `MATCHED` features, whereas all the valid entries $\mathcal{I} \subseteq \mathcal{A}$ and $\mathcal{J} \subseteq \mathcal{B}$ that were not assigned are labeled as `UNMATCHED`.

5.4.5 Loss Function

A classical approach for descriptor learning is to apply the triplet-ranking-loss [Balntas et al., 2016c, Suárez et al., 2021] with hard negative mining [Mishchuk et al., 2017]. However, repetitive structures are often present along lines, which may produce detrimental hard negatives. We resort instead to minimizing the negative log-likelihood of point and line assignments $\mathbf{S}^p_{\text{final}}$ and $\mathbf{S}^l_{\text{final}}$:

$$\mathcal{L} = \frac{\text{NLL}(\mathbf{S}^p_{\text{final}}, \mathcal{M}^p) + \text{NLL}(\mathbf{S}^l_{\text{final}}, \mathcal{M}^l)}{2} , \tag{5.11}$$

where for an assignment matrix $\mathbf{A}$ and GT matches $\mathcal{M}$:

$$\begin{aligned} \text{NLL}(\mathbf{A}, \mathcal{M}) = &- \sum_{(i,j)\in\mathcal{M}} \log \mathbf{A}_{i,j} \\ &- \sum_{i\in\mathcal{I}} \log \mathbf{A}_{i,N+1} - \sum_{j\in\mathcal{J}} \log \mathbf{A}_{M+1,j} . \end{aligned} \tag{5.12}$$

5.5 Experiments

We pre-train our model on pairs of images synthetically warped by a homography, using the one million distractor images of [Radenović et al., 2018], increasing the difficulty of the homographies gradually and speeding up convergence. We then fine-tune the model on MegaDepth [Li and Snavely, 2018] that contains 195 scenes of outdoor landmarks. We select image pairs with a minimum overlap of 10% of 3D points and resize each to 640×640 px. The wireframe threshold d to merge nodes is set to 3 pixels, and the parameters to generate the GT are: $T_{\text{occlusion}} = 0.1$, $T_{\text{dist}} = 5$, and $T_{\text{overl}} = 0.2$. Our GNN contains 9 blocks of [self-attention, line message passing, cross-attention], and the matching threshold is set to $\eta = 0.2$. Features inside the network have size $D = 256$. We optimize our network using Adam with learning rate 10^{-4} for the homography pre-training and 10^{-5} for MegaDepth. To limit computational cost during training, we set a maximum number of 1000 keypoints and 250 line segments per image. Training takes 10 days on 2 NVIDIA RTX2080 GPUs.

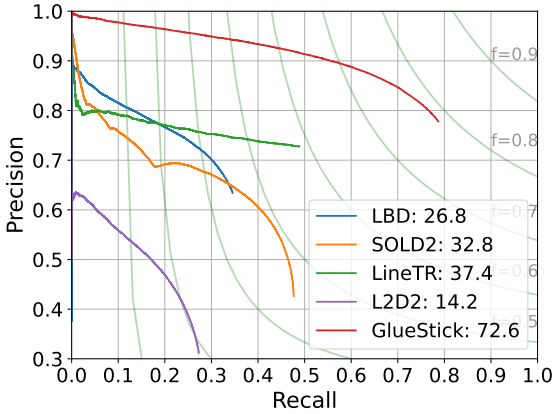
5.5.1 Baselines

In the following, we compare GlueStick with several state-of-the-art line matchers: the handcrafted LBD[1] [Zhang and Koch, 2013], the self-supervised SOLD2 [Pautrat et al., 2021], the transformer-based LineTR [Yoon and Kim, 2021], and the L2D2 [Abdellali et al., 2021] descriptors. SOLD2 uses its own detector since it is integrated with the descriptor. For all the other methods we use LSD [Von Gioi et al., 2008]. We also compare to the Point-Line Localization (PL-Loc) [Yoon and Kim, 2021], the point-line matcher combining SuperPoint [DeTone et al., 2018] and LineTR [Yoon and Kim, 2021]. Whenever possible, we also compare to two additional point-based matchers: ClusterGNN[2] [Shi et al., 2022] and the Local Feature Transformer (LoFTR) [Sun et al., 2021].

[1] We use the authors' code instead of the binary version from OpenCV.
[2] We reuse the numbers of the paper as the code is not publicly available.

	P	L	W	LMP	AP (↑)
SG + Endpts	✓	✓			54.5
SG + W	✓	✓	✓		67.6
SG + LMP	✓	✓		✓	69.9
GlueStick-L		✓	✓	✓	64.0
GlueStick	✓	✓	✓	✓	**72.6**

(a) Ablation study



(b) Comparison to the state of the art.

Figure 5.5: Ablation study and comparison to the State of the Art (SOTA) on the ETH3D dataset [Schöps et al., 2017]. We compute the line matching precision-recall curves and Average Precision (AP), displayed in the legend. (a) We compare several variations of our method using points (P), lines (L), wireframe connectivity (W), and Line Message Passing (LMP). (b) GlueStick surpasses all SOTA line matchers.

5.5.2 Ablation Study

Line segments are especially challenging to match in 3D due to occlusions, background changes, or partial visibility. We advocate for a proper evaluation of line matching covering these scenarios. Our ablation study is thus led on the ETH3D [Schöps et al., 2017] dataset, an indoor-outdoor dataset of multiple scenes with GT LiDAR depth, and poses. We use the 13 scenes of the training set of the high-resolution multi-view images (downsampled by a factor of 8), and sample all pairs of images with at least 500 GT keypoints in common, similarly as in [Pautrat et al., 2021]. We apply the same methodology as in Sec. 5.4.4 to compute the GT line matches. Given this GT, we can compute the precision-recall curve of the line matching by ordering lines by decreasing matching score.

We compare several variations of our method in Figure 5.5a. *SG + Endpts* refers to the pre-trained outdoor model of SuperGlue (SG) [Sarlin et al., 2020a] to match the line endpoints, and use our proposed line association of Sec. 5.4.3 agnostic to the ordering of endpoints. *SG + W* is similar, but with our proposed wireframe preprocessing connecting line segments together. *SG + LMP* represents a SuperGlue backbone with the addition of our Line Message Passing, but no wireframe preprocessing. Finally, *GlueStick-L* is our proposed model without keypoints and matching lines only. The AP shows that both the wireframe pre-processing and LMP bring a large boost of performance on the SuperGlue baseline. Their combination - our proposed model GlueStick - obtains the highest performance. *GlueStick-L* loses performance, but remains competitive, showing that the line matching is not relying only on points.

5.5.3 Line Matching Evaluation on ETH3D

We compare our method with previous state-of-the-art line matchers on the ETH3D dataset [Schöps et al., 2017], and show the blatant superiority of GlueStick in Figure 5.5b. It recovers almost 80% of the GT line matches, whereas the best previous methods do

not manage to reach 50% of recall. At equivalent recalls, it outperforms the previous best method, LineTR, by more than 15% in precision, and is almost doubling the AP. This major improvement is due to the possibility of leveraging points in the matching, and to the rich signal provided by the wireframe structure. Note that GlueStick without points (in Figure 5.5a), is still significantly better than other pure line matchers. It obtains good results thanks to the inclusion of a graph matching strategy combining appearance similarities and geometric consistencies. Despite having powerful descriptors, L2D2 and SOLD2 obtain worse results, because they neither use scene points nor geometric consistency between matches.

In terms of run time, GlueStick is also competitive. It runs in 258 ms on average on the images of ETH3D (around 775×515 pixels), which is similar to the timing of SuperGlue of 235 ms. Other line matchers are slower: 419 ms for SOLD2 and 304 ms for LineTR.

5.5.4 Homography Estimation

We evaluate in the following our method on the task of homography estimation. While HPatches [Balntas et al., 2017] is the most popular dataset, it is now very saturated [Sarlin et al., 2020a, Sun et al., 2021], and contains few structural lines that would be necessary to properly estimate a homography. Thus, lines do not help much to improve the current performance obtained by point methods. We nevertheless show in Section 5.5.4.1 that GlueStick ranks first among all considered methods on HPatches. To obtain more contrasted results, we also implement two meaningful experiments evaluating the homography estimation task in real-world scenarios: relative pose from planar surfaces (Sec. 5.5.4.2), and relative pose with pure rotations (Sec. 5.5.4.3).

5.5.4.1 Homography Estimation on HPatches

HPatches [Balntas et al., 2017] is one of the most frequently used datasets to evaluate image matching. It contains 108 sequences where the scene contains only one dominant plane, with 6 images per sequence. Each sequence has either illumination or viewpoint changes. Similarly as in [Sarlin et al., 2020a], we compute a homography from point and/or line correspondences and RANSAC, and compute the Area Under the Curve (AUC) of the reprojection error of the four image corners. We report the results for thresholds 3 / 5 / 10 pixels. We also compute the precision and recall of the Ground Truth (GT) matches obtained by the GT homography.

We report the results in Table 5.1. HPatches is clearly saturated and the precision/recall metrics are already very high for point-based methods. Note the strong performance of GlueStick on line matching, with an increase by nearly 10% in precision compared to the previous state of the art. Regarding point-based methods and homography scores, GlueStick obtains a very similar performance as the previous point matchers SuperGlue [Sarlin et al., 2020a] and LoFTR [Sun et al., 2021], and ranks first (with a very small margin) in terms of homography estimation. In addition to the fact that HPatches is saturated, it contains also very few structural lines that could have been useful to refine the homography fitting. Thus, the improvement brought by line segments is not significant here.

		AUC ↑			Points ↑		Lines ↑	
		3px	5px	10px	P	R	P	R
L	L2D2	43.73	55.98	69.39	-	-	55.55	38.76
	SOLD2	26.60	36.41	48.82	-	-	80.57	77.43
	LineTR	42.90	55.74	69.26	-	-	78.78	58.74
	LBD	46.82	59.13	71.82	-	-	82.73	56.38
	GlueStick-L	46.61	61.45	76.32	-	-	**90.27**	76.69
P	SuperGlue	66.21	77.77	88.05	**98.85**	97.44	-	-
	LoFTR	66.15	75.28	84.54	97.60	**99.38**	-	-
	GlueStick-P	65.88	77.41	87.72	**98.85**	97.08	-	-
P+L	PL-Loc	60.03	71.44	83.08	90.80	77.60	80.33	50.35
	GlueStick-PL	**66.88**	**78.14**	**88.12**	98.00	94.86	89.54	**80.44**

Table 5.1: Homography estimation on HPatches [Balntas et al., 2017]. We report the Area Under the Curve (AUC) of the cumulative error curve generated by the re-projection error of the four image corners at different thresholds (3px, 5px, 10px), as well as the precision (P) and recall (R) of the matches.

5.5.4.2 Dominant Plane on ScanNet

ScanNet [Dai et al., 2017] is a large-scale RGB-D indoor dataset with GT camera poses, which pictures some hard cases for feature points with low texture, and where lines are expected to provide better constraints. We use the same test set of 1500 images as in [Sarlin et al., 2020a], where the overlap between image pairs is computed from GT poses and depth. For each image pair, we match them with different state-of-the-art point, line, and point-line matchers. We then use a hybrid RANSAC [Sattler et al., 2019, Camposeco et al., 2018] to estimate a homography from these feature correspondences. This is a common way to initialize SLAM systems [Mur-Artal et al., 2015]. Since the GT homography is not known, we rely on the GT relative pose to evaluate the quality of the retrieved homography, as was done in previous works [Baráth et al., 2023]. The relative pose corresponding to the predicted homography can be extracted using [Malis and Vargas, 2007]. We report the pose error, computed as the maximum of the angular error in translation and rotation [Yi et al., 2018, Brachmann and Rother, 2019, Sarlin et al., 2020a], as well as the corresponding pose AUC at error thresholds 10 / 20 / 30 degrees error. Note that this evaluation is valid regardless of the plane selected by each method to estimate the homography: all planes lead to the same relative pose.

The results are shown in Tab. 5.2. It can be seen first that GlueStick matching points only obtains better results than SuperGlue. This shows that our re-trained network is able to match and even outperform SuperGlue network for keypoint matching. Secondly, when matching lines only, GlueStick significantly exceeds the previous state of the art for line matching. This demonstrates that leveraging context from neighboring lines and being aware of their interconnection is highly beneficial. Finally, we obtain the best results overall when combining points and lines. The network can leverage both kinds of features and may rely

		Pose error ↓	Pose AUC ↑
Points	SuperGlue (SG)	18.1	15.6 / 29.8 / 39.4
	LoFTR	16.8	15.8 / 30.9 / 41.4
	GlueStick	**15.7**	**17.4 / 32.8 / 42.9**
Lines	LBD	49.2	3.7 / 8.2 / 13.4
	SOLD2	55.6	4.9 / 10.8 / 16.1
	LineTR	51.6	4.5 / 11.0 / 16.8
	L2D2	60.0	2.8 / 6.5 / 10.5
	SG + Endpts (no KP)	36.0	7.1 / 15.0 / 22.2
	GlueStick	**27.6**	**9.4 / 20.0 / 28.6**
Points + Lines	PL-Loc	26.2	12.2 / 24.1 / 32.2
	SG + Endpts	17.1	17.5 / 31.8 / 41.2
	GlueStick	**14.1**	**19.3 / 35.4 / 46.0**

Table 5.2: Homography estimation on ScanNet [Dai et al., 2017]. We first estimate a homography based on point-only, line-only or points+lines matches, then decompose it into the corresponding relative pose. We report the median pose error in degrees, as well as the AUC at 10° / 20° / 30° error.

more on the accurate points on well-textured images, while using lines in scenarios with scarce points.

This experiment is meant to evaluate the quality of homographies retrieved from points, lines or points+lines features. However, there exists better methods to obtain a relative pose between images from points only. For the sake of completeness, we also report here the results that would be obtained with the 5-point algorithm to obtain the essential matrix [Nistér, 2003], later decomposed as a relative pose. Table 5.3 demonstrates that the quality of relative poses retrieved through essential matrices is much higher than with homographies - as could be expected. GlueStick remains nevertheless the top performing method among all baselines. Note that we use here the outdoor models for all methods, for fairness reasons as GlueStick was only trained on outdoor data.

		Pose error ↓	Pose AUC ↑
Pose from H	SuperGlue	18.1	15.6 / 29.8 / 39.4
	LoFTR	16.8	15.8 / 30.9 / 41.4
	GlueStick	**14.1**	**19.3 / 35.4 / 46.0**
Pose from E	SuperGlue	8.6	30.5 / 46.0 / 54.1
	LoFTR	11.7	23.6 / 39.6 / 48.4
	GlueStick	**8.4**	**30.9 / 46.8 / 55.1**

Table 5.3: Using essential matrices instead of homographies on ScanNet [Dai et al., 2017]. While our experiment on ScanNet is meant to evaluate homographies, we display here the results that would be obtained when using essential matrices to get a relative pose, instead of homographies. We report the median pose error in degrees, as well as the AUC at 10° / 20° / 30° error. Essential matrices are naturally more robust and obtain better results when evaluated on relative pose estimation.

5.5.4.3 Pure Rotations on SUN360

We also evaluate our method in estimating pure camera rotations, which are the cornerstone of some applications such as image stitching, or visual-guided sensor fusion. We use the SUN360 [Xiao et al., 2012] dataset containing 360º images. From each original 360º image, we crop 10 pairs of perspective images (640×480 pixels) with a field-of-view of 80º. Pairs are randomly sampled with an angular difference in range $\pm\,[50°, 70°]$ for yaw and $\pm\,[0°, 30°]$

for pitch. We first extract the local feature matches, then we estimate a rotation using a hybrid RANSAC [Sattler et al., 2019, Camposeco et al., 2018]. We evaluate the angular error between the predicted and GT relative rotations.

To be more precise, images are cropped from the panorama images of SUN360 [Xiao et al., 2012] and projected with a calibration matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$. The relation between both images is defined by:

$$\mathbf{x}^B = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}\mathbf{x}^A \ , \tag{5.13}$$

where $\mathbf{R} \in SO(3)$ is the rotation matrix between the cameras. Thus, we can calibrate the features detected on each image, multiplying them by $\mathbf{K}^{-1}$. This way, we only have to robustly estimate the 3 Degrees-of-Freedom (DoF) of the rotation.

Point features are sampled uniformly, and lines are sampled proportionally to the square root of their length to give priority to larger lines. The probability of choosing one type of feature is proportional to its number of matches. For example, if a pair has 60 point matches and 40 line matches, the probability of choosing a point is 60% and 40% for lines.

The minimal solver randomly chooses 2 feature matches (point-point, point-line, or line-line) and estimates the rotation based on them. This can be seen as aligning 2 sets of 3D vectors. Homogeneous points are already 3D vectors going from the camera center to the plane $Z = 1$. To get a vector from a line segment with homogeneous endpoints $\mathbf{x}_s$ and $\mathbf{x}_e$ we use its line-plane normal $\mathbf{n} = \mathbf{x}_s \times \mathbf{x}_e$. To make the method invariant to the order of the endpoints, we force the normals of the segments to have a positive dot product. This simple heuristic works as long as the sought rotation is less than 180°. Therefore, we can obtain a 3D vector from any of the types of features. For two (or more) correspondences, the optimal rotation can then be found with Singular Value Decomposition (SVD) using the Kabsch algorithm [Kabsch, 1976].

In Figure 5.6, GlueStick-PL (with points and lines) obtains the best results because lines help to match pairs where there is not enough texture. Specifically, long lines can be detected very precisely, thus contributing to an accurate estimation. Point-based methods (SG and GlueStick-P) obtain already 3 points less of AUC. PL-Loc [Yoon and Kim, 2021] is ranked fourth because it effectively uses points and lines, though independently and without spatial reasoning for point matches.

We also provide a more extensive table of results in Table 5.4, as well as visual examples of the point and line matches obtained by GlueStick, and the resulting image stitching output in Figure 5.7.

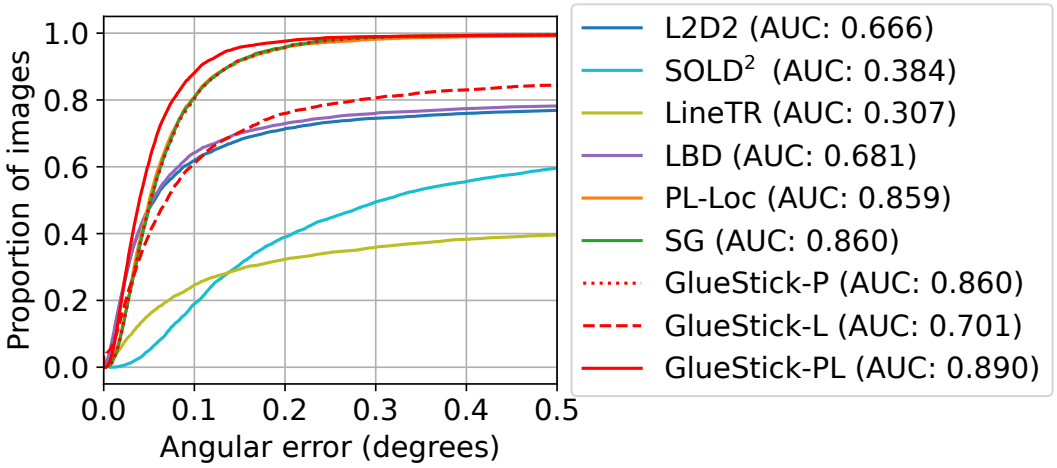


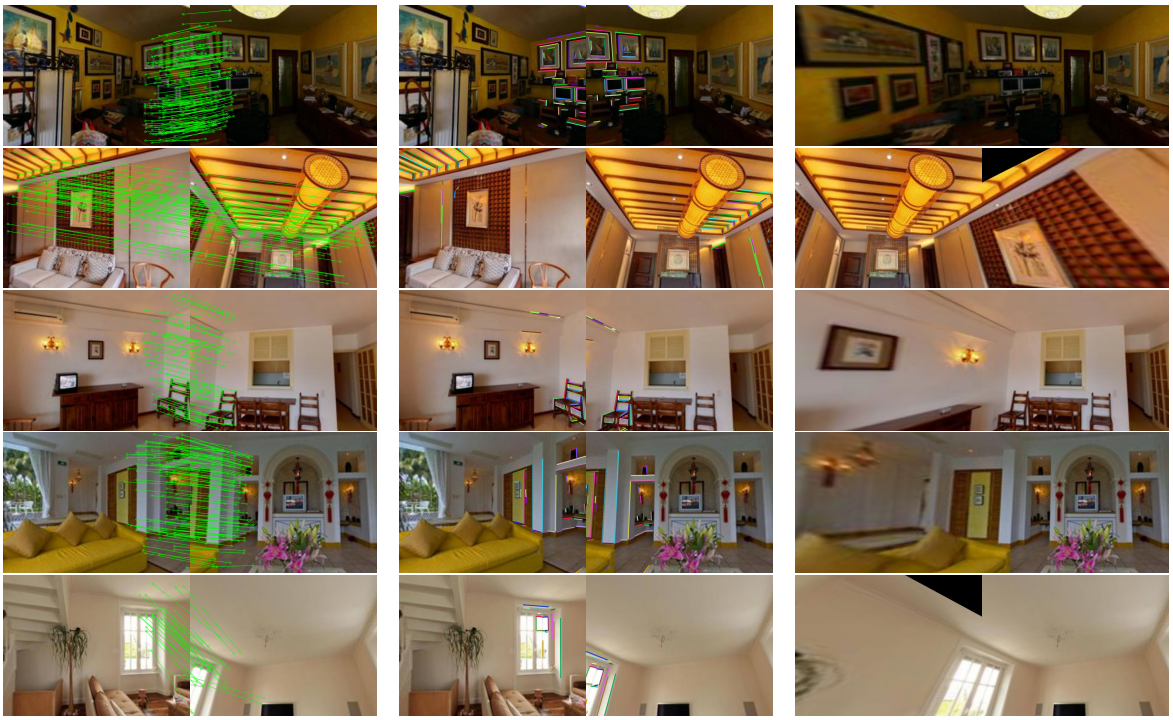
Figure 5.6: Camera rotation estimation in SUN360 [Xiao et al., 2012]. We show the cumulative angular error for all pairs of images. We report the AUC up to an error threshold of 0.5°.

	Rotation error ↓		AUC ↑					
	Avg	Med	0.25°	0.5°	1°	2°	5°	10°
LineTR	60.37	10.80	0.239	0.307	0.366	0.414	0.455	0.474
LBD	19.57	0.054	0.593	0.681	0.736	0.768	0.790	0.799
SOLD2	23.74	0.308	0.232	0.384	0.515	0.609	0.682	0.713
L2D2	18.31	0.056	0.578	0.667	0.725	0.765	0.795	0.808
GlueStick-L	8.79	0.070	0.579	0.701	0.780	0.830	0.869	0.885
SG	**0.135**	0.052	0.730	0.860	0.929	0.964	0.985	**0.992**
GlueStick-P	0.230	0.052	0.729	0.860	0.928	0.963	0.984	0.991
PL-Loc	0.338	0.050	0.733	0.859	0.927	0.961	0.982	0.989
GlueStick-PL	0.327	**0.039**	**0.789**	**0.890**	**0.943**	**0.970**	**0.986**	0.991

Table 5.4: Pure rotation estimation on SUN360 [Xiao et al., 2012]. We estimate a rotation based on point-only, line-only, or points+lines matches. We report the average and median rotation error in degrees, as well as the AUC at 0.25 / 0.5 / 1 / 2 / 5 / 10 degrees error.

5.5.5 Visual Localization

We introduce here the downstream task of localizing a query image, given the known poses of database images. We follow the *hloc* pipeline [Sarlin et al., 2019, Sarlin, 2020], and integrate line features and our own matcher in the existing code. We use NetVLAD [Arandjelović et al., 2016] for image retrieval, detect SuperPoint [DeTone et al., 2018] feature points and LSD [Von Gioi et al., 2008] lines, and match these features with either SuperGlue [Sarlin et al., 2020a] + a line matcher, or with GlueStick. We use the GT depth to back-project lines in 3D: points are sampled along each line, un-projected to 3D, and a 3D line is fit to these un-projected points. We use the solvers of [Kukelova et al., 2016, Zhou et al., 2018, Larsson, 2020] to generate poses from a minimal set of 3 features (3 points, 2 points and 1 line, 1 point and 2 lines, or 3 lines), then combine them in a hybrid RANSAC [Sattler et al., 2019, Camposeco et al., 2018] to recover the query camera poses.



(a) Point matches **(b)** Line matches **(c)** Image stitching results

Figure 5.7: Examples of GlueStick matches on image pairs of SUN360 [Xiao et al., 2012]. We provide the point and line matches, as well as the stitching of the two images using the resulting matches.

Datasets. We compare our method to other baselines on two datasets. The 7Scenes dataset [Shotton et al., 2013] is an RGB-D dataset displaying 7 indoor scenes with GT poses and depth. It is however limited in scale, and most scenes are already saturated for point-based localization. One scene remains extremely challenging for feature points: the Stairs scene, as illustrated in Figure 5.8. Due to the lack of texture and repeated steps of the stairs, current point-based methods are still struggling on this scene [Brachmann and Rother, 2022]. We report median translation and rotation error, as well as the percentage of successfully recovered poses under a 5 cm / 5° threshold. InLoc [Wijmans and Furukawa, 2017, Taira et al., 2018] is a large-scale indoor dataset with GT poses and depth, with two test scenes: DUC1 and DUC2. It is challenging for point-based methods due to images with low texture and large viewpoint changes. We report the pose AUC at 0.25m / 0.5m / 1m and 10°.

Results. The results can be found in Tab. 5.5. GlueStick-P is able to surpass SuperGlue, confirming the strong matching performance of isolated keypoints already. In particular, GlueStick obtains an improvement of 44% in pose accuracy over SuperGlue on Stairs. Adding line features significantly improves the performance for Stairs and brings a small improvement on InLoc as well. This demonstrates the importance of lines in texture-less areas and with repeated structures. Combining points and lines in a single network allows GlueStick to reason about neighboring features and can thus beat the other methods that match points and lines independently.

For completeness, we also provide the evaluation on the full 7Scenes dataset [Shotton

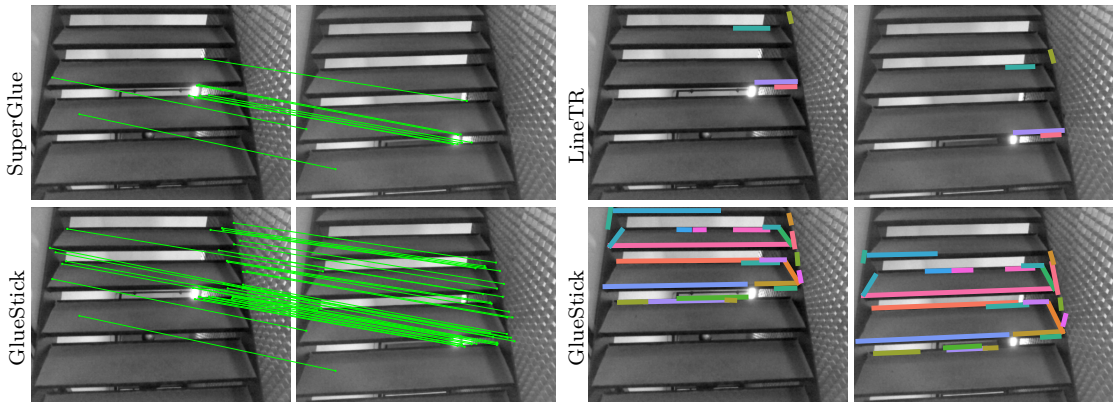


Figure 5.8: Correct matches on 7Scenes Stairs [Shotton et al., 2013]. Lines can guide the point matching in very challenging scenarios with low texture and repeated patterns.

		7Scenes Stairs		InLoc	
		T / R err.	Acc.	DUC 1	DUC 2
P	SuperGlue	4.7 / 1.25	53.4	48.5 / 68.2 / 80.3	53.4 / 75.6 / 82.4
	ClusterGNN	-	-	47.5 / 69.7 / 79.8	53.4 / 77.1 / 84.7
	LoFTR	**4.4** / **0.95**	53.9	47.5 / **72.2** / **84.8**	54.2 / 74.8 / 85.5
	GlueStick	**4.4** / 1.21	**55.4**	**49.0** / 70.2 / 84.3	**55.0** / **83.2** / **87.0**
P+L	SOLD2	3.2 / 0.83	75.8	44.9 / 69.7 / 79.8	54.2 / 75.6 / 80.2
	LineTR	3.7 / 1.02	66.6	46.0 / 67.2 / 76.3	53.4 / 77.1 / 80.9
	L2D2	4.1 / 1.15	55.8	46.5 / 68.7 / 80.3	51.9 / 75.6 / 79.4
	SG + Endpts	3.1 / 0.81	75.6	45.5 / 71.2 / 81.8	45.5 / 71.2 / 81.8
	GlueStick	**2.9** / **0.79**	**79.7**	**47.5** / **73.7** / **85.9**	**57.3** / **83.2** / **87.0**

Table 5.5: Visual localization on 7Scenes [Shotton et al., 2013] and InLoc [Taira et al., 2018]. We report the median translation (cm), rotation error (deg), and pose accuracy at a 5 cm / 5° threshold for the scene Stairs of 7Scenes, and the pose AUC at 0.25m / 0.5m / 1m and 10° error for InLoc. GlueStick ranks first both for points-only (P) and point-line (P+L) results.

et al., 2013] in Table 5.6. As mentioned earlier, the dataset is already largely saturated for point-based methods. Adding line segments into the pipeline can improve the results only in a few scenes such as Fire, Office and mostly on Stairs. While all methods obtain close results on such a saturated dataset, GlueStick is slightly ahead of the baselines, and largely outperforms them on the most challenging scene, Stairs.

5.5.6 Additional Insights

In this section, we give extra insights and motivate our design choices.

Generalization to other line detectors. In all our training and experiments, we used the Line Segment Detector (LSD) [Von Gioi et al., 2008] to extract line segments. For a certain application, such as indoor wireframe parsing [Huang et al., 2018], learned methods largely overtake classic ones [Dai et al., 2021, Xu et al., 2021b, Xue et al., 2020]. However, learned methods struggle to generalize this power to other contexts, tasks, or types of images. For this reason, we have chosen LSD as the generic method to train GlueStick. Furthermore, we believe that our line pre-processing, turning an unordered set of lines into a connected graph, is beneficial to make the endpoints more repeatable across views, thus potentially making LSD more repeatable.

However, it is important for GlueStick to generalize and perform well with other line

	Points			Points + Lines				
	SuperGlue	LoFTR	GlueStick - P	SOLD2	LineTR	L2D2	SG + Endpts	GlueStick - PL
Chess	**2.4** / 0.81 / 94.5	2.5 / 0.86 / 93.8	**2.4** / **0.80** / 94.3	**2.4** / 0.82 / 94.4	**2.4** / 0.81 / 94.5	**2.4** / 0.83 / 94.5	**2.4** / 0.82 / **94.6**	**2.4** / 0.82 / 94.5
Fire	1.9 / 0.76 / 96.4	1.7 / **0.66** / 96.8	2.0 / 0.78 / 96.6	**1.6** / 0.69 / 96.8	**1.6** / 0.69 / 97.0	**1.6** / 0.69 / 96.4	1.7 / 0.69 / 97.2	1.7 / 0.69 / **97.4**
Heads	1.1 / 0.74 / 99.0	1.1 / 0.78 / 98.2	1.1 / 0.74 / 99.2	**1.0** / **0.72** / **99.4**	1.1 / 0.75 / 99.1	**1.0** / 0.73 / 99.3	1.1 / 0.74 / 99.2	**1.0** / 0.74 / **99.4**
Office	2.7 / 0.83 / 83.9	2.7 / 0.83 / 82.0	2.7 / 0.83 / 83.6	**2.6** / 0.80 / **84.7**	**2.6** / **0.79** / 84.4	**2.6** / 0.81 / 83.9	**2.6** / **0.79** / 84.4	**2.6** / **0.79** / 84.6
Pumpkin	4.0 / 1.05 / 62.0	**3.9** / 1.12 / **62.4**	**3.9** / **1.04** / 62.2	4.0 / 1.07 / 60.2	4.0 / 1.08 / 61.5	4.0 / 1.05 / 61.3	4.0 / 1.06 / 61.2	4.0 / 1.06 / 61.5
Kitchen	3.3 / **1.12** / 72.5	3.3 / 1.14 / **73.8**	3.3 / **1.12** / 72.8	**3.2** / 1.15 / 72.6	3.3 / 1.15 / 72.6	**3.2** / 1.14 / 72.9	**3.2** / 1.14 / 72.8	**3.2** / 1.13 / 73.0
Stairs	4.7 / 1.25 / 53.4	4.4 / 0.95 / 53.9	4.4 / 1.21 / 55.4	3.2 / 0.83 / 75.8	3.7 / 1.02 / 66.6	4.1 / 1.15 / 55.8	3.1 / 0.81 / 75.6	**2.9** / **0.79** / **79.7**
Total	2.9 / 0.94 / 80.2	2.8 / 0.91 / 80.1	2.8 / 0.93 / 80.6	2.6 / 0.87 / 83.4	2.7 / 0.90 / 82.2	2.7 / 0.91 / 80.6	2.6 / **0.86** / 83.6	**2.5 / 0.86 / 84.3**

Table 5.6: Visual localization on the full 7Scenes dataset [Shotton et al., 2013]. We report the median translation error (cm) / median rotation error (deg) / pose AUC at a 5 cm / 5 deg threshold. Most scenes are already saturated for point methods, and lines can hardly make a difference.

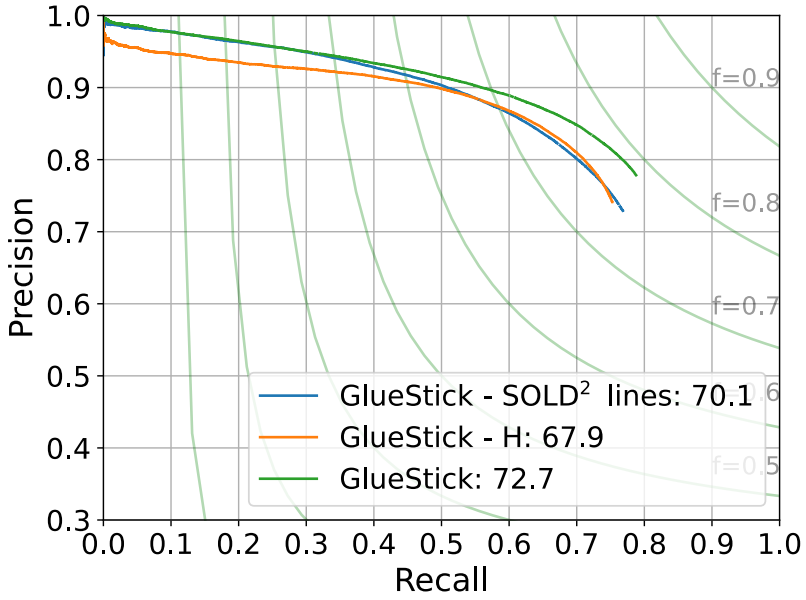


Figure 5.9: Additional ablation study on the ETH3D dataset [Schöps et al., 2017]. We report the precision-recall curve of the line matching, as well as Average Precision (AP) in the legend. Our final *GlueStick* model running with LSD [Von Gioi et al., 2008] lines is compared to its pre-trained version on homographies (*GlueStick - H*), and the final model using SOLD2 lines [Pautrat et al., 2021] (*GlueStick - SOLD2 lines*).

segment detectors. In Figure 5.9 we run GlueStick using either LSD or SOLD2 [Pautrat et al., 2021] lines, and we evaluate the precision-recall of both methods on the ETH3D dataset [Schöps et al., 2017]. It can be seen from the precision-recall curves that 1) our GlueStick model trained on LSD lines is able to generalize to other lines such as SOLD2 [Pautrat et al., 2021], and 2) the performance is slightly better with LSD lines. This is reasonable, since GlueStick was already trained on these lines.

Effect of the fine-tuning. Again in Figure 5.9, we compare our final GlueStick model with its pre-trained version on homographies, *GlueStick - H*. The plot shows that pre-training on homographies is already sufficient to get very high performance on ETH3D - better than the previous state-of-the-art line matchers. Fine-tuning on MegaDepth [Li and Snavely, 2018] with real viewpoint changes can however further improve the robustness of our matcher, as demonstrated by the stronger performance of the final model.

Dependence on point matches. When jointly matching two kinds of features, one caveat is often that one type of feature takes the lead and the other relies mainly on the first one. While we know from SuperGlue [Sarlin et al., 2020a] that point-only matching is already very strong on its own, we show here that our architecture is very robust to the absence of keypoints and that line-only matching is still possible. We ran an evaluation of the precision, recall, and AP of the line matching on 1000 validation images of our homography dataset (images taken from the 1M distractor images of [Radenović et al., 2018]), and tested different maximum numbers of keypoints per image. The results showed in Figure 5.10, highlight that our line matching is extremely robust to the lack of keypoints. The precision remains indeed constant, and the recall and AP are decreasing by at most 5% when switching from 1000 keypoints to no keypoints. Thus, this study confirms that our matcher can be used in

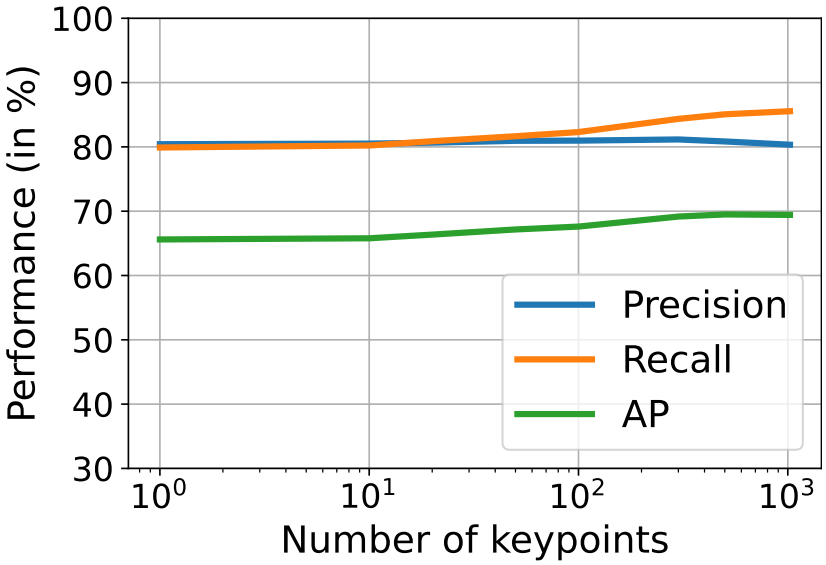


Figure 5.10: Analysis of the dependence on keypoints. We run GlueStick on 1000 image pairs warped by a homography (taken from the 1M distractor images of [Radenović et al., 2018]), with varying numbers of keypoints, and report the precision, recall and AP of the line matching. GlueStick robustly matches lines even when few or no keypoints are present.

texture-less areas where no keypoints are present, and is still able to match lines with high accuracy.

Impact of the line length. While we adopted a line representation based on the endpoints, one may wonder whether GlueStick can handle very long lines, and how it performs with respect to the line length. We studied this on the ETH3D [Schöps et al., 2017] by categorising lines into three categories of length (in pixels): *Short* ($[0, 50)$), *Medium* ($[50, 150)$), and *Long* ($[150, +\infty)$). The results are shown in Figure 5.11. It can be seen that the best performance is obtained for long lines, showing that GlueStick is still able to match lines even without context in the middle of the line. This result is due to the fact that long lines are more stable across views, while short ones are often noisy and not very repeatable.

Robustness to small image overlaps. The image overlap and scale changes between images can play a large role in matching. To study the effect of image overlap on GlueStick, we revisited our line matching experiment on the ETH3D dataset [Schöps et al., 2017] and separated the pairs of images into three categories of image overlap: *Small* ($[0, 0.33)$), *Medium* ($[0.33, 0.66)$), and *Large* ($[0.66, 1]$). Overlap is defined as the proportion of pixels falling into the other image after reprojection. It is computed symmetrically between the two images, and the minimum of the two values is kept. Results are available in Figure 5.12. While the performance naturally decreases with smaller overlaps, GlueStick maintains a strong performance on such hard cases.

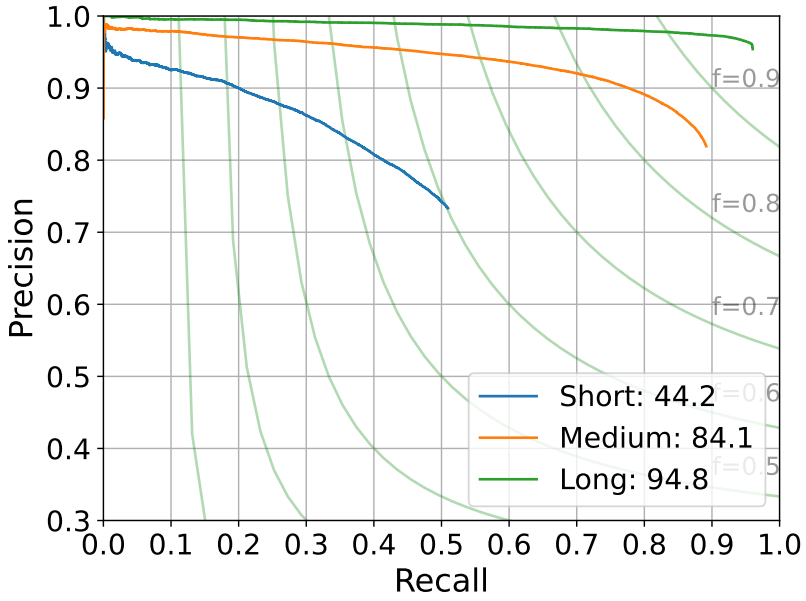


Figure 5.11: Analysis of the impact of line length. We run GlueStick on the ETH3D dataset [Schöps et al., 2017] and evaluate separately the matching of Short, Medium, and Long lines. The best performance is obtained for long lines, as they are more stable than short ones. GlueStick can thus match long lines even with an endpoint representation.

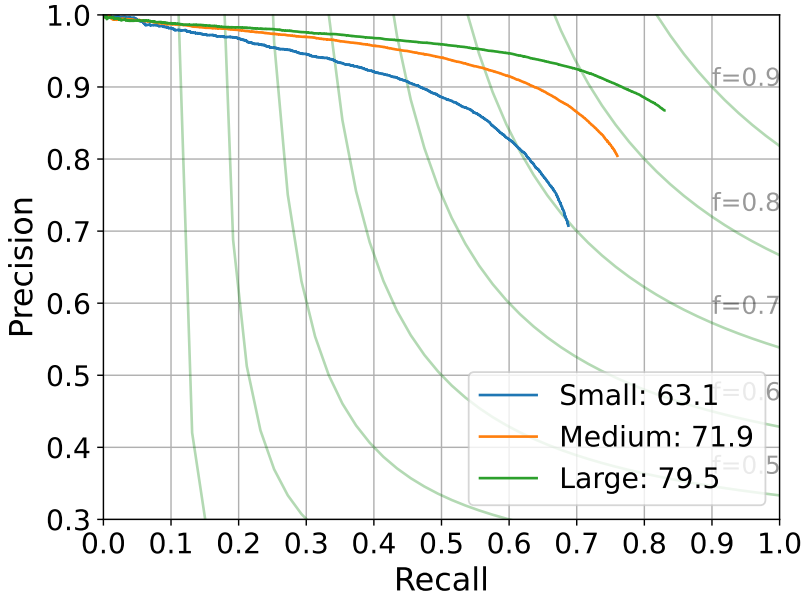


Figure 5.12: Analysis of the impact of the image overlap. We run GlueStick on the ETH3D dataset [Schöps et al., 2017] and classify image pairs into three categories: Small, Medium, and Large overlap. While the performance of GlueStick decreases with smaller overlaps, it is able to maintain a high performance for all kinds of overlaps.

5.5.7 Qualitative Examples

5.5.7.1 Feature Matches

Figure 5.13 displays some examples of line matching on the ETH3D dataset [Schöps et al., 2017]. We plot in green the correct matches and in red the incorrect ones. Thanks to its spatial reasoning in the GNN and context-awareness, GlueStick is consistently matching more lines and with a higher precision than previous works. This is in particular true for scenes with repeated structures, such as the one in the right column, where the descriptors of SOLD2[Pautrat et al., 2021] and L2D2[Abdellali et al., 2021] do not have context from neighboring lines, and can only match a few lines.

5.5.7.2 Visualization of the Camera Pose Estimation

We visualize the reprojection of points and lines on the scene Stairs of the 7Scenes dataset [Shotton et al., 2013] in Figure 5.14. We plot in green the points and lines that were originally detected in 2D, and re-project in red the corresponding 3D features using the estimated camera pose. The reprojections of GlueStick are almost perfectly aligned compared to the ones of hloc [Sarlin et al., 2019, Sarlin, 2020], highlighting the quality of the poses retrieved by our method.

5.5.7.3 Failure Cases and Limitations

While jointly matching keypoints and lines in the same matching network helps disambiguating many challenging scenarios, GlueStick may still underperform in some scenarios. We list in the following some limitations of our method, and report some failure cases.

Limitations. Currently, the main performance bottleneck of GlueStick lies in the line segment detection. Even though combining our DeepLSD [Pautrat et al., 2023a] with GlueStick can provide the best performance in point-line applications [Liu et al., 2023], the issues of lines mentioned in previous chapters are still valid. Partially occluded lines are for example a potential issue for GlueStick, as it represents lines with their two endpoints. However, we observed a surprisingly good robustness of GlueStick to partially occluded lines, probably thanks to the neighboring points and lines that are not occluded. Note that it is also possible to equip GlueStick with a similar mechanism as in SOLD2 [Pautrat et al., 2021], by sampling several points along the line segments, and matching them with the Needleman-Wunsch algorithm. We tried this option and observed a small increase in performance (notably in areas with occluded lines), but at the cost of higher running time. Therefore, we did not incorporate this feature in our final method.

Another issue is that points and lines are still detected with different methods for now. Thus, three networks / algorithms need to be run to detect and describe keypoints, detect lines, and finally match them. Jointly detecting and describing points and lines would be an interesting future direction of research. Furthermore, the extraction of discrete features such as points and lines is usually non-differentiable, such that one cannot get a fully differentiable pipeline going from the feature extraction to their matching. Enabling such end-to-end

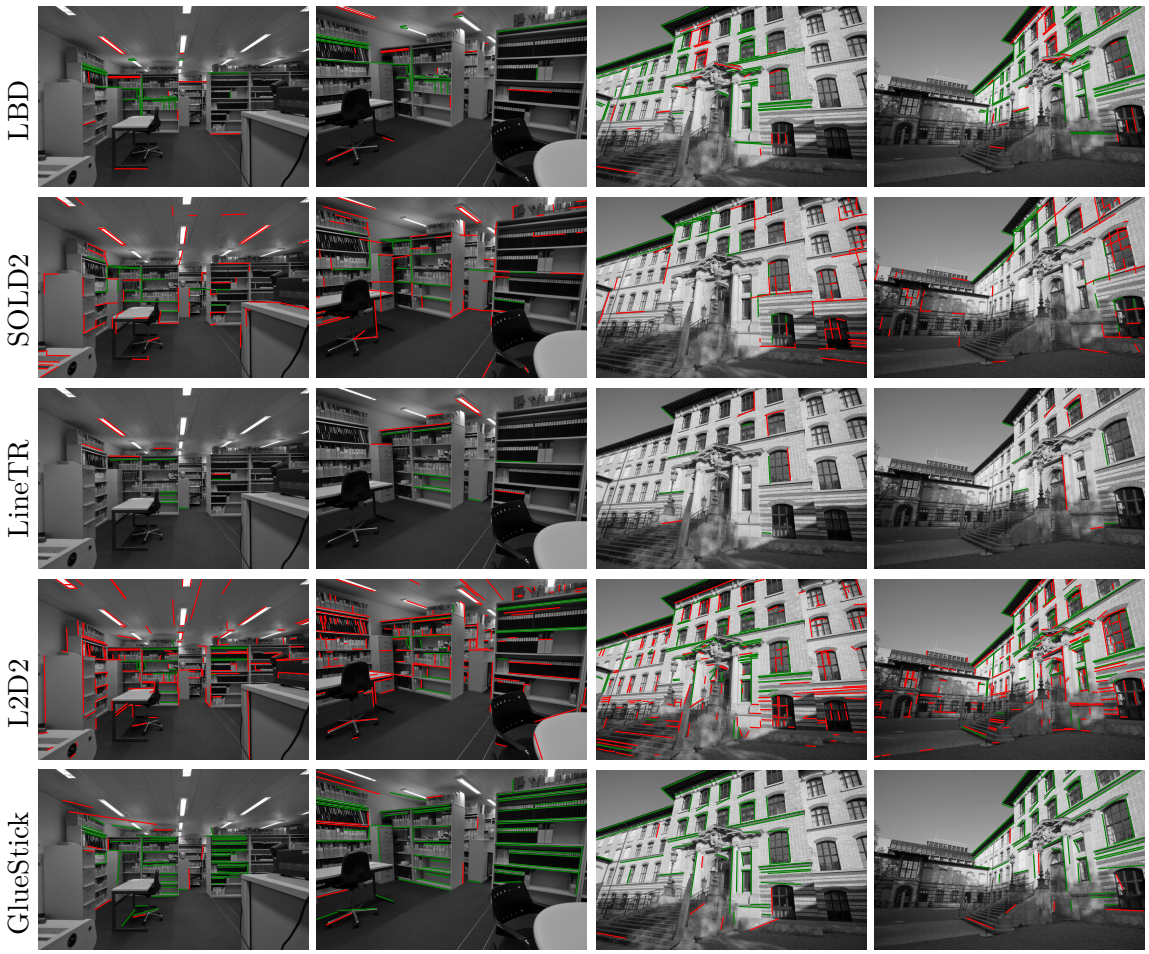


Figure 5.13: Line matches examples on ETH3D [Schöps et al., 2017]. We display correct line matches in green and incorrect ones in red for several state-of-the-art line matchers.

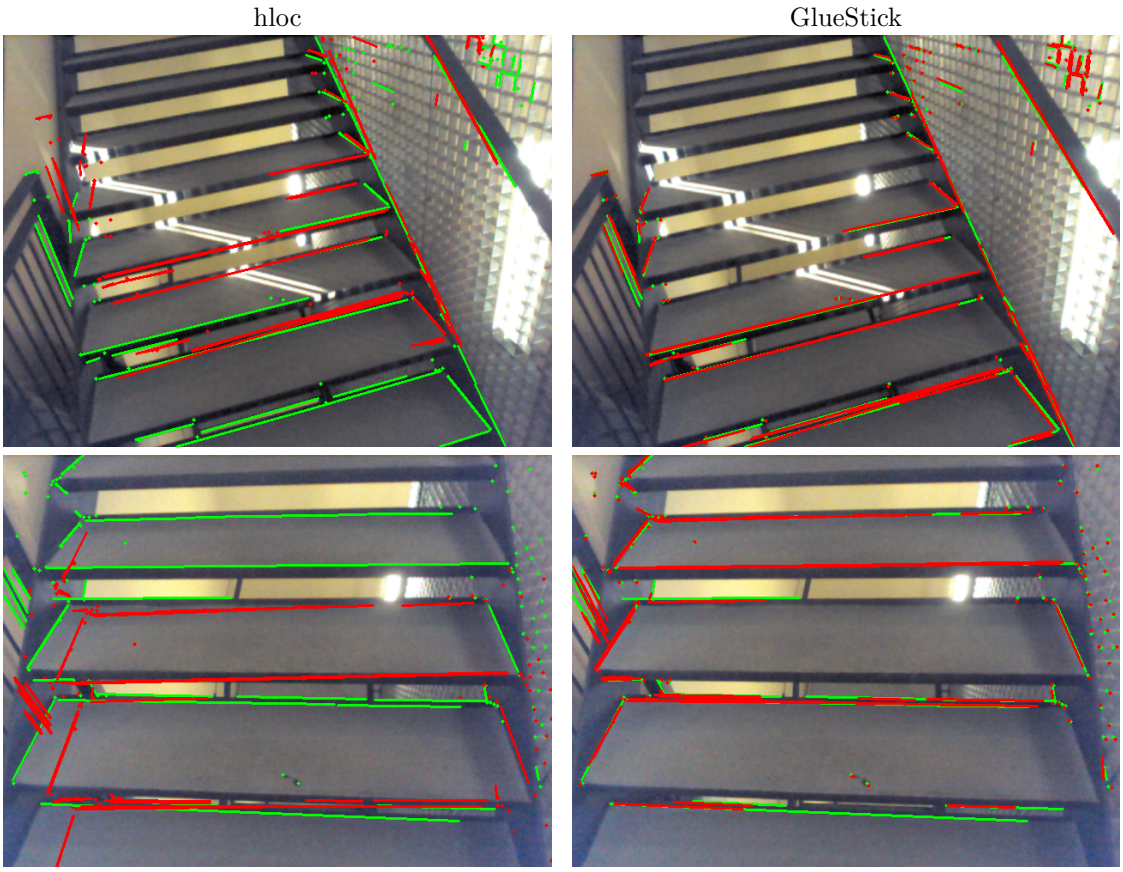


Figure 5.14: Camera pose estimation visualizations. We compare the originally detected keypoints and lines (in green) with the re-projected points and lines using the estimated camera pose (in red), for hloc [Sarlin et al., 2019, Sarlin, 2020] with SuperPoint [DeTone et al., 2018] + SuperGlue [Sarlin et al., 2020a] and our method. The reprojections of GlueStick align almost perfectly with the 2D detections, showing a high quality estimated pose.

training could potentially make features better specialized for matching.

Finally, our current supervision requires ground truth correspondences of points and lines across images (usually obtained through reprojection with depth and camera poses). Other supervision signals such as epipolar constraints and using two-view geometry would be an interesting direction of improvement in the future.

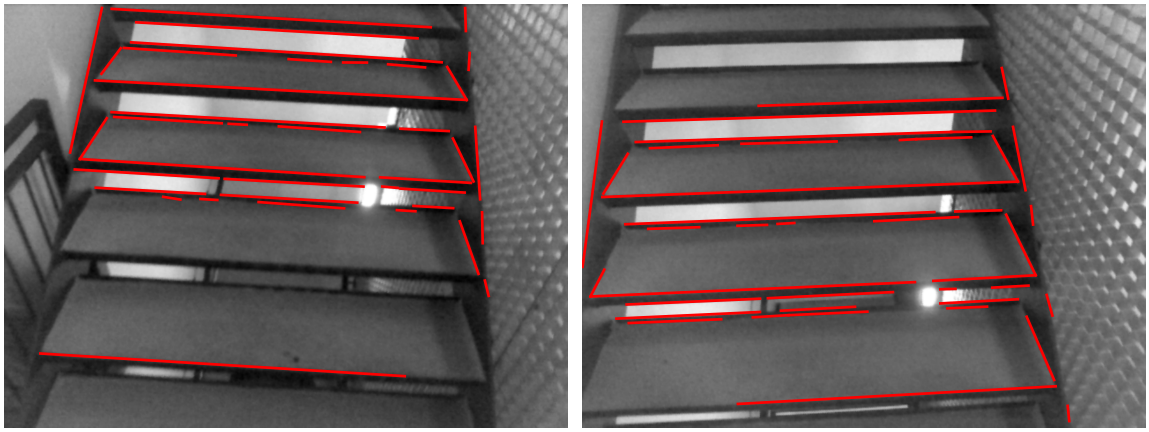
Failure cases. We display in Figure 5.15 a few examples of scenarios where GlueStick may still fail or underperform. First, in scenes with repeated patterns, GlueStick is able to find a consistent matching, but can be displaced by one pattern if there is no additional hint to disambiguate the transform between the two images. This is for example the case on 7Scenes Stairs [Shotton et al., 2013], when the camera is only seeing several steps, and it is unclear which step should be matching with which one in the other image.

Secondly, GlueStick has not been trained for large rotations beyond 45° and often fails in these scenarios. The pre-training with homographies was done with rotations lower than 45°, and real viewpoint changes are rarely with such rotations. Nonetheless, a simple fix is to rotate one of the two images by $0°, 90°, 180°$ and 270°, match it with the other image, and keep the best matching among the four.

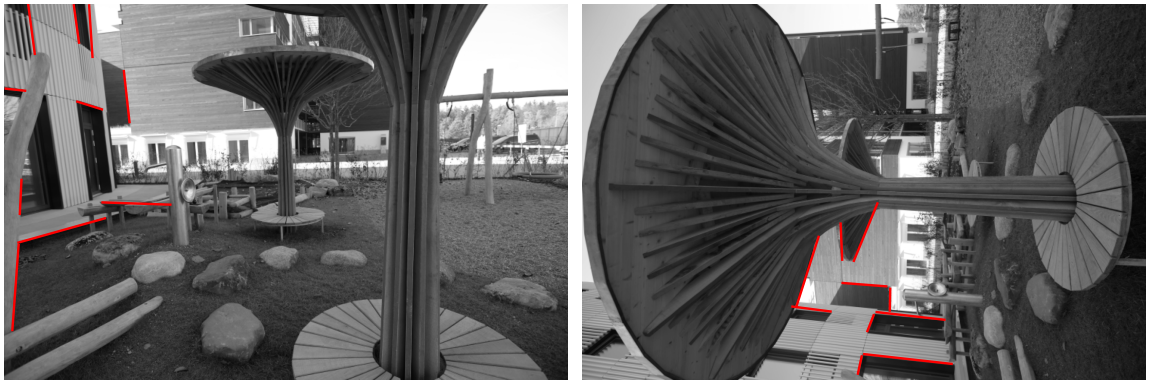
Finally, a challenging scenario happens when the images have low texture in combination with symmetric structures. The former makes visual descriptors less reliable, while the latter makes it harder to disambiguate matches from the spatial context. The performance is then degraded in such situations. Having access to sequential data and feature tracking may help solving such cases.

5.5.7.4 Attention visualization

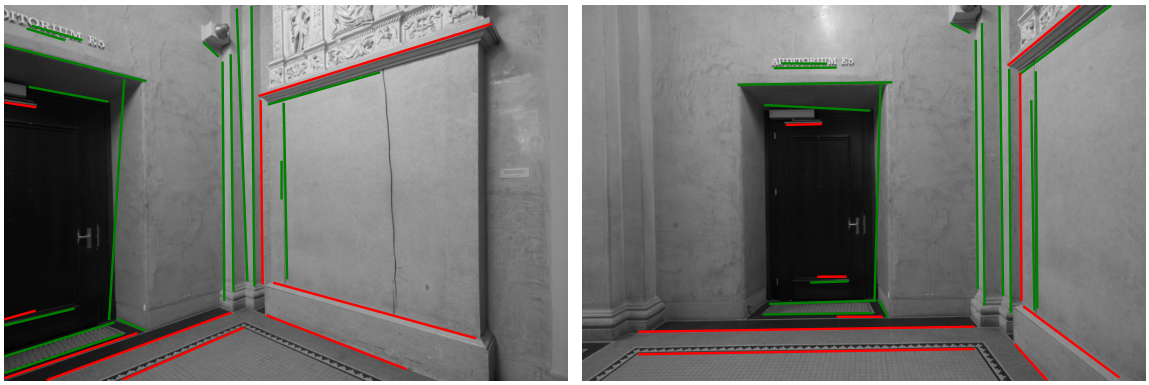
We display the attention for some nodes in Figure 5.16. This visualization is obtained by taking the attention matrix at various cross layers, averaging it across all heads, and taking the top 20 activated nodes. Green lines are used for nodes with connectivity greater than 0 (i.e. line endpoints), and cyan for nodes that are isolated keypoints. It can be seen from the left column that keypoint attention is leveraging the line structure to look for the right points along the line. In the right column, we can see that line endpoints can benefit from both keypoint and line endpoint attention. The attention is initially looking broadly at the image, before gradually focusing on the corresponding node in the other image. Thus, both points and line endpoints can complement each other to disambiguate the matching process.



(a) Repeated patterns: GlueStick finds consistent line matches, but displaced by one step.



(b) GlueStick is not trained on large rotations.



(c) The lack of texture / symmetric structures make visual descriptors / spatial descriptors less reliable.

Figure 5.15: Failure cases. We display correct line matches in green and wrong ones in red. GlueStick may still fail or underperform in some situations, such as (a) perfectly repeated patterns that are hard to disambiguate, (b) large rotation (e.g. $> 45°$), and (c) lack of texture and symmetric structures.

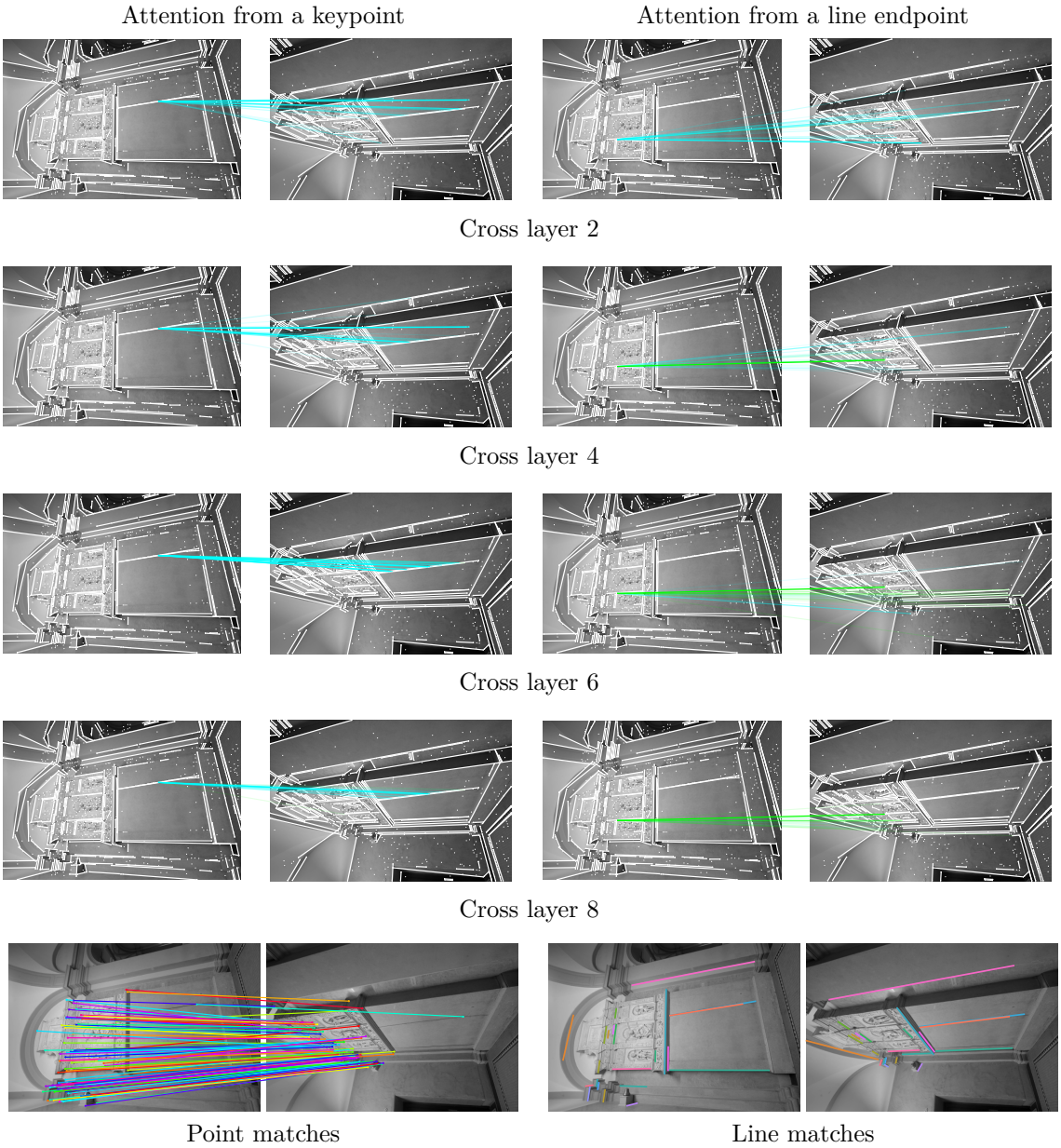


Figure 5.16: Cross attention visualization. We plot in the first column the attention from a keypoint and in the right column the attention of a line endpoint, for various layers in the Graph Neural Network. We compute here the average attention across all heads and keep the top 20 activated nodes. More opaque lines means higher attention, green matches are connected to a line endpoint in the second image, and cyan matches are connected to an isolated keypoint. The last row pictures the final matches.

5.6 Discussion

Summary. Matching points across two views and matching line segments are traditionally treated as two separate independent problems. In this chapter, we challenge this paradigm and present GlueStick, a learned matcher that jointly establishes correspondences between points and lines. By processing both types of features together, the network is able to propagate strong matches of either type to neighboring features that might have less discriminative appearance.

In our experiments, we show an improved matching performance across the board, for both points and lines. In particular for line matching, GlueStick provides a significant leap forward compared to the current descriptor-based state of the art. The key insight of this chapter is that line segments do not appear randomly scattered in the image, but rather form connected structures. This connectivity is explicitly encoded and exploited in our network architecture. Finally, we show that the improved matches we obtain directly translate to better results in downstream tasks such as homography and camera pose estimation.

Future works. As described in Section 5.5.7.3, the performance of GlueStick is still hindered by the inherent challenges coming with line segments. We believe nevertheless that the combination of DeepLSD and GlueStick is currently the best approach to mitigate these issues and to leverage line segments in practical applications. The next promising step would be to enrich GlueStick with additional constraints, such as epipolar constraints, and enforcing more consistency between the point and line matches. Finally, this chapter highlights the added value of matching point and line features in the same network. The next logical step to make the whole point-line pipeline more efficient, would be to detect and describe points and lines simultaneously as well.

Part IV

Conclusions

Chapter 6
Conclusions

6.1 Summary

In the course of this thesis, we explored some of the challenges of modern local features and offered solutions to handle them. Starting from point features, we moved on with line features and saw how they could benefit from similar techniques as points, but also require special processing to overcome their specific limitations. Finally, we brought both types of features into a single pipeline and showed how their combination could outperform their independent use. We summarize in the following the main conclusions of this work:

- Chapter 2 drew up the fact that feature descriptors suffer from an inherent trade-off between their discriminative power and their robustness to image changes. We thus advocate that local descriptors should not always seek the highest level of invariance to viewpoint and appearance changes, but should instead select in advance the right amount of invariance, based on the task at hand. Therefore, we proposed LISRD, a method to choose among several descriptors which one has the most adapted invariance to each situation in an online fashion. Our proposed solution offers a soft selection of the best descriptor, is suitable for both handcrafted and learned descriptors, and benefits from a coarse-to-fine mechanism through the use of meta descriptors.

- In Chapter 3, we transferred some of the successes of point features to the field of line features. We thus proposed to jointly learn the tasks of line segment detection and description in a single network, and demonstrated how to train such a network in a self-supervised way and to get rid of the bias of previous learned methods towards wireframe lines. We also put an emphasis on the repeatability of lines across different views, and improved the robustness of line descriptors to appearance changes and partial occlusion.

- In spite of the advances made by the previous chapter, we established in Chapter 4 that learned line detectors were still less accurate than their handcrafted counterparts. We thus proposed DeepLSD, a hybrid approach to fuse the robustness of deep learning with the accuracy of traditional methods. While deep learning is well-suited to pre-process the images to extract relevant details and to remove noise, handcrafted heuristics and classical optimization are still valid to extract the exact location of line segments.

Thanks to these new generic, robust, and highly accurate lines, we showed that line features could be employed in a wide range of scenarios.

- Finally, we came to the conclusion that points and lines are complementary features, and that they perform best when used together. We thus designed GlueStick, a graph neural network to jointly match points and lines across images. Not only is GlueStick surpassing the previous state of the art in terms of performance, but it is also more efficient compared to matching the features independently and avoids previous heuristic geometric strategies based on point-line structures. The key insight was that lines are usually connected to each other and form a wireframe, which is an essential pattern that was overlooked in previous approaches.

In summary, we showed that, even with the advent of deep learning, modern local features still suffer from some limitations. The perfect generic method working for all tasks and conditions is still beyond our reach, due to several trade-offs, to which we strived to bring solutions. First, local descriptors must balance their invariance and discriminability, and we proposed LISRD to ensemble multiple descriptors and overcome this. Second, while deep detectors are outperforming handcrafted ones in terms of robustness to appearance changes, they are still sometimes lagging behind the latter in terms of accuracy. We thus introduced DeepLSD as a hybrid method keeping the best of both worlds. Lastly, different features are relevant to specific environments: lines shine in indoor scenarios where points are scarce because of lack of texture, while points predominate in natural environments. Our solution is to leverage both features in parallel and to let them complement each other, as demonstrated with GlueStick.

Overall, points and lines share several similarities, as they can be defined with high intensity image gradients, and they share common description mechanisms. Thus, extracting and processing them simultaneously does not only improve the performance of downstream tasks, but it is also more efficient. With this thesis, we would like to encourage researchers to explore in the future additional ways to combine and to create synergies between point and line features.

6.2 Future Work

While this thesis brings solutions to several issues in local features, there remains promising areas of improvement. As highlighted in Chapter 5 and in the previous section, matching points and lines together simplifies the pipeline requiring both features. But since these features also share a lot of similarities, it would also make sense to extract them simultaneously. Currently, point-line applications are typically using a different method for each of the detection, description, and matching tasks of each features, resulting in up to 6 algorithms. Unifying everything in a single network would significantly simplify and speed up the process. Some tasks can in particular be shared to make it even more efficient. One could imagine detecting keypoints, connecting some of them to form line segments, predicting a single joint dense descriptor, and matching all the features with GlueStick.

Some of the methods proposed in this thesis are also not end-to-end, and may thus be sub-optimal. This is the case in DeepLSD, where the extraction of segments with LSD and the final refinement are not differentiable. Similarly for GlueStick, the input features and their descriptors are trained separately from the GNN, while training the whole pipeline together could potentially yield descriptors that are directly compatible with the GNN. Overall, more end-to-end frameworks would not only make the predictions more optimal, but they would also allow directly fine-tuning the models to specific tasks.

Finally, we chose to study points and lines in this thesis, as they are the simplest structures available. But one could also explore higher-level shapes, such as planes, or discriminative local surfaces. Moving to higher dimensions has multiple advantages, as it makes the detection task closer to semantic segmentation and may better generalize across scenes, it is more scalable, and it can simplify 3D reconstruction pipelines by directly producing meshes. While these mid-level features may be less accurate than local ones, one may consider using both mid-level and local ones in a coarse-to-fine fashion, or leveraging recent works such as scene coordinates that can obtain very accurate localizations with dense surfaces.

6.3 Outlook

Overall, this thesis is only a small leap forward for the world of local features, but we hope that the open-source tools developed during this work will prove to be useful in real-life scenarios. Whether it is an autonomous car leveraging LISRD to robustly adapt to adverse weather changes, an app building on top of SOLD2 and DeepLSD to provide automated reconstructions of your flat, or the highly accurate localization brought by GlueStick to provide the most comfortable AR experience, we encourage the community to adopt these tools to develop innovative solutions.

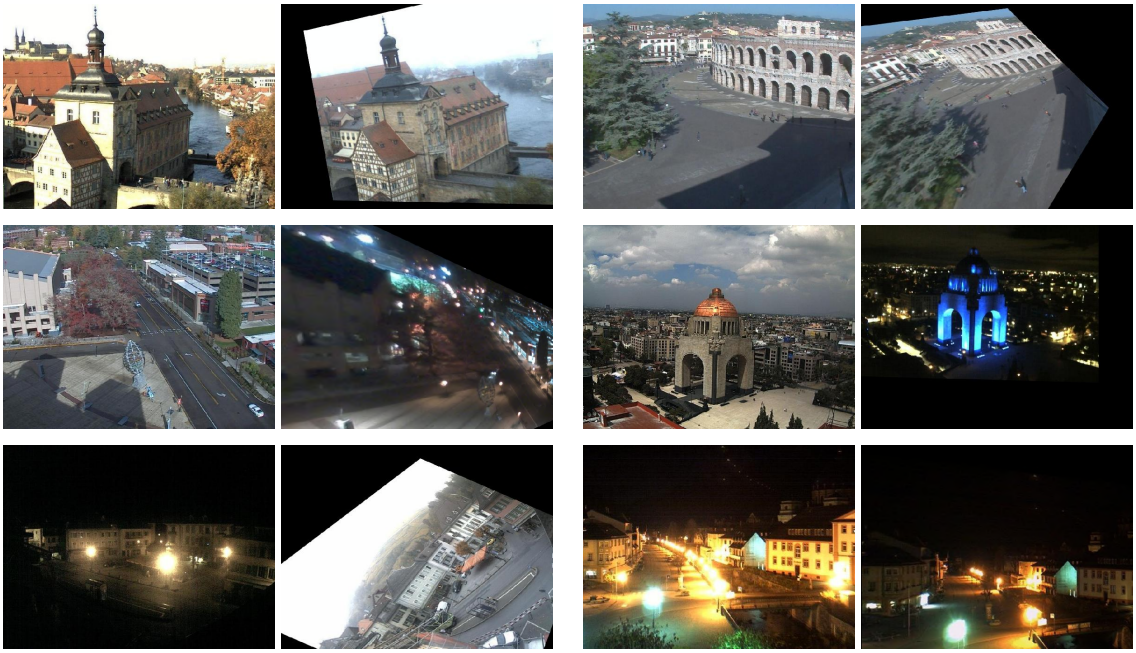
Appendices

Appendices A

The Rotated Day-Night Image Matching Dataset

This appendix provides additional details about our proposed Rotated Day-Night Image Matching (RDNIM) dataset. It offers a benchmark for local feature matching methods in the presence of illumination and rotation changes.

The Day-Night Image Matching (DNIM) dataset [Zhou et al., 2016] was originally released to evaluate the impact of day-night changes on local features matching. It consists of 1722 images grouped in 17 sequences of a fixed webcam taking pictures at regular time spans over 48h. In order to obtain pairs of images to match, a day and a night references are chosen for each sequence: the image with timestamp closest to noon is selected as day reference and similarly for the timestamp closest to midnight for the night reference. We then pair all the images in a sequence both with the corresponding day and night references, thus resulting in two benchmark datasets of 1722 pairs of images each. One dataset matches day references to all the DNIM images and is composed of day-day and day-night pairs, while the other dataset matches the night references to the DNIM images and displays night-night and night-day pairs. To simultaneously evaluate the robustness of our method to rotation and its discriminative power for non rotated images, we also warp the second image of each pair (i.e. the non reference image of the pair) with homographies. Similarly as in [DeTone et al., 2018], these homographies are generated by combining random translations, rotations, scalings, and perspective distortions, with an equal distribution of rotated and non rotated images. Thus, we call this augmented dataset RDNIM, for *Rotated* DNIM. Examples of the RDNIM image pairs are available in Figure A.1. The images and homographies used in this benchmark are accessible at `https://www.polybox.ethz.ch/index.php/s/P89YkZyOfdhmdPN/download` to let other researchers compare with their own methods.



Supplementary Figure. A.1: Sample images of the DNIM [Zhou et al., 2016] dataset augmented with rotations. All combinations among day-day, day-night and night-night pairs are available. Homographies are generated with random translations, rotations, scalings and perspective distortions, and images with and without rotation are equally distributed. When matching the images, the black artifacts created by the homography warping are masked out and ignored.

Bibliography

[Abdellali et al., 2021] Abdellali, H., Frohlich, R., Vilagos, V., and Kato, Z. (2021). L2D2: Learnable line detector and descriptor. In *International Conference on 3D Vision (3DV)*. 8, 63, 99, 112

[Agarap, 2018] Agarap, A. F. (2018). Deep learning using rectified linear units (ReLU). In *arXiv*. 68

[Agarwal and Mierle, 2012] Agarwal, S. and Mierle, K. (2012). Ceres solver. http://ceres-solver.org. 72

[Akinlar and Topal, 2011] Akinlar, C. and Topal, C. (2011). Edlines: Real-time line segment detection by edge drawing. In *International Conference on Image Processing (ICIP)*. 7, 38

[Arandjelović et al., 2016] Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). NetVLAD: CNN architecture for weakly supervised place recognition. In *Computer Vision and Pattern Recognition (CVPR)*. 2, 20, 22, 105

[Asadi et al., 2019] Asadi, K., Ramshankar, H., Noghabaei, M., and Han, K. (2019). Real-time image localization and registration with bim using perspective alignment for indoor monitoring of construction. *Journal of Computing in civil Engineering*, 33(5):04019031. 90

[Baker et al., 2007] Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J., and Szeliski, R. (2007). A database and evaluation methodology for optical flow. In *International Conference on Computer Vision (ICCV)*. 68

[Balntas et al., 2016a] Balntas, V., Johns, E., Tang, L., and Mikolajczyk, K. (2016a). Pn-net: Conjoined triple deep network for learning local image descriptors. In *Computer Vision and Pattern Recognition (CVPR)*. 6

[Balntas et al., 2017] Balntas, V., Lenc, K., Vedaldi, A., and Mikolajczyk, K. (2017). Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Computer Vision and Pattern Recognition (CVPR)*. 24, 25, 26, 27, 73, 74, 81, 82, 83, 101, 102

[Balntas et al., 2016b] Balntas, V., Riba, E., Ponsa, D., and Mikolajczyk, K. (2016b). Learning local feature descriptors with triplets and shallow convolutional neural networks. In *British Machine Vision Conference (BMVC)*. 20, 44

[Balntas et al., 2016c] Balntas, V., Riba, E., Ponsa, D., and Mikolajczyk, K. (2016c). Learning local feature descriptors with triplets and shallow convolutional neural networks. In *British Machine Vision Conference (BMVC)*. 99

[Balntas et al., 2015] Balntas, V., Tang, L., and Mikolajczyk, K. (2015). Bold - binary online learned descriptor for efficient image matching. In *Computer Vision and Pattern Recognition (CVPR)*. 18

[Barath et al., 2021] Barath, D., Ding, Y., Kukelova, Z., and Larsson, V. (2021). Image stitching with locally shared rotation axis. *International Conference on 3D Vision (3DV)*. 2

[Barath and Matas, 2019] Barath, D. and Matas, J. (2019). Progressive-X: Efficient, anytime, multi-model fitting algorithm. In *International Conference on Computer Vision (ICCV)*. 71, 79

[Barroso-Laguna et al., 2019] Barroso-Laguna, A., Riba, E., Ponsa, D., and Mikolajczyk, K. (2019). Key.Net: Keypoint Detection by Handcrafted and Learned CNN Filters. In *International Conference on Computer Vision (ICCV)*. 6

[Baráth et al., 2023] Baráth, D., Mishkin, D., Polic, M., Förstner, W., and Matas, J. (2023). A large scale homography benchmark. In *Computer Vision and Pattern Recognition (CVPR)*. 102

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359. 5, 6

[Bay et al., 2005] Bay, H., Ferraris, V., and Van Gool, L. (2005). Wide-baseline stereo matching with line segments. In *Computer Vision and Pattern Recognition (CVPR)*. 8

[Bazin and Pollefeys, 2012] Bazin, J.-C. and Pollefeys, M. (2012). 3-line ransac for orthogonal vanishing point detection. In *International Conference on Intelligent Robots and Systems (IROS)*. 2

[Bazin et al., 2012] Bazin, J.-C., Seo, Y., Demonceaux, C., Vasseur, P., Ikeuchi, K., Kweon, I., and Pollefeys, M. (2012). Globally optimal line clustering and vanishing point estimation in manhattan world. In *Computer Vision and Pattern Recognition (CVPR)*. 2

[Brachmann and Rother, 2019] Brachmann, E. and Rother, C. (2019). Neural-guided RANSAC: Learning where to sample model hypotheses. In *International Conference on Computer Vision (ICCV)*. 102

[Brachmann and Rother, 2022] Brachmann, E. and Rother, C. (2022). Visual camera relocalization from rgb and rgb-d images using dsac. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 44. 76, 106

[Brown et al., 2010] Brown, M., Hua, G., and Winder, S. (2010). Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 27

[Brown and Lowe, 2003] Brown, M. A. and Lowe, D. G. (2003). Recognising panoramas. *International Conference on Computer Vision (ICCV)*. 2

[Brown and Lowe, 2007] Brown, M. A. and Lowe, D. G. (2007). Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision (IJCV)*. 2

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *European Conference on Computer Vision (ECCV)*. 3, 6

[Camposeco et al., 2018] Camposeco, F., Cohen, A., Pollefeys, M., and Sattler, T. (2018). Hybrid Camera Pose Estimation. In *Computer Vision and Pattern Recognition (CVPR)*. 76, 102, 104, 105

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 7

[Cao et al., 2020] Cao, B., de Araújo, A. F., and Sim, J. (2020). Unifying deep local and global features for image search. In *European Conference on Computer Vision (ECCV)*. 2

[Cavalli et al., 2020] Cavalli, L., Larsson, V., Oswald, M. R., Sattler, T., and Pollefeys, M. (2020). Adalam: Revisiting handcrafted outlier detection. *European Conference on Computer Vision (ECCV)*. 7

[Chen et al., 2021] Chen, H., Luo, Z., Zhang, J., Zhou, L., Bai, X., Hu, Z., Tai, C.-L., and Quan, L. (2021). Learning to match features with seeded graph matching network. In *International Conference on Computer Vision (ICCV)*. 7, 93

[Chen et al., 2022] Chen, H., Luo, Z., Zhou, L., Tian, Y., Zhen, M., Fang, T., McKinnon, D. N. R., Tsin, Y., and Quan, L. (2022). Aspanformer: Detector-free image matching with adaptive span transformer. In *European Conference on Computer Vision (ECCV)*. 93

[Christiansen et al., 2019] Christiansen, P. H., Kragh, M. F., Brodskiy, Y., and Karstoft, H. (2019). Unsuperpoint: End-to-end unsupervised interest point detector and descriptor. *arXiv*. 6, 40

[Cohen and Welling, 2016] Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*. 18

[Coughlan and Yuille, 1999] Coughlan, J. and Yuille, A. (1999). Manhattan world: compass direction from a single image by bayesian inference. In *International Conference on Computer Vision (ICCV)*. 2

[Dai et al., 2017] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017). ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Computer Vision and Pattern Recognition (CVPR)*. 102, 103

[Dai et al., 2021] Dai, X., Yuan, X., Gong, H., and Ma, Y. (2021). Fully convolutional line parsing. In *arXiv*. 8, 63, 107

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*. 73

[Denis et al., 2008] Denis, P., Elder, J. H., and Estrada, F. J. (2008). Efficient edge-based methods for estimating manhattan frames in urban imagery. In *European Conference on Computer Vision (ECCV)*. 47, 50, 79, 80, 81, 82, 83, 84

[DeTone et al., 2018] DeTone, D., Malisiewicz, T., and Rabinovich, A. (2018). Superpoint: Self-supervised interest point detection and description. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*. 6, 7, 17, 18, 19, 23, 24, 26, 27, 28, 29, 30, 39, 40, 41, 43, 47, 48, 53, 54, 56, 58, 67, 73, 76, 85, 94, 99, 105, 114, 127

[Dieny et al., 2011] Dieny, R., Thevenon, J., del rincon, J. M., and christophe Nebel, J. (2011). Bioinformatics inspired algorithm for stereo correspondence. In *International Conference on Computer Vision Theory and Applications (VISAPP)*. 39, 45

[Dusmanu et al., 2019] Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., and Sattler, T. (2019). D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. In *Computer Vision and Pattern Recognition (CVPR)*. 6, 7, 17, 18, 19, 23, 24, 26, 27, 39, 40, 44, 54

[Ebel et al., 2019] Ebel, P., Mishchuk, A., Yi, K. M., Fua, P., and Trulls, E. (2019). Beyond cartesian representations for local descriptors. In *International Conference on Computer Vision (ICCV)*. 6, 16

[Edstedt et al., 2023] Edstedt, J., Athanasiadis, I., Wadenbäck, M., and Felsberg, M. (2023). DKM: Dense kernelized feature matching for geometry estimation. In *Computer Vision and Pattern Recognition (CVPR)*. 93

[Elder et al., 2020] Elder, J. H., Almazán, E. J., Qian, Y., and Tal, R. (2020). MCMLSD: A probabilistic algorithm and evaluation framework for line segment detection. In *arXiv*. 7

[Fan et al., 2010] Fan, B., Wu, F., and Hu, Z. (2010). Line matching leveraged by point correspondences. In *Computer Vision and Pattern Recognition (CVPR)*. 40

[Fan et al., 2012a] Fan, B., Wu, F., and Hu, Z. (2012a). Robust line matching through line–point invariants. *Pattern Recognition*, 45(2):794–805. 9

[Fan et al., 2012b] Fan, B., Wu, F., and Hu, Z. (2012b). Robust line matching through line–point invariants. *Pattern Recognition*, 45. 40

[Fischler and Bolles, 1981a] Fischler, M. A. and Bolles, R. C. (1981a). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the Association for Computing Machinery (ACM)*, 24. 4, 32, 33

[Fischler and Bolles, 1981b] Fischler, M. A. and Bolles, R. C. (1981b). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communication of the ACM*, 24(6). 53

[Fu et al., 2020a] Fu, Q., Wang, J., Yu, H., Ali, I., Guo, F., He, Y., and Zhang, H. (2020a). PL-VINS: Real-time monocular visual-inertial SLAM with point and line features. In *arXiv*. 62

[Fu et al., 2020b] Fu, Q., Wang, J., Yu, H., Ali, I., Guo, F., and Zhang, H. (2020b). Pl-vins: Real-time monocular visual-inertial slam with point and line features. *arXiv*. 9

[Furukawa and Ponce, 2010] Furukawa, Y. and Ponce, J. (2010). Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 3

[Gao et al., 2021] Gao, S., Wan, J., Ping, Y., Zhang, X., Dong, S., Li, J., and Guo, Y. (2021). Pose refinement with joint optimization of visual points and lines. In *arXiv*. 8, 9, 62

[Gomez-Ojeda et al., 2019] Gomez-Ojeda, R., Moreno, F.-A., Zuñiga-Noël, D., Scaramuzza, D., and Gonzalez-Jimenez, J. (2019). PL-SLAM: A stereo SLAM system through the combination of points and line segments. *IEEE Transactions on Robotics*, 35. 9, 62, 90

[Gu et al., 2022] Gu, G., Ko, B., Go, S., Lee, S.-H., Lee, J., and Shin, M. (2022). Towards real-time and light-weight line segment detection. In *Conference on Artificial Intelligence (AAAI)*. 8, 63

[Guo et al., 2021] Guo, Z., Lu, H., Yu, Q., Guo, R., Xiao, J., and Yu, H. (2021). HDPL: a hybrid descriptor for points and lines based on graph neural networks. *Industrial Robot: the international journal of robotics research and application*. 93

[Han et al., 2015] Han, X., Leung, T., Jia, Y., Sukthankar, R., and Berg, A. (2015). Matchnet: Unifying feature and metric learning for patch-based matching. In *Computer Vision and Pattern Recognition (CVPR)*. 6

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*. 5, 16

[He et al., 2018] He, K., Lu, Y., and Sclaroff, S. (2018). Local descriptors optimized for average precision. In *Computer Vision and Pattern Recognition (CVPR)*. 6

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*. 49

[Heinly et al., 2015] Heinly, J., Schönberger, J. L., Dunn, E., and Frahm, J.-M. (2015). Reconstructing the World* in Six Days *(As Captured by the Yahoo 100 Million Image Dataset). In *Computer Vision and Pattern Recognition (CVPR)*. 2, 16

[Hirose and Saito, 2012] Hirose, K. and Saito, H. (2012). Fast line description for line-based slam. In *British Machine Vision Conference (BMVC)*. 8

[Hofer et al., 2017] Hofer, M., Maurer, M., and Bischof, H. (2017). Efficient 3d scene abstraction using line segments. *Computer Vision and Image Understanding (CVIU)*, 157. 62, 75, 76, 77, 90

[Holynski et al., 2020] Holynski, A., Geraghty, D., Frahm, J.-M., Sweeney, C., and Szeliski, R. (2020). Reducing drift in structure from motion using extended features. *arXiv*. 9, 38

[Hough, 1962] Hough, P. V. (1962). Method and means for recognizing complex patterns. US Patent 3,069,654. 7

[Huang et al., 2018] Huang, K., Wang, Y., Zhou, Z., Ding, T., Gao, S., and Ma, Y. (2018). Learning to parse wireframes in images of man-made environments. In *Computer Vision and Pattern Recognition (CVPR)*. 8, 38, 39, 47, 48, 49, 51, 52, 53, 54, 57, 58, 63, 67, 72, 80, 81, 107

[Huang et al., 2020] Huang, S., Qin, F., Xiong, P., Ding, N., He, Y., and Liu, X. (2020). Tp-lsd: Tri-points based line segment detector. In *European Conference on Computer Vision (ECCV)*. 8, 38, 49, 57, 63, 73, 75, 76, 80, 81

[Huang et al., 2021] Huang, Z., Zhou, H., Li, Y., Yang, B., Xu, Y., Zhou, X., Bao, H., Zhang, G., and Li, H. (2021). VS-Net: Voting with segmentation for visual localization. In *Computer Vision and Pattern Recognition (CVPR)*. 65

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*. 68

[Jiang et al., 2021] Jiang, W., Trulls, E., Hosang, J., Tagliasacchi, A., and Yi, K. M. (2021). COTR: Correspondence Transformer for Matching Across Images. In *International Conference on Computer Vision (ICCV)*. 91, 93

[Kabsch, 1976] Kabsch, W. (1976). A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923. 104

[Kaliroff and Gilboa, 2019] Kaliroff, D. and Gilboa, G. (2019). Self-supervised unconstrained illumination invariant representation. In *arXiv*. 18

[Kendall and Cipolla, 2017] Kendall, A. and Cipolla, R. (2017). Geometric loss functions for camera pose regression with deep learning. In *Computer Vision and Pattern Recognition (CVPR)*. 44, 55

[Kendall et al., 2018] Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Computer Vision and Pattern Recognition (CVPR)*. 44, 55

[Kim and Lee, 2010] Kim, H. and Lee, S. (2010). A novel line matching method based on intersection context. In *International Conference on Robotics and Automation (ICRA)*. 40

[Kingma and Ba, 2014] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*. 23, 46, 72

[Kluger et al., 2020] Kluger, F., Brachmann, E., Ackermann, H., Rother, C., Yang, M. Y., and Rosenhahn, B. (2020). CONSAC: Robust multi-model fitting by conditional sample consensus. In *Computer Vision and Pattern Recognition (CVPR)*. 79, 80, 81, 82

[Kropp et al., 2018] Kropp, C., Koch, C., and König, M. (2018). Interior construction state recognition with 4D BIM registered image sequences. *Automation in Construction*, 86. 90

[Kuhn, 1955] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97. 98

[Kukelova et al., 2016] Kukelova, Z., Heller, J., and Fitzgibbon, A. (2016). Efficient intersection of three quadrics and applications in computer vision. In *Computer Vision and Pattern Recognition (CVPR)*. 76, 105

[Lange et al., 2019a] Lange, M., Raisch, C., and Schilling, A. (2019a). LVO: Line only stereo visual odometry. In *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 62

[Lange et al., 2020] Lange, M., Raisch, C., and Schilling, A. (2020). Wld: A wavelet and learning based line descriptor for line feature matching. In *International Symposium on Vision, Modeling, and Visualization (VMV)*. 8, 39, 43, 50, 56

[Lange et al., 2019b] Lange, M., Schweinfurth, F., and Schilling, A. (2019b). DLD: A Deep Learning Based Line Descriptor for Line Feature Matching. In *International Conference on Intelligent Robots and Systems (IROS)*. 8, 39, 43, 50

[Larsson, 2020] Larsson, V. (2020). PoseLib - Minimal Solvers for Camera Pose Estimation. 76, 105

[Lebeda et al., 2012] Lebeda, K., Matas, J., and Chum, O. (2012). Fixing the Locally Optimized RANSAC. In *British Machine Vision Conference (BMVC)*. 53, 73

[Lepetit et al., 2009] Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision (IJCV)*. 2

[Li et al., 2021] Li, H., Yu, H., Wang, J., Yang, W., Yu, L., and Scherer, S. (2021). ULSD: Unified line segment detection across pinhole, fisheye, and spherical cameras. *Journal of Photogrammetry and Remote Sensing (ISPRS)*, 178. 63

[Li et al., 2019] Li, H., Zhao, J., Bazin, J.-C., Chen, W., Liu, Z., and Liu, Y.-H. (2019). Quasi-globally optimal and efficient vanishing point estimation in manhattan world. In *International Conference on Computer Vision (ICCV)*. 2

[Li et al., 2016a] Li, K., Yao, J., Lu, M., Heng, Y., Wu, T., and Li, Y. (2016a). Line segment matching: a benchmark. In *Winter Conference on Applications of Computer Vision (WACV)*. 9

[Li et al., 2016b] Li, K., Yao, J., Lu, M., Heng, Y., Wu, T., and Li, Y. (2016b). Line segment matching: a benchmark. In *Winter Conference on Applications of Computer Vision (WACV)*. 40

[Li et al., 2014] Li, K., Yao, J., and Lu, X. (2014). Robust line matching based on ray-point-ray structure descriptor. In *Asian Conference on Computer Vision (ACCV)*, pages 554–569. 40

[Li et al., 2016c] Li, K., Yao, J., Lu, X., Li, L., and Zhang, Z. (2016c). Hierarchical line matching based on line–junction–line structure descriptor and local homography estimation. *Neurocomputing*, 184:207–220. 9, 91, 94

[Li et al., 2016d] Li, K., Yao, J., Lu, X., Li, L., and Zhang, Z. (2016d). Hierarchical line matching based on line–junction–line structure descriptor and local homography estimation. *Neurocomputing*, 184:207–220. 40

[Li and Snavely, 2018] Li, Z. and Snavely, N. (2018). MegaDepth: Learning single-view depth prediction from internet photos. In *Computer Vision and Pattern Recognition (CVPR)*. 18, 72, 81, 99, 109

[Lim et al., 2021] Lim, H., Jeon, J., and Myung, H. (2021). Uv-slam: Unconstrained line-based slam using vanishing points for structural mapping. *IEEE Robotics and Automation Letters*, 7. 9

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2014). Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*. 23

[Lin et al., 2020] Lin, Y., Pintea, S. L., and van Gemert, J. C. (2020). Deep hough-transform line priors. In *European Conference on Computer Vision (ECCV)*. 8, 45, 49, 57, 63

[Lindeberg, 1998] Lindeberg, T. (1998). Feature detection with automatic scale selection. *International Journal of Computer Vision (IJCV)*. 5

[Lindenberger et al., 2021] Lindenberger, P., Sarlin, P.-E., Larsson, V., and Pollefeys, M. (2021). Pixel-Perfect Structure-from-Motion with Featuremetric Refinement. In *International Conference on Computer Vision (ICCV)*. 63, 66

[Lindenberger et al., 2023] Lindenberger, P., Sarlin, P.-E., and Pollefeys, M. (2023). LightGlue: Local Feature Matching at Light Speed. In *International Conference on Computer Vision (ICCV)*. 7, 93

[Liu et al., 2010] Liu, H.-M., Wang, Z.-H., and Deng, C. (2010). Extend point descriptors for line, curve and region matching. In *International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 1, pages 214 – 219. 8

[Liu et al., 2023] Liu, S., Yu, Y., Pautrat, R., Pollefeys, M., and Larsson, V. (2023). 3d line mapping revisited. In *Computer Vision and Pattern Recognition (CVPR)*. 9, 10, 62, 76, 86, 112

[Liu et al., 2019] Liu, Y., Shen, Z., Lin, Z., Peng, S., Bao, H., and Zhou, X. (2019). Gift: Learning transformation-invariant dense visual descriptors via group cnns. In *Neural Information Processing Systems (NeurIPS)*. 17, 18, 27

[Lourakis et al., 1998] Lourakis, M. I. A., Halkidis, S. T., and Orphanoudakis, S. C. (1998). Matching disparate views of planar surfaces using projective invariants. In *British Machine Vision Conference (BMVC)*. 40

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60. 5, 6, 7, 16, 22, 28, 29, 30, 50, 54, 91

[Luo et al., 2019] Luo, Z., Shen, T., Zhou, L., Zhang, J., Yao, Y., Li, S., Fang, T., and Quan, L. (2019). Contextdesc: Local descriptor augmentation with cross-modality context. In *Computer Vision and Pattern Recognition (CVPR)*. 6, 19

[Luo et al., 2018] Luo, Z., Shen, T., Zhou, L., Zhu, S., Zhang, R., Yao, Y., Fang, T., and Quan, L. (2018). Geodesc: Learning local descriptors by integrating geometry constraints. In *European Conference on Computer Vision (ECCV)*. 6

[Luo et al., 2020] Luo, Z., Zhou, L., Bai, X., Chen, H., Zhang, J., Yao, Y., Li, S., Fang, T., and Quan, L. (2020). Aslfeat: Learning local features of accurate shape and localization. *Computer Vision and Pattern Recognition (CVPR)*. 6, 40, 54

[Ma et al., 2021] Ma, Q., Jiang, G., Wu, J., Cai, C., Lai, D., Bai, Z., and Chen, H. (2021). WGLSM: An end-to-end line matching network based on graph convolution. *Neurocomputing*, 453:195–208. 93

[Malis and Vargas, 2007] Malis, E. and Vargas, M. (2007). Deeper understanding of the homography decomposition for vision-based control. Research Report RR-6303, INRIA. 102

[Matas et al., 2002] Matas, J., Chum, O., Urban, M., and Pajdla, T. (2002). Robust wide-baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference (BMVC)*. 5

[Mateus et al., 2022] Mateus, A., Tahri, O., Aguiar, A. P., Lima, P. U., and Miraldo, P. (2022). On incremental structure from motion using lines. *IEEE Transactions on Robotics*, 38. 62

[Meng et al., 2020] Meng, Q., Zhang, J., Hu, Q., He, X., and Yu, J. (2020). LGNN: A context-aware line segment detector. In *ACM International Conference on Multimedia*. 8

[Micusik and Wildenauer, 2014] Micusik, B. and Wildenauer, H. (2014). Structure from motion with line segments under relaxed endpoint constraints. In *International Conference on 3D Vision (3DV)*, volume 1. 48

[Micusik and Wildenauer, 2017] Micusik, B. and Wildenauer, H. (2017). Structure from motion with line segments under relaxed endpoint constraints. *International Journal of Computer Vision (IJCV)*, 124. 62, 63

[Mikolajczyk and Schmid, 2002] Mikolajczyk, K. and Schmid, C. (2002). An affine invariant interest point detector. In *European Conference on Computer Vision (ECCV)*. 5

[Mikolajczyk et al., 2005] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Gool, L. V. (2005). A comparison of affine region detectors. *International Journal of Computer Vision (IJCV)*. 16

[Mildenhall et al., 2020] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*. 3

[Milgram, 1975] Milgram, D. (1975). Computer methods for creating photomosaics. *IEEE Transactions on Computers*. 2

[Mishchuk et al., 2017] Mishchuk, A., Mishkin, D., Radenović, F., and Matas, J. (2017). Working hard to know your neighbor's margins:local descriptor learning loss. In *Neural Information Processing Systems (NIPS)*. 6, 20, 26, 44, 99

[Mishkin et al., 2018] Mishkin, D., Radenovic, F., and Matas, J. (2018). Repeatability is not enough: Learning affine regions via discriminability. In *European Conference on Computer Vision (ECCV)*. 6, 16

[Mitra et al., 2018] Mitra, R., Doiphode, N., Gautam, U., Narayan, S., Ahmed, S., Chandran, S., and Jain, A. (2018). A Large Dataset for Improving Patch Matching. *arXiv*. 26

[Moravec, 1977] Moravec, H. P. (1977). Towards automatic visual obstacle avoidance. In *International Joint Conference on Artificial Intelligence*. 5

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*. 3

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015). Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5). 102

[Murmann et al., 2019] Murmann, L., Gharbi, M., Aittala, M., and Durand, F. (2019). A multi-illumination dataset of indoor object appearance. In *International Conference on Computer Vision (ICCV)*. 19, 23

[Nathan Silberman and Fergus, 2012] Nathan Silberman, Derek Hoiem, P. K. and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision (ECCV)*. 79, 80

[Needleman and Wunsch, 1970] Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453. 39, 45

[Newell et al., 2016] Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision (ECCV)*. 45, 49

[Nistér, 2003] Nistér, D. (2003). An efficient solution to the five-point relative pose problem. In *Computer Vision and Pattern Recognition (CVPR)*. 103

[Noh et al., 2016] Noh, H., de Araújo, A. F., Sim, J., Weyand, T., and Han, B. (2016). Large-scale image retrieval with attentive deep local features. *International Conference on Computer Vision (ICCV)*. 2

[Ono et al., 2018] Ono, Y., Trulls, E., Fua, P., and Yi, K. M. (2018). Lf-net: Learning local features from images. In *Neural Information Processing Systems (NIPS)*. 6, 16, 40

[Park et al., 2020] Park, J., Joo, K., Hu, Z., Liu, C.-K., and Kweon, I. S. (2020). Non-local spatial propagation network for depth completion. In *Proc. of European Conference on Computer Vision (ECCV)*. 51

[Park et al., 2019] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019). Deepsdf: Learning continuous signed distance functions for shape representation. In *Computer Vision and Pattern Recognition (CVPR)*. 3

[Pautrat et al., 2023a] Pautrat, R., Barath, D., Larsson, V., Oswald, M. R., and Pollefeys, M. (2023a). Deeplsd: Line segment detection and refinement with deep image gradients. In *Computer Vision and Pattern Recognition (CVPR)*. 10, 61, 112

[Pautrat et al., 2020] Pautrat, R., Larsson, V., Oswald, M. R., and Pollefeys, M. (2020). Online invariance selection for local feature descriptors. In *European Conference on Computer Vision (ECCV)*. 10, 15, 73, 74, 83

[Pautrat et al., 2021] Pautrat, R., Lin, J.-T., Larsson, V., Oswald, M. R., and Pollefeys, M. (2021). Sold2: Self-supervised occlusion-aware line description and detection. In *Computer Vision and Pattern Recognition (CVPR)*. 10, 37, 63, 67, 73, 76, 99, 100, 109, 112

[Pautrat et al., 2023b] Pautrat, R., Liu, S., Hruby, P., Pollefeys, M., and Barath, D. (2023b). Vanishing point estimation in uncalibrated images with prior gravity direction. In *International Conference on Computer Vision (ICCV)*. 2, 10

[Pautrat et al., 2023c] Pautrat, R., Suárez, I., Yu, Y., Pollefeys, M., and Larsson, V. (2023c). GlueStick: Robust image matching by sticking points and lines together. In *International Conference on Computer Vision (ICCV)*. 10, 89

[Pumarola et al., 2017] Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., and Moreno-Noguer, F. (2017). PL-SLAM: Real-time monocular visual SLAM with points and lines. In *International Conference on Robotics and Automation (ICRA)*. 9, 62, 90

[Qin et al., 2018] Qin, T., Li, P., and Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*. 3

[Quan and Lan, 1999] Quan, L. and Lan, Z.-D. (1999). Linear n-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 2

[Quan et al., 2021] Quan, M., Chai, Z., and Liu, X. (2021). LOF: Structure-aware line tracking based on optical flow. In *arXiv*. 62

[Radenović et al., 2018] Radenović, F., Iscen, A., Tolias, G., Avrithis, Y., and Chum, O. (2018). Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *Computer Vision and Pattern Recognition (CVPR)*. 99, 109, 110

[Ramalingam et al., 2015a] Ramalingam, S., Antunes, M., Snow, D., Hee Lee, G., and Pillai, S. (2015a). Line-sweep: Cross-ratio for wide-baseline matching and 3D reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*. 9

[Ramalingam et al., 2015b] Ramalingam, S., Antunes, M., Snow, D., Hee Lee, G., and Pillai, S. (2015b). Line-sweep: Cross-ratio for wide-baseline matching and 3d reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*. 40

[Ren et al., 2022] Ren, G., Cao, Z., Liu, X., Tan, M., and Yu, J. (2022). Plj-slam: Monocular visual slam with points, lines, and junctions of coplanar lines. *IEEE Sensors Journal*, 22. 9

[Revaud et al., 2019] Revaud, J., Weinzaepfel, P., de Souza, C. R., and Humenberger, M. (2019). R2D2: repeatable and reliable detector and descriptor. In *Neural Information Processing Systems (NeurIPS)*. 6, 7, 17, 18, 19, 23, 24, 26, 27, 28, 29, 30, 39, 40

[Revaud et al., 2015] Revaud, J., Weinzaepfel, P., Harchaoui, Z., and Schmid, C. (2015). EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Computer Vision and Pattern Recognition (CVPR)*. 18

[Roberts et al., 2021] Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M. A., Paczan, N., Webb, R., and Susskind, J. M. (2021). Hypersim: A photorealistic synthetic

dataset for holistic indoor scene understanding. In *International Conference on Computer Vision (ICCV)*. 75, 76, 77

[Rocco et al., 2018] Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., and Sivic, J. (2018). Neighbourhood consensus networks. In *Neural Information Processing Systems (NIPS)*. 94, 97

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 68, 69

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European Conference on Computer Vision (ECCV)*. 5

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. R. (2011). Orb: An efficient alternative to sift or surf. *International Conference on Computer Vision (ICCV)*. 3, 6

[Salaün et al., 2016] Salaün, Y., Marlet, R., and Monasse, P. (2016). Multiscale line segment detector for robust and accurate SfM. In *International Conference on Pattern Recognition (ICPR)*. 7

[Sarlin, 2020] Sarlin, P.-E. (2020). Visual localization made easy with hloc. https://github.com/cvg/Hierarchical-Localization/. 76, 105, 112, 114

[Sarlin et al., 2019] Sarlin, P.-E., Cadena, C., Siegwart, R., and Dymczyk, M. (2019). From coarse to fine: Robust hierarchical localization at large scale. In *Computer Vision and Pattern Recognition (CVPR)*. 2, 19, 40, 44, 55, 76, 105, 112, 114

[Sarlin et al., 2020a] Sarlin, P.-E., DeTone, D., Malisiewicz, T., and Rabinovich, A. (2020a). SuperGlue: Learning feature matching with graph neural networks. In *Computer Vision and Pattern Recognition (CVPR)*. 7, 76, 91, 92, 93, 96, 97, 100, 101, 102, 105, 109, 114

[Sarlin et al., 2020b] Sarlin, P.-E., DeTone, D., Malisiewicz, T., and Rabinovich, A. (2020b). Superglue: Learning feature matching with graph neural networks. In *Computer Vision and Pattern Recognition (CVPR)*. 19

[Sarlin et al., 2021] Sarlin, P.-E., Unagar, A., Larsson, M., Germain, H., Toft, C., Larsson, V., Pollefeys, M., Lepetit, V., Hammarstrand, L., Kahl, F., and Sattler, T. (2021). Back to the Feature: Learning Robust Camera Localization from Pixels to Pose. In *Computer Vision and Pattern Recognition (CVPR)*. 63, 66

[Sattler et al., 2019] Sattler, T. et al. (2019). RansacLib - A Template-based *SAC Implementation. 76, 102, 104, 105

[Sattler et al., 2018] Sattler, T., Maddern, W., Toft, C., Torii, A., Hammarstrand, L., Stenborg, E., Safari, D., Okutomi, M., Pollefeys, M., Sivic, J., Kahl, F., and Pajdla, T. (2018).

Benchmarking 6dof outdoor visual localization in changing conditions. In *Computer Vision and Pattern Recognition (CVPR)*. 16, 32

[Sattler et al., 2017] Sattler, T., Torii, A., Sivic, J., Pollefeys, M., Taira, H., Okutomi, M., and Pajdla, T. (2017). Are large-scale 3d models really necessary for accurate visual localization? In *Computer Vision and Pattern Recognition (CVPR)*. 16

[Schmid and Zisserman, 1997a] Schmid, C. and Zisserman, A. (1997a). Automatic line matching across views. In *Computer Vision and Pattern Recognition (CVPR)*, pages 666–671. 9, 91

[Schmid and Zisserman, 1997b] Schmid, C. and Zisserman, A. (1997b). Automatic line matching across views. In *Computer Vision and Pattern Recognition (CVPR)*. 39, 40

[Schonberger and Frahm, 2016] Schonberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited. In *Computer Vision and Pattern Recognition (CVPR)*. 2, 16

[Schönberger and Frahm, 2016] Schönberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 51

[Schönberger et al., 2016] Schönberger, J. L., Zheng, E., Pollefeys, M., and Frahm, J.-M. (2016). Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*. 3

[Schöps et al., 2017] Schöps, T., Schönberger, J. L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., and Geiger, A. (2017). A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Computer Vision and Pattern Recognition (CVPR)*. 51, 52, 56, 100, 109, 110, 111, 112, 113

[Schönberger et al., 2017] Schönberger, J. L., Pollefeys, M., Geiger, A., and Sattler, T. (2017). Semantic visual localization. In *Computer Vision and Pattern Recognition (CVPR)*. 16

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. *Computer Vision and Pattern Recognition (CVPR)*. 5

[Shi et al., 2016] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Computer Vision and Pattern Recognition (CVPR)*. 45

[Shi et al., 2022] Shi, Y., Cai, J.-X., Shavit, Y., Mu, T.-J., Feng, W., and Zhang, K. (2022). ClusterGNN: Cluster-based coarse-to-fine graph neural network for efficient feature matching. In *Computer Vision and Pattern Recognition (CVPR)*. 7, 93, 99

[Shotton et al., 2013] Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., and Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in rgb-d

images. In *Computer Vision and Pattern Recognition (CVPR)*. 76, 79, 106, 107, 108, 112, 115

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*. 23

[Sinkhorn and Knopp, 1967] Sinkhorn, R. and Knopp, P. (1967). Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348. 97

[Sobel and Feldman, 1968] Sobel, I. and Feldman, G. (1968). *A 3x3 Isotropic Gradient Operator for Image Processing*. Stanford Artificial Intelligence Project (SAIL). 5

[Suárez et al., 2021] Suárez, I., Buenaposada, J. M., and Baumela, L. (2021). Revisiting binary local image description for resource limited devices. *IEEE Robotics and Automation Letters (RAL)*, 6(4):8317–8324. 99

[Sun et al., 2023] Sun, J., Ji, L., and Zhu, J. (2023). Shared coupling-bridge scheme for weakly supervised local feature learning. *IEEE Transactions on Multimedia*. 6

[Sun et al., 2021] Sun, J., Shen, Z., Wang, Y., Bao, H., and Zhou, X. (2021). LoFTR: Detector-free local feature matching with transformers. In *Computer Vision and Pattern Recognition (CVPR)*. 91, 93, 94, 97, 99, 101

[Suárez et al., 2022] Suárez, I., Buenaposada, J. M., and Baumela, L. (2022). ELSED: Enhanced line segment drawing. *Pattern Recognition*. 7, 73, 83

[Szeliski, 2004] Szeliski, R. (2004). *Image Alignment and Stitching: A Tutorial*. Now Foundations and Trends. 2

[Taira et al., 2018] Taira, H., Okutomi, M., Sattler, T., Cimpoi, M., Pollefeys, M., Sivic, J., Pajdla, T., and Torii, A. (2018). Inloc: Indoor visual localization with dense matching and view synthesis. In *Computer Vision and Pattern Recognition (CVPR)*. 106, 107

[Tang et al., 2020] Tang, J., Kim, H., Guizilini, V., Pillai, S., and Ambrus, R. (2020). Neural outlier rejection for self-supervised keypoint learning. In *International Conference on Learning Representations (ICLR)*. 6

[Tardif, 2009] Tardif, J.-P. (2009). Non-iterative approach for fast and accurate vanishing point detection. In *International Conference on Computer Vision (ICCV)*. 62, 71, 79

[Taylor and Kriegman, 1995] Taylor, C. J. and Kriegman, D. J. (1995). Structure and motion from line segments in multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 17(11):1021–1032. 38

[Teplyakov et al., 2022] Teplyakov, L., Erlygin, L., and Shvets, E. (2022). Lsdnet: Trainable modification of lsd algorithm for real-time line segment detection. *IEEE Access*, 10. 63, 65, 73, 75

[Tian et al., 2020] Tian, Y., Balntas, V., Ng, T., Barroso, A., Demiris, Y., and Mikolajczyk, K. (2020). D2d: Keypoint extraction with describe to detect approach. *Asian Conference on Computer Vision (ACCV)*. 6, 40

[Tian et al., 2017] Tian, Y., Fan, B., and Wu, F. (2017). L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *Computer Vision and Pattern Recognition (CVPR)*. 6

[Tian et al., 2019] Tian, Y., Yu, X., Fan, B., Wu, F., Heijnen, H., and Balntas, V. (2019). Sosnet: Second order similarity regularization for local descriptor learning. In *Computer Vision and Pattern Recognition (CVPR)*. 6, 26

[Tola et al., 2010a] Tola, E., Lepetit, V., and Fua, P. (2010a). Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32. 6

[Tola et al., 2010b] Tola, E., Lepetit, V., and Fua, P. (2010b). Daisy: An efficient dense descriptor applied to wide baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32. 16

[Truong et al., 2021] Truong, P., Danelljan, M., Van Gool, L., and Timofte, R. (2021). Learning accurate dense correspondences and when to trust them. In *Computer Vision and Pattern Recognition (CVPR)*. 91, 93

[Tyszkiewicz et al., 2020] Tyszkiewicz, M., Fua, P., and Trulls, E. (2020). Disk: Learning local features with policy gradient. *Neural Information Processing Systems (NeurIPS)*, 33. 6

[Vakhitov and Lempitsky, 2019] Vakhitov, A. and Lempitsky, V. (2019). Learnable line segment descriptor for visual slam. *IEEE Access*, 7. 8, 39, 40, 50, 55, 56

[van Drongelen, 2007] van Drongelen, W. (2007). Wavelet analysis: Time domain properties. *Signal Processing for Neuroscientists*, pages 245–263. 6

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Neural Information Processing Systems (NIPS)*. 96

[Verdie et al., 2014] Verdie, Y., Yi, K. M., Fua, P. V., and Lepetit, V. (2014). Tilde: A temporally invariant learned detector. *Computer Vision and Pattern Recognition (CVPR)*. 5

[Verhagen et al., 2014] Verhagen, B., Timofte, R., and Van Gool, L. (2014). Scale-invariant line descriptors for wide baseline matching. In *Winter Conference on Applications of Computer Vision (WACV)*. 8

[Von Gioi et al., 2008] Von Gioi, R. G., Jakubowicz, J., Morel, J.-M., and Randall, G. (2008). Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(4):722–732. 7, 38, 49, 52, 54, 56, 57, 63, 64, 65, 66, 67, 71, 73, 75, 76, 83, 85, 94, 99, 105, 107, 109

[Wang et al., 2021] Wang, B., Chen, C., Cui, Z., Qin, J., Lu, C. X., Yu, Z., Zhao, P., Dong, Z., Zhu, F., Trigoni, N., and Markham, A. (2021). P2-net: Joint description and detection of local features for pixel and point matching. In *International Conference on Computer Vision (ICCV)*. 6

[Wang et al., 2023] Wang, C., Zhang, G., Cheng, Z., and Zhou, W. (2023). Rethinking low-level features for interest point detection and description. In *Asian Conference on Computer Vision (ACCV)*. 6

[Wang et al., 2009a] Wang, L., Neumann, U., and You, S. (2009a). Wide-baseline image matching using line signatures. In *International Conference on Computer Vision (ICCV)*. 9, 40

[Wang et al., 2022] Wang, Q., Zhang, J., Yang, K., Peng, K., and Stiefelhagen, R. (2022). Matchformer: Interleaving attention in transformers for feature matching. In *Asian Conference on Computer Vision (ACCV)*. 93

[Wang et al., 2020] Wang, Q., Zhou, X., Hariharan, B., and Snavely, N. (2020). Learning feature descriptors using camera pose supervision. In *European Conference on Computer Vision (ECCV)*. 7

[Wang et al., 2009b] Wang, Z., Liu, H.-M., and Wu, F. (2009b). Hld: A robust descriptor for line matching. In *Conference on Computer-Aided Design and Computer Graphics*. 8

[Wang et al., 2009c] Wang, Z., Wu, F., and Hu, Z. (2009c). Msld: A robust descriptor for line matching. *Pattern Recognition*, 42(5):941–953. 8, 39

[Weng et al., 1992] Weng, J., Huang, T. S., and Ahuja, N. (1992). Motion and structure from line correspondences; closed-form solution, uniqueness, and optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(3):318–336. 38

[Wijmans and Furukawa, 2017] Wijmans, E. and Furukawa, Y. (2017). Exploiting 2D floorplan for building-scale panorama rgbd alignment. In *Computer Vision and Pattern Recognition (CVPR)*. 106

[William T. Freeman, 1994] William T. Freeman, M. R. (1994). Orientation histograms for hand gesture recognition. Technical Report TR94-03, MERL - Mitsubishi Electric Research Laboratories. 6

[Wu et al., 2008] Wu, C., Li, X., Frahm, J., and Pollefeys, M. (2008). 3d model matching with viewpoint-invariant patches (vip). In *Computer Vision and Pattern Recognition (CVPR)*. 17

[Xiao et al., 2012] Xiao, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2012). Recognizing scene viewpoint using panoramic place representation. In *Computer Vision and Pattern Recognition (CVPR)*. 103, 104, 105, 106

[Xu et al., 2017] Xu, C., Zhang, L., Cheng, L., and Koch, R. (2017). Pose estimation from line correspondences: A complete analysis and a series of solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 39(6). 90

[Xu et al., 2023] Xu, K., Hao, Y., Yuan, S., Wang, C., and Xie, L. (2023). Airvo: An illumination-robust point-line visual odometry. In *International Conference on Intelligent Robots and Systems (IROS)*. 9

[Xu et al., 2021a] Xu, Y., Xu, W., Cheung, D., and Tu, Z. (2021a). Line segment detection using transformers without edges. In *Computer Vision and Pattern Recognition (CVPR)*. 8

[Xu et al., 2021b] Xu, Y., Xu, W., Cheung, D., and Tu, Z. (2021b). Line segment detection using transformers without edges. In *Computer Vision and Pattern Recognition (CVPR)*. 107

[Xue et al., 2019] Xue, N., Bai, S., Wang, F., Xia, G.-S., Wu, T., and Zhang, L. (2019). Learning attraction field representation for robust line segment detection. In *Computer Vision and Pattern Recognition (CVPR)*. 8, 63, 65, 66, 67

[Xue et al., 2020] Xue, N., Wu, T., Bai, S., Wang, F., Xia, G.-S., Zhang, L., and Torr, P. H. (2020). Holistically-attracted wireframe parsing. In *Computer Vision and Pattern Recognition (CVPR)*. 8, 38, 45, 48, 49, 57, 63, 64, 65, 66, 67, 70, 73, 80, 81, 82, 107

[Xue et al., 2022] Xue, N., Wu, T., Bai, S., Wang, F.-D., Xia, G.-S., Zhang, L., and Torr, P. H. (2022). Holistically-attracted wireframe parsing: From supervised to self-supervised learning. *arXiv*. 8, 65, 73

[Yang et al., 2020] Yang, T.-Y., Nguyen, D.-K., Heijnen, H., and Balntas, V. (2020). Ur2kid: Unifying retrieval, keypoint detection, and keypoint description without local correspondence supervision. In *arXiv*. 19, 40

[Yi et al., 2016] Yi, K., Trulls, E., Lepetit, V., and Fua, P. (2016). Lift: Learned invariant feature transform. In *European Conference on Computer Vision (ECCV)*. 5, 6, 16, 18

[Yi et al., 2018] Yi, K. M., Trulls, E., Ono, Y., Lepetit, V., Salzmann, M., and Fua, P. (2018). Learning to find good correspondences. In *Computer Vision and Pattern Recognition (CVPR)*. 102

[Yoon and Kim, 2021] Yoon, S. and Kim, A. (2021). Line as a visual sentence: Context-aware line descriptor for visual localization. *IEEE Robotics and Automation Letters (RAL)*, 6. 8, 91, 93, 99, 104

[Zeng et al., 2020] Zeng, H., Joseph, K., Vest, A., and Furukawa, Y. (2020). Bundle pooling for polygonal architecture segmentation problem. In *Computer Vision and Pattern Recognition (CVPR)*. 90

[Zhang et al., 2021a] Zhang, H., Luo, Y., Qin, F., He, Y., and Liu, X. (2021a). Elsd: Efficient line segment detector and descriptor. In *International Conference on Computer Vision (ICCV)*. 8, 63

[Zhang et al., 2019a] Zhang, J., Sun, D., Luo, Z., Yao, A., Zhou, L., Shen, T., Chen, Y., Quan, L., and Liao, H. (2019a). Learning two-view correspondences and geometry using order-aware network. *International Conference on Computer Vision (ICCV)*. 7

[Zhang and Koch, 2013] Zhang, L. and Koch, R. (2013). An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24(7). 8, 9, 39, 50, 56, 73, 91, 99

[Zhang et al., 2015] Zhang, L., Lu, H., Hu, X., and Koch, R. (2015). Vanishing point estimation and line classification in a manhattan world with a unifying camera model. *International Journal of Computer Vision (IJCV)*, 117. 2

[Zhang et al., 2021b] Zhang, Y., Wei, D., and Li, Y. (2021b). AG3line: Active grouping and geometry-gradient combined validation for fast line segment extraction. *Pattern Recognition*, 113. 7

[Zhang et al., 2019b] Zhang, Z., Li, Z., Bi, N., Zheng, J., Wang, J., Huang, K., Luo, W., Xu, Y., and Gao, S. (2019b). Ppgnet: Learning point-pair graph for line segment detection. In *Computer Vision and Pattern Recognition (CVPR)*. 8

[Zhao et al., 2023] Zhao, X., Wu, X., Chen, W., Chen, P. C. Y., Xu, Q., and Li, Z. (2023). Aliked: A lighter keypoint and descriptor extraction network via deformable transformation. *IEEE Transactions on Instrumentation & Measurement*. 6, 7

[Zhao et al., 2021] Zhao, X., Wu, X., Miao, J., Chen, W., Chen, P. C. Y., and Li, Z. (2021). Alike: Accurate and lightweight keypoint detection and descriptor extraction. *IEEE Transactions on Multimedia*. 6

[Zhou et al., 2016] Zhou, H., Sattler, T., and Jacobs, D. W. (2016). Evaluating local features for day-night matching. In *European Conference on Computer Vision Workshops (ECCVW)*. 16, 17, 27, 28, 83, 84, 127, 128

[Zhou et al., 2022] Zhou, L., Huang, G. P., Mao, Y., Wang, S., and Kaess, M. (2022). Edplvo: Efficient direct point-line visual odometry. *International Conference on Robotics and Automation (ICRA)*. 9

[Zhou et al., 2018] Zhou, L., Ye, J., and Kaess, M. (2018). A stable algebraic camera pose estimation for minimal configurations of 2d/3d point and line correspondences. In *Asian Conference on Computer Vision (ACCV)*. 76, 105

[Zhou et al., 2019a] Zhou, Y., Qi, H., and Ma, Y. (2019a). End-to-end wireframe parsing. In *International Conference on Computer Vision (ICCV)*. 8, 38, 39, 45, 47, 48, 49, 57, 63

[Zhou et al., 2019b] Zhou, Y., Qi, H., Zhai, Y., Sun, Q., Chen, Z., Wei, L.-Y., and Ma, Y. (2019b). Learning to reconstruct 3d manhattan wireframes from a single image. In *International Conference on Computer Vision (ICCV)*. 38, 42, 90

[Zuo et al., 2017] Zuo, X., Xie, X., Liu, Y., and Huang, G. (2017). Robust visual SLAM with point and line features. In *International Conference on Intelligent Robots and Systems (IROS)*. 9, 62, 90

List of Acronyms

3D 3 dimensions

AF Angle Field

AFM Attraction Field Map

AP Average Precision

AR Augmented Reality

AUC Area Under the Curve

BOLD Binary Online Learned Descriptor

BRIEF Binary Robust Independent Elementary Features

CNN Convolutional Neural Networks

CVG Computer Vision and Geometry

CVPR Conference on Vision and Pattern Recognition

DeepLSD Deep Line Segment Detector

DF Distance Field

DLD Deep Line Descriptor

DNIM Day-Night Image Matching

DoF Degrees-of-Freedom

DoG Difference of Gaussians

DoH Determinant of the Hessian

ELSD Efficient Line Segment Detector and Descriptor

ELSED Enhanced Line SEgment Drawing

FPR False Positive Rate

GIFT Group Invariant Feature Transform

GNN Graph Neural Network

GT Ground Truth

HAWP Holistically Attracted Wireframe Parsing

HDPL Hybrid Descriptor for Points and Lines

HF-Net Hierarchical Feature Network

HG HourGlass

HOG Histogram of Oriented Gradients

KP keypoints

L2D2 Learnable Line Detector and Descriptor

LBD Line Band Descriptor

LCNN Line Convolutional Neural Network

LE Localization Error

LiDAR Light Detection and Ranging

LIFT Learned Invariant Feature Transform

LISRD Local Invariance Selection at Runtime for Descriptors

LLD Learnable Line Descriptor

LMP Line Message Passing

LoFTR Local Feature Transformer

LoG Laplacian of Gaussian

LO-RANSAC Locally Optimized RANSAC

LSD Line Segment Detector

LSDNet Line Segment Detector Network

MLP Multi-Layer Perceptron

MS multi-scale

MS-COCO Microsoft Common Objects in COntext

MSER Maximally Stable Extremal Regions

MVS Multi-View Stereo

NLSPN Non-Local Spatial Propagation Network

NMS Non-Maximum Suppression

NN Nearest Neighbor

NW Needleman-Wunsch

NetVLAD Network for Vector of Locally Aggregated Descriptors

NYU New York University

ORB Oriented FAST and Rotated BRIEF

PhD Doctor of Philosophy

PE Positional Encoder

PL-Loc Point-Line Localization

R2D2 Repeatable and Reliable Detector and Descriptor

RANSAC Random Sample Consensus

ReLU Rectified Linear Unit

Rep Repeatability

ResNet Residual Network

RDNIM Rotated Day-Night Image Matching

RGB Red Green Blue

RGB-D Red Green Blue Depth

ROC Receiver Operating Characteristic

SfM Structure-from-Motion

SG SuperGlue

SIFT Scale-Invariant Feature Transform

SLAM Simultaneous Localization and Mapping

SOLD2 Self-supervised Occlusion-aware Line Description and Detection

SOTA State of the Art

SP SuperPoint

SURF Speeded Up Robust Features

SVD Singular Value Decomposition

TILDE Temporally Invariant Learned DEtector

TP-LSD Tri-Point Line Segment Detector

TPR True Positive Rate

YUD York Urban Dataset

VGG Vision and Geometry Group

VP Vanishing Point

VR Virtual Reality

WLD Wavelet Line Descriptor