Scan in Motion

Master Thesis
Golnoosh Samei
April 22, 2010

Advisors: Prof. Dr. Marc Pollefeys, Dr. Gabriel J. Brostow, Dr. Christopher Zach
Department of Computer Science, ETH Zürich
Abstract

In this thesis the problem of automatically building 3D models of moving vehicles seen from a stationary camera is investigated. In particular, we would like to recover the structure of the vehicle using its motion. We have sought to solve this problem in outdoor scenarios where there is complex background and uncontrolled environment. These settings bring about varying distance of the camera to the object, which itself results in a drastic change in the size of the object in different frames. To that end, we have used background subtraction along with relative camera poses to obtain a probabilistic visual hull of the object of interest. We have made extensions to the probabilistic visual hull to take into account the varying distance of the cameras from the object. The setting also suffers from the fact that the vehicle is not viewed from all directions. We strive to utilize the fact that the vehicles under consideration are symmetric to overcome this condition. Therefore propose a method for reliably computing the natural coordinate system of the car using structure from motion. We finally obtain the 3D model by symmetrizing the probabilistic visual hull and automatically extracting a suitable iso-surface. We conclude that the suggested pipeline results in 3D models which are very close to the actual shape of the vehicles.
Contents

List of Figures

1 Introduction 1
   1.1 General Overview .......................... 1
   1.2 Scanning objects in Motion .................... 2
   1.3 Outline of the Thesis ........................ 3

2 Related Work 5
   2.1 Introduction ................................ 5
   2.2 Reconstruction Methods ...................... 5
   2.3 Laboratory Scanning ........................ 6
   2.4 Outdoor Scenarios .......................... 7
       2.4.1 Model-based Approaches ................. 7

3 The System Framework 9
   3.1 Introduction ................................ 9
   3.2 Structure from Motion ....................... 10
       3.2.1 Finding Features ...................... 10
       3.2.2 Camera Pose Calculation ................. 12
   3.3 Background Modeling ....................... 12
       3.3.1 Learning Background Model ............... 13
       3.3.2 Finding the Probability of a Given Pixel 14
   3.4 Natural Coordinate System of the Car and Symmetrizing 14
       3.4.1 Error Criterion ........................ 15
   3.5 Probabilistic Visual Hull .................... 15

4 Probabilistic Visual Hull 17
   4.1 Introduction ............................. 17
## Contents

4.2 Probabilistic Framework .......................... 18
4.3 Sampling Variable or Mis-calibration Term .............. 19
  4.3.1 Probability Distribution of Projected Points ...... 21
  4.3.2 Covariance of the Camera Parameters .......... 22
  4.3.3 Jacobian of the Projection Function ........... 22
4.4 Discussion: Independence Assumption of Cameras ....... 24

5 Ackermann Property of Vehicle Motion ................ 25
  5.1 Introduction .................................. 25
  5.2 The Ackermann Steering Principle ................ 25
  5.3 Ackermann Principle’s Application to This Problem .. 26
  5.4 Finding the Natural Coordinate System of the Car .... 28
    5.4.1 The Vertical Component or the Rotation Axis .. 29
    5.4.2 Extracting the Heading Vector .............. 30

6 Experiments ..................................... 37
  6.1 Introduction .................................. 37
  6.2 Synthetic Data .................................. 37
  6.3 Outdoor Scenes .................................. 37
    6.3.1 Jimmy’s Pizzeria .......................... 37
    6.3.2 Sonnegrassese Sequence .................... 40

7 Conclusion and Future work .......................... 47
  7.1 Future Work .................................. 48

Bibliography ........................................ 49
List of Figures

3.1 Schematic Overview of the Pipeline .............................. 11
4.1 The probability distribution around the center of projection ... 20
4.2 The footprint of the projected voxel .............................. 20
4.3 The miscalibration probability distribution around the center of projection. 20
5.1 Vehicle Coordinate System ........................................ 26
5.2 Relation between the distance to the back axle L and the angle of the displacement vector $\varphi$ 27
5.3 Natural Coordinate System of A Car ............................... 29
5.4 Change of Steering Angle .......................................... 31
5.5 Fan of displacement vectors ...................................... 33
5.6 Histogram of displacement vectors ............................... 34
6.1 (a)The CAD BMW M3 model (b) The extracted 3D model .... 38
6.2 Images from the sequence Jimmy’s ............................... 39
6.3 Background probabilities for images in the Jimmy’s sequence .. 40
6.4 Visual hull and 3D model of sequence Jimmy’s .................. 41
6.5 The coordinate axes overlayed on the extracted iso-surface ...... 41
6.6 Visual hull and surrounding cameras for sequence Sonnegstrasse 42
6.7 Overlay of the extracted 3D model for the sequence Jimmy’s on different frames 42
6.8 The final 3D model for Jimmy’s sequence ........................ 43
6.9 Images from the sequence Sonnegstrasse ........................ 43
6.10 Visual hull and 3D model of Sonnegstrasse sequence .......... 44
6.11 Overlay of the Extracted 3D model of Sonnegstrasse sequence on the frames. 45
6.12 The 3D Model for Sonnegstrasse sequence from different viewing angles 45
Chapter 1

Introduction

1.1 General Overview

3D reconstruction or recovering the 3D shape of an object using images is one of the most active fields in computer vision. On the one hand, it is a very interesting mathematical problem, on the other hand it has many applications in today’s world. In this thesis we are focusing on a special case of 3D reconstruction. We are constraining ourselves to a single stationary camera. We would like to recover the structure of the object- in this case a vehicle- using its motion. For example, when an observer watches a vehicle turn on the street, he has a good knowledge about the 3D shape of the car. This is the case, because the car is viewed from many angles as it moves. This intuition has motivated us to search for a computer vision system that would recover the 3D shape of such a moving object. One of the areas in which this could be used if done automatically, is in detecting cars based on their 3D shape in surveillance cameras. Surveillance cameras are installed in different locations in the cities and it would be very helpful if they could be equipped with a software that could scan the 3D shape of the vehicles as they pass by. This information can then be used to detect the type of the vehicle such as transporters, passenger cars etc.

Another application would be reconstructing an event given enough information about the scene. This could be useful in sport events where the scene could be reconstructed in advance. Later during the game, the cars could be scanned as they drive and consequently the scene could be rendered from different views.

The advantage of using a single camera is that this method could be applied to already available videos. This also means that the system does not require complex and costly equipments which means that it could be used by everyone with a hand held camera and a tripod.
This method is especially interesting in cases where there is a video of a scene, the contents of which are no longer accessible. As an example, one might want to reconstruct a scene from an old film, where the cars are no longer existent, or very difficult to obtain.

This degree of freedom comes at a price, and makes the problem more challenging. Below we explain some of these challenges.

### 1.2 Scanning objects in Motion

Scanning objects to recover their 3D model is a well studied topic in computer vision. The applications are numerous and varied. Depending on the size and location of the object of interest, the methods vary.

For moderately small objects such as figurines which could be transported to laboratory environments, one could use controlled situations. In most cases the object of interest is placed on a turntable and photographed by one or more stationary cameras. In these cases, visual hull is usually only the first step and more elaborate methods will be used to recover details of the 3D model. Impressive results have been obtained for these settings and improving the state of the art seems challenging. In other cases where the object is moving such as humans the laboratory is equipped with multiple cameras with different viewing points. Hence, a 3D model could be calculated for each time slice. In most of these cases, the position and orientation of the cameras are known beforehand, making it easier to calibrate them. This is in stark contrast to the situation where there is only one camera and the relative position of the camera and the object must be calculated as the object moves. This also introduces noise on the calculations which requires a method that is robust to these noises.

In this thesis we would like to scan large objects that do not fit on a turn table. Moreover, we do not want to require laboratory conditions and would like to view different sides of the object during their motion. Consequently, we would like to use these different views and the knowledge of the relative position of the camera and the object at each instance to reconstruct the 3D shape. Nevertheless, this also means that we have to deal with complex backgrounds, suboptimal lighting conditions, secularities, etc.

This shows that our problem differs from many 3D reconstruction problems in that the calibration of the camera in each frame is not as accurate and is sometimes numerically very unstable. Also the complex backgrounds means we have difficulty in estimating the foreground mask. Another challenging aspect of this work is the fact that the distance of the cameras to the object varies significantly through the sequence. At its closest point the car might be 2 meters away from the camera and at is farthest 30 meters or
even more. This requires special attention to the effects of the distance on certainty and accuracy of projection of the model onto the images.

Furthermore, this causes another problem and that is the size of the object in each images is different. This could cause problems, since we have mixed pixels in the images the car is far away. This means that a larger portion of the volume projects to these pixels which could be only partially occupied with the object.

Also, one might not observe all sides of the object in this situation. Here, other cues -if they exist- might come to aid to predict the true shape of the object. For symmetric objects, for example, it seems enough to have images that observe only one side of the object.

We have looked at the problem of fixed camera and moving object as equivalent to the case of a fixed object and a rotating camera. The only difference is that in cases where the camera is moving, all the features points found in the image undergo a more or less same motion and therefore one can reliably perform structure from motion. However in our case one could only perform structure from motion on a small portion of the image and that is the region where the car is present.

1.3 Outline of the Thesis

The rest of the thesis is organized as follows: in chapter 3 we explain the general framework and different steps and explain each in detail. In chapter 4 we explain the probabilistic frame work for visual hull and the modifications we made in order to be robust towards varying distances of the cameras. In chapter 5 the Ackermann principle is explained and how it relates to our problem. We will find the natural coordinate system of the car and use it to enforce symmetry on the 3D model. Chapter 6 discusses the results obtained on 2 outdoor scenes. In chapter 7 conclusions are drawn and a number of ways in which the pipeline could improve are discussed as future work.
Chapter 2

Related Work

2.1 Introduction

Modeling the 3D shape of objects is one of the most active and well-studied fields in computer vision. Different approaches have existed for different purposes. In one end of the spectrum are methods that aim for computing highly accurate geometrical models of the object. On the other end of the spectrum are methods which aim only to render realistic images of the object from different viewing angles with little information about the underlying geometry.

In a more general sense, one could categorize these efforts to those dealing with outdoor scenarios and those which are pursuing their goal under laboratory conditions. In some of these works the camera is fixed and the object is moving and in the others the object is stationary and a camera is moving around the object. In the former the object is put on the center of a turntable and, therefore shows a different view while keeping the same distance to the camera at each frame.

2.2 Reconstruction Methods

One of the oldest methods is visual hull[4] or shape frame silhouettes. In the classic visual hull one intersects the silhouette cone of each image to compute the visual hull. The basic idea is that while we can’t be sure a point in 3D is inside the object if it projects to the interior of the silhouette, we could be sure that it is outside the object if it project outside the silhouette. This approach lies on perfect silhouettes which are typically obtained by placing the object in front of a simple background which produces high
2. Related Work

contrast. In outdoor scenarios this could not be achieved and the task of segmentation which is usually done by background subtraction fails in some regions. In these situations, if one follows the classic visual hull one might end up with a model that is smaller than the actual object. Franco and Boyer [6] suggested a probabilistic visual hull which deals with occupancy probabilities based on evidence from each image and could recover from imperfections of the segmentation task.

In their paper they have used this method to estimate the shape a person. This method has also been used by Guan et al [9] in a scenario where there are walking people in outdoor scenarios. However in this case there are fixed cameras and the distance to the cameras remains in the same range for all the sequences. The appearance model of the foreground objects -people- are distinctive from the background and is not specular. In our pipeline we have based our method on this Bayesian approach.

2.3 Laboratory Scanning

In frameworks with laboratory conditions, often small objects are studied. In these situations, the background is as simple as possible and chosen in a way that would give as much contrast with the object as possible, hence making the task of segmentation simple. In [10], Matusik et al have described a setting where the object of interest is placed on a turntable with an array of cameras on a vertical column capturing images from the shape as it turns. They acquire alpha mattes of the object from multiple viewpoints which are used in what they define as the opacity hull. The opacity hull is the visual hull of the object with view-dependent opacity. In most of these works visual hull is only used as a first step and further refinements are performed such as multifidi stereo method which use the photoconsistency criterion[18]. Others try to use both silhouette and photoconsistency cues to obtain a smooth and accurate surface [1][7]. These methods are especially useful to extract concavities of the object where the visual hull fails. The use of these methods remain unfeasible in our problem as they are sensitive to highly specular and shiny surfaces in ill-conditioned lightings.

Other efforts in laboratory conditions are those which do not use the turntable instead they rely on a structured light to obtain the depth field of the object [5]. These efforts fall into shape from shading category and are out of the scope of this thesis.
2.4 Outdoor Scenarios

Reconstructing objects and buildings in outdoor scenarios is often performed with different methods. The first step to almost all of the methods is structure from motion to calibrate the cameras and estimate their poses. One of the most important works in this area is that of Pollefeys et al.\cite{14} where using a hand held camera the 3D structure of a building or a facade is reconstructed. In these methods, the first step is to find reliable features on the scene and then to track these features. Consequently using structure from motion methods, the position of the cameras is calculated. At this point one has a sparse 3D point cloud of the tracked features. These points are not enough to construct the 3D model, therefore other methods are applied to come up with a dense reconstruction.

A series of papers from van den Hengel et al have tried to solve the problem of 3D reconstruction in a trade-off between automatic reconstruction and user provided information\cite{16, 17, 15}. In their framework they have a stationary object and moving camera. They apply structure and motion to track the cameras and then have the user specify points on the object of interest, choosing between a set of simple geometry models and the sparse 3D point cloud they would find the actual 3D model of the object. In \cite{15} they have developed a graphic interface, where the user could interactively construct the 3D shape of more complex objects such as a car. In this framework the user specifies a plane in 3D drawing lines and points on the surface of object. The user would later be able to adjust the surface by viewing it from different viewing angles in the video. This method gives good quality 3D mesh at the price of manual intervention of the user. Moreover, this approach is only useful if you have a video of a stationary object. However in the case of a moving object with a fixed camera, this wouldn not work as the overwhelming number of stationary feature points would make it impossible to correctly calculate the camera poses.

There are few examples of approaches where the camera is not moving. One of these papers which deals with surveillance cameras is \cite{8} where the images are frames of a street scene with vehicular traffic recorded by a stationary camera. The final geometry is an overly simple one which is the convex hull of the 3D points as an approximation to a volumetric model of the moving object.

2.4.1 Model-based Approaches

Probably the closest work in spirit to the subject of this thesis is that of Leotta and Mundy\cite{11}, which tries to find the 3D model of a car given a single image. However their method differs from ours in that they assume
a deformable model and also use a training set of CAD 3D models of cars. They learn their training set by fitting each CAD model to the deformable model. They use PCA to reduce the dimension of the data-space. Given a new image, they try to fit the deformable model based on the learnt models. They have heavy model assumptions and require not only the silhouette but also the interior lines of a car extracted using an edge detection method. Our method also differs in that we require no user intervention.
Chapter 3

The System Framework

3.1 Introduction

In this chapter we look at the framework of the project. In order to obtain a visual hull which looks close to the original model of the car, we had to go through different step. To put all these steps together, we developed a pipeline.

In the following sections different steps are explained in detail. The first step would be to acquire image frames from the video stream. Next we will run the 2d tracker on the data and save the tracking information for each frame. Consequently we use structure from motion to find the camera poses. We use the Urbanscape pipeline developed in UNC university to model urban environments[3]. However since there are an overwhelming number of stationary 2d features (on points of the scene other than the moving object), the structure from motion process fails. Therefore, we have added an intermediary step to filter out the feature points lying outside of the moving object. Having filtered out the stationary 2d feature points, we will write them again to the files with the same format as the Urbanscape pipeline and have Urbanscape read them as if it had generated them and do the structure from motion part. The final step would be to construct the probabilistic visual hull. Hence we need background probabilities for the pixels. Having the background probabilities we then perform the next step, which is the probabilistic visual hull. Finally we notice that since we never see the car from all views, the probabilistic visual hull has a bias towards regions that were never seen. Therefore we use the symmetry of the cars and put the same set of cameras mirrored with respect to the axis symmetry of the car. To that end, we will turn our attention to the trajectory of the car and will try to find the natural coordinate system of the car. Having found the plane of symmetry of the car we will be able to use the symmetry cue
to construct a visual hull that is symmetric and hence, more accurate. A schematic overview of the pipeline can be seen in figure 3.1.

3.2 Structure from Motion

One of the most important parts in the pipeline is the structure from motion part. In order to find an accurate visual hull, we must be sure to have accurate camera poses. To that end we must provide the structure from motion system with reliable features tracks. One of the problems in our setting is the relatively small portion of the image that occupies the object of interest. As the vehicle moves in the street its image might get smaller or larger according to its distance from the camera. In best cases it occupies only 5 percent of the image and in worst cases when it is far it could only occupy 1/10 of this amount. In these situations it is difficult to achieve accurate poses and this problem can be numerically very unstable. Therefore it is important to apply the triangulation on the features with the lowest possible number of outliers while keeping as much inliers as possible. Below we describe different steps in finding reliable camera poses.

One of the places where this problem becomes serious is in the initialization steps. The Urbanscape pipeline that we run, initializes the camera tracker using the relative pose of the the first three frames in which there is enough motion using five point algorithm [12][13], consequently for the next frames the poses are computed using RANSAC and hypothesis-generation based on 2D to 3D constraints of the features[13].

This shows how sensitive the pipeline is to initialization. Of course this is not a source of problems in the context the pipeline is used because all the frames are more or less equally conditioned. However, in the setting of a moving vehicle, the conditioning of the features could vary greatly. It was observed that when the vehicle is in a distance and therefore they span a very small range of angles with respect to the center of camera, and also when the motion is towards the camera - no or small sideways motion, the pipeline is initialized incorrectly. Hence we start the pipeline not in the beginning of the sequence, but in the middle of the sequence depending on when the car is closet an has considerable sideways motion.

Below we describe different steps in finding the camera poses.

3.2.1 Finding Features

In the first step we find the feature points in the image and also track them through the whole sequence. These features might or might not lie on the
Figure 3.1: Schematic Overview of the Pipeline
object of interest. However we can use binary masks -labeling foreground and background pixels- produced in other steps to filter out the features outside of the foreground.

**Filtering out the Irrelevant Features**

In this step, we need to omit the features that do not lie on the car. To this end, we can use a number of methods. One could use a mask that defines which parts of the scene belong to the foreground object. Another way would be to focus on the displacement of each features point and assuming that the moving object is the only moving element of the scene, one could set a threshold to omit the non-relevant feature points. The latter has the risk of losing all the feature points once the moving object reaches a halt for a short period of time. Also if there is some sort of motion in the background, such as leaves moving, it would be hard to identify this motion and omit it. The former has the advantage that it is more accurate and that we are sure to have included all the features on the object of interest and omitted all the features points outside of it no matter how much they move. This is especially robust if we have obtained the masks using other cues than motion, such as background subtraction. In the next section we will explain the procedure for finding the background subtraction masks. But it is important to note that we can overcome problems with the leaves using this method.

This is not so strict and other moving objects might be included as long as the outliers are few enough not to interfere with the structure from motion process.

### 3.2.2 Camera Pose Calculation

Next, we run the structure from motion code designed for Urban Reconstruction [3]. In this pipeline, it is assumed that the camera is moving and the object is being fixed. Therefore the camera extrinsic parameters obtained through this pipeline are based on these assumptions. This means we get different positions for the camera in each frame. Therefore, we can treat the problem as one of having a fixed object with cameras placed around it. The output of this step is fed to visual hull code and also the Ackermann analysis code.

### 3.3 Background Modeling

In this chapter we’ll explain how we model the background. As mentioned in the introduction, in order to be able to find a good visual hull, we need
3.3. Background Modeling

a probabilistic segmentation. That is to say, we need to find out for each pixel, with what probability it belongs to the background. Hence, we need to know what the background "looks like" or more accurately, we need the background model. Once we have the background model, we will be able to tell how probable it is for a pixel to lie on the background given its color. Therefore, we need two steps: First learning the background color, Secondly: predicting the probability of a given pixel. To perform the operations in this part the code and method of Christoper Zach [19]is used.

3.3.1 Learning Background Model

Preprocessing

The first step of background subtraction is learning the background. However, in our sequences we do not have explicit frames where there is no foreground objects. However, we do have the advantage of having a video and hence a lot of frames. Therefore if we miss some part of the background in a sequence of frames we can make up for that by collecting information about that part in other frames where that particular part of the background is visible. One the one hand, we don’t know which pixels belong to a foreground object or we had our problem solved in the first place. On the other hand, we don’t really need a precise segmentation of the foreground in this step, we only need to make sure we do no include any foreground at the price of excluding many background pixels. This will lead to no problems, since we can recover those pixels in other frames where there is no foreground object in the neighborhood.

This coarse segmentation is done utilizing optical flow. First we apply optical flow on the sequence of frames. Then we choose an appropriate threshold - here I’m using Utsu’s threshold on grayscale images- in order to identify foreground. Next we apply some morphological operations to erase the small regions with some motion such as pedestrians. Now we have our masks to learn the background model.

Learning Step

In this step, for each pixel position, information from all frames in which at this pixel the mask is zero are collected and then a mixture of gaussian model is fitted to this data. In the end for each pixel we have a mixture of Gaussian and given a new value for this pixel, we could calculate the probability of this new value belonging to a foreground object or background.
3.3.2 Finding the Probability of a Given Pixel

Having found the background model, we can now run the code to obtain background probability for each pixel. The outputs of this code also include a binary mask which is obtained from the probabilistic images using graph-cuts. This is especially interesting because we could also use this for the purpose of filtering the 2d features.

In spite of the meaning behind probabilistic visual hull, which needs probabilities of belonging to the background for each pixel, we opted to use binary masks. The reason is that the probabilities for many pixels are not independent and therefore, regions such as windows or the lower part of the surface of the car reflect or refract light and therefore resemble the background in many frames. This will lead to empty voxels for the windows and many regions of the car. Here again we incorporate the prior on the car, and that is it does not have holes inside. Which means if there is a pixel surrounded by high probability background pixels, this pixel is also part of the foreground because the foreground object does not have holes.

3.4 Natural Coordinate System of the Car and Symmetrizing

All the measurements including camera poses and the 3D point cloud are in an arbitrary coordinate system given by the urbanscape pipeline. However, we would like to compute the probabilistic visual hull and also perform the surface extraction in the natural coordinate system of the car to avoid sampling errors. This is also crucial to our task of symmetrizing the car which needs the axis of symmetry. In chapter 5, it will be explained how one finds the natural coordinate system of the car. However, what we end up with, is only correct up to an offset, meaning the origin is not specified. Due to reasons that will become clear in chapter 5, the origin of this coordinate system should be on the back axel of the car. However, we have no information about this before performing the visual hull. Therefore we transform all the cameras to some coordinate system which is parallel to the coordinate system of the car. We then perform the probabilistic visual hull which will be explained in summary in section 3.5 and in more detail in chapter 4. Because of asymmetric distribution of the cameras around the car, the visual hull is biased and also not symmetric. This is because the voxels that are on the side of the car that is not facing the cameras project within the silhouettes in most of the frames. Since we know that the car is symmetric around its x-axis we would like to use this cue. What we do at this point is to put the same set of cameras mirrors around the symmetry axis of the car. However we know this axis up to an offset. What we do at this point is to have an
initial guess which is the center of gravity of the asymmetric visual hull. We then use this point to compute a new visual hull using 2 set of mirrored cameras. In order to find out how correct we have been in choosing this offset we backproject the extracted surface of the probabilistic visual hull into the images to check the silhouette consistency of the volume. If we have chosen an offset too close to the original set of cameras the resulting volume would be smaller than what it should be and if the offset if further from the cameras the resulting volume would be larger. For each image we back project the volume at a different threshold and record a given error criterion. Since this error criterion has a global minimum on the correct center of rotation, we would be able to find this by iterating the process.

### 3.4.1 Error Criterion

The error criterion could be defined in many ways. One could take into account the interior volume of the final surface and try to keep it the same while symmetrizing the car. One could also try to stay within the silhouette-consistency criterion. While we symmetrize the car we wouldn’t want to increase the probability of voxels that were given low probabilities in the image.

On the plane of symmetry we could assume the same path of cameras on the other side of the car. Therefore the probability for each voxel would be calculated once again using that information. We could then shift the offset of the symmetry plane to find the most suitable point that matches the silhouettes.

### 3.5 Probabilistic Visual Hull

After having the camera poses and the background probabilities we can now run the probabilistic visual hull.

We use the camera matrices derived from the structure from motion part. We also need to know a bounding box around the object in 3D world coordinates. To this end we turn to the 3D point cloud and since there are always outliers present we exclude the 10 percent of the points that are furthest from the mean of the points. The distance is of course a Mahalanobis distance. Therefore we first calculate the covariance of the points and and the distance to the mean is calculated accordingly.

Now we find the bounding box as the axis aligned box that would contain all the points.
3. The System Framework

The details of the probabilistic visual hull method are explained in the chapter 4.
Chapter 4

Probabilistic Visual Hull

4.1 Introduction

Given a set of cameras and their corresponding binary mask of foreground, one could extract the visual hull of the foreground object. This is, however, very unforgiving towards faulty masks. For example, if in only one of the masks, a part of the foreground object is not detected, the space carving algorithm would carve all the corresponding parts in the visual hull. Franco and Boyer[6] proposed a probabilistic framework to calculate probabilities in a space occupancy grid. Their method is able to deal with the aforementioned situations. In this project we have based our visual hull algorithm on this probabilistic framework.

The settings of our problem are different with problems in which visual hull is used. In laboratory situations one usually has a number of fixed cameras which are uniformly distributed around the object. Furthermore the distance between the object and each camera is approximately constant. These two conditions are not present in our problem. Namely, we have a highly non-uniform distribution of viewpoints. When the car is closest to the camera - and this is usually when the car is turning - the projected motion of the car is faster and this leads to fewer frames from that viewing angle. In contrast when the car is driving down the road or moving away from the cameras, one basically observes the same viewing point in a large number of frames. Therefore, when we arrange the virtual cameras around the object we are faced with a situation in which too many cameras in the same viewing angle are present and too few cameras in another viewing angle. This might not seem like a severe problem in the original visual hull, where only one camera is enough to carve away areas that are seen empty. However, in the probabilistic framework, this does pose a problem. This issue is addressed in section 4.4.
The other problem is varying distance of the cameras. When arranging the cameras on a circle, the projection of each voxel on the image plane of different cameras would have the same area. Therefore, in most applications, based on the settings, one chooses the radius of this projection and uses the same projection footprint for all the cameras. However, in our settings, this would cause serious problems. First of all fixing a projection window would exclude many of the pixels viewed from a close camera, and include many irrelevant pixels in the image plane of a far camera. The other problem is the certainty about the projection of the 3D points. It seems unrealistic that a point project to a far away camera as accurately as it would on a nearby camera. Furthermore, in cases where we don’t have perfect calibration of cameras—which is not so much unlikely in our case—the mis-calibration has different effects depending on the relative position of the camera and the voxel. These issues are addressed in section 4.3.

In section 4.2, we first give an overview of the probabilistic framework and different terms. In the following sections we make the connection between the raised issue and how we can solve them within the probabilistic framework.

4.2 Probabilistic Framework

In the framework suggested by Franco and Boyer, a Bayesian approach is taken in the sense that instead of focusing on the effect of the images on the visual hull, one goes in the other direction and defines the likelihood of observing the images given different states of the volume. An occupancy grid is defined on the volume of interest and each element of this grid is called a voxel; Therefore the Bayesian framework models the effect of voxels in the visual hull on the images.

If we denote the volume for which we would like to calculate the vacancy probabilities by $G$ and the set of images by $I_1 \ldots I_n$ we will have:

$$p(G|I_1, I_2, \ldots I_n) = \frac{p(I_1, I_2, \ldots I_n|G)p(G)}{p(I_1, I_2, \ldots I_n)}$$

$$p(G|I_1, I_2, \ldots I_n) = \frac{p(I_1, I_2, \ldots I_n|G = 1)p(G = 1)}{p(I_1, I_2, \ldots I_n|G = 1)p(G = 1) + p(I_1, I_2, \ldots I_n|G = 0)p(G = 0)}$$

under the assumption that the images are independent we will have:

$$p(G|I_1, I_2, \ldots I_n) = \frac{\prod_{i=1}^{n} p(I_i|G)}{\prod_{i=1}^{n} p(I_i)}$$
In order to be able to relate the vacancy of a voxel to the color we observe at a particular pixel, a latent variable called silhouette detection variable $F$ is introduced which will be 1 if the pixel belongs to the foreground object and is 0 otherwise. Now the probability will be as follows:

$$p(I | G) = p(I | G, F = 1)p(F = 1 | G) + p(I | G, F = 0)p(F = 0 | G)$$

The term $p(I | G, F = 0)$ is the probability distribution of the colors knowing that the pixels are on the background. This term can be calculated using the background modeling method explained in the section 3.3. The term $p(I | G, F = 1)$ is the probability distribution of the colors knowing that the pixels are on the foreground. To calculate $p(F = 1 | G)$ one has to consider two cases. Below we explain these cases:

$p(F = 1 | G = 1)$: Probability that a pixel is on the foreground given the fact that the projecting voxel is full. Here again another variable called the sampling variable $S$ is introduced which measures the probability that the given voxel is indeed in the viewing line of the given pixel. When the voxel is not in the viewing line of the pixel, the information about the pixel gives us no information about the vacancy of the voxel and hence the probability of the voxel is uniform and independent of the pixel ($u$). However, when the voxel is in the viewing line of the voxel the probability that the silhouette term is one is determined by the detection probability of the sensors ($P_{detection}$).

$$p(F = 1 | G = 1) = p(S = 0)u + p(S = 1)P_{detection}$$

$p(F = 1 | G = 1)$: Probability that a pixel is on the background given the fact that the projecting voxel is empty. Another variable introduced here is the external detection cause variable $R$ and the False Alarm Rate $P_{FAR}$

$$p(F = 0 | G = 0)p(S = 0)u + p(S = 1)[p(R = 1)P_{detection} + p(R = 0)P_{FAR}]$$

### 4.3 Sampling Variable or Mis-calibration Term

In the previous section we have given an introductory summary on the framework. The sampling variable is in fact the probability that the voxel is projected correctly onto the given pixel. Even though this variable is assumed constant in the Boyer and Franco paper, we have defined it for each pair of pixel and voxel depending on their relative position.

The probability that a pixel and a voxel are on the same viewing line depends on two factors, firstly the 3D point-voxel- to be projected might not
project exactly to the pixel given by the projection matrix due to noise on the calibration process. Given the noise on the calibration parameters, we derive the probability distribution of the projected point. Therefore for each pixel in the image plane we will be able to calculate the probability that it is in the viewing line of the voxel.

However, in case of a projection footprint of more than one pixel, there are a window of pixels each with such a probability distribution around them, therefore for a pixel, the probability that its neighboring pixels were the center of projection should also add to its probability. In effect, one should convolve the footprint with the probability distribution at each point. This is illustrated in figures: 4.1, 4.2 and 4.3.

In the section 4.3.1 we derive a relationship between the probability distri-
bution of projected points and the probability distribution of the noise on camera parameters. In section 4.3.2 we will look at the covariance matrix of the camera parameters. In section 4.3.3, we derive relationships for deriving the Jacobian of the projection function.

### 4.3.1 Probability Distribution of Projected Points

If we assume that the noise on camera’s extrinsic parameters is Gaussian, and also that the projection function is linear around the point \( P \) in camera parameters’ space, the problem would be reduced to finding the covariance matrix of the projected points given the covariance matrix of the extrinsic parameters.

A projection is a function:

\[
f : \mathbb{R}^{3N} \times \mathbb{R}^P \rightarrow \mathbb{R}^{2N}
\]

(4.1)

Which is to say \( f \) takes \( N \) points in \( \mathbb{R}^{3N} \) and a \( P \)-vector of camera parameters in \( \mathbb{R}^P \) and outputs \( N \) points in \( \mathbb{R}^2 \). We assume that the points in \( \mathbb{R}^3 \) are accurate and therefore, the uncertainty stems from the uncertainty of parameters in \( \mathbb{R}^P \). For a given set of \( N \) points in \( \mathbb{R}^3 \) called \( X \), if we assume that the function \( f \) is essentially linear around a point in \( \mathbb{R}^P \), then we could model \( f \) by its Jacobian around that point, we have:

\[
f(X, v) \approx f(X, \bar{v}) + J(v - \bar{v})
\]

(4.2)

For a given a set of \( N \) points in \( \mathbb{R}^3 \), we would like to know the effect of perturbation in camera parameters on the projected points in \( \mathbb{R}^2 \). Therefore we define the function

\[
g : \mathbb{R}^P \rightarrow \mathbb{R}^{2N}
\]

\[
g(v) = f(\hat{X}, v)
\]

we could derive the covariance of the \( g \) function around the point \( \bar{v} \) using the following relation:

\[
cov(g(v)) = J_g \Sigma J_g^T
\]

(4.3)

where \( \Sigma \) is the covariance of the variable \( v \).
4.3.2 Covariance of the Camera Parameters

Covariance of the camera parameters could be derived as a by-product of bundle-adjustment. This would give us a large covariance matrix with all the parameters of the camera and all the 3D points that were used to find the poses from which we should be able to find the covariance matrix of the camera parameters.

Having different covariance matrices for the cameras, would result in down-weighting the effect of those cameras that are weakly linked to the set of cameras and hence we would have a more accurate reconstruction. In this work, however, we have assumed constant noise on the position of the camera.

4.3.3 Jacobian of the Projection Function

We define the \( \Pi \) function as the projection function:

\[
\Pi(m, \theta) = \frac{1}{z(m, \theta)} \begin{bmatrix} x(m, \theta) \\ y(m, \theta) \end{bmatrix}
\]  

(4.4)

\[
\frac{\partial \Pi(m, \theta)}{\partial \theta_i} = \frac{1}{z^2(m, \theta)} \begin{bmatrix} z(m, \theta) \frac{\partial x(m, \theta)}{\partial \theta_i} - x(m, \theta) \frac{\partial z(m, \theta)}{\partial \theta_i} \\ z(m, \theta) \frac{\partial y(m, \theta)}{\partial \theta_i} - y(m, \theta) \frac{\partial z(m, \theta)}{\partial \theta_i} \end{bmatrix}
\]  

(4.5)

So, the Jacobian will be the following:

\[
J(\Pi(m, \theta)) = \begin{bmatrix} \frac{\partial \Pi(m, \theta)}{\partial \theta_1} & \ldots & \frac{\partial \Pi(m, \theta)}{\partial \theta_6} \end{bmatrix}
\]  

(4.6)

Let the affine transformation that takes a point from real world coordinates to the coordinate system of the camera be defined as \( A(m, \theta) \) with \( m \) the position of the 3D point and \( \theta \) the vector representing the parameters (rotation and position of the center of the camera):

\[
A(m, \theta) = \begin{bmatrix} x(m, \theta) \\ y(m, \theta) \\ z(m, \theta) \end{bmatrix} = R(\theta_{1:3})(m - c(\theta_{4:6}))
\]  

(4.7)

where \( R(\xi) \) is the rotation matrix and \( c(\xi) \) is the center of the camera. To ease the derivation process we use the exponential mapping for the rotation matrix. We have:
The partial derivative of $A(m, \theta)$ with respect to $\theta_i$ $1 \leq i \leq 3$ would be:

$$\frac{\partial A(m, \theta)}{\partial \theta_i} = \begin{bmatrix} \frac{\partial x(m, \theta)}{\partial \theta_i} \\ \frac{\partial y(m, \theta)}{\partial \theta_i} \\ \frac{\partial z(m, \theta)}{\partial \theta_i} \end{bmatrix} = R(\theta_{1:3}) \frac{\partial \hat{\theta}}{\partial \theta_i} (m - c(\theta_{4:6})) \quad (4.8)$$

so for example for $i = 1$ we have:

$$\frac{\partial A(m, \theta)}{\partial \theta_1} = \begin{bmatrix} \frac{\partial x(m, \theta)}{\partial \theta_1} \\ \frac{\partial y(m, \theta)}{\partial \theta_1} \\ \frac{\partial z(m, \theta)}{\partial \theta_1} \end{bmatrix} = R(\theta_{1:3}) \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} (m - c(\theta_{4:6})) \quad (4.9)$$

For $4 \leq i \leq 6$ the above partial differential equation would be:

$$\frac{\partial A(m, \theta)}{\partial \theta_i} = \begin{bmatrix} \frac{\partial x(m, \theta)}{\partial \theta_i} \\ \frac{\partial y(m, \theta)}{\partial \theta_i} \\ \frac{\partial z(m, \theta)}{\partial \theta_i} \end{bmatrix} = -R(\theta_{1:3}) \frac{\partial c(\theta_{4:6})}{\partial \theta_i} \quad (4.10)$$

And, again for $i = 4$ we have:

$$\frac{\partial A(m, \theta)}{\partial \theta_4} = -R(\theta_{1:3}) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (4.11)$$

To get the Jacobian matrix one should calculate each of the partial derivatives.

$$c(\theta_{4:6}) = \begin{bmatrix} \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} \quad (4.12)$$

$$R(\theta) = e^{\hat{\theta}} \quad (4.13)$$

where

$$\hat{\theta} = \begin{bmatrix} 0 & -\theta_3 & \theta_2 \\ \theta_3 & 0 & -\theta_1 \\ -\theta_2 & \theta_1 & 0 \end{bmatrix}$$
4.4 Discussion: Independence Assumption of Cameras

In the opening paragraph of this section we explained how the problem of finding the likelihoods will be simplified by assuming the image of different cameras are independent. However, if one looks closely, will realize that images of two nearby cameras could be highly correlated. Therefore, it looks unreasonable to assume every two image are independent. To make the matter more clear, let’s imagine the extreme case where we have two images from the same viewpoint. In this case If we write the Bayes rule we will have

\[
p(G|I_1, I_2) = \frac{p(I_1, I_2|G)P(G)}{P(I_1, I_2)}
\]

In the case of independence we will have:

\[
p(G|I_1, I_2) = \frac{p(I_1|G)p(I_2|G)P(G)}{P(I_1)P(I_2)}
\]

Since \( I_1 \) and \( I_2 \) are essentially the same image taken from the same point, defining \( \rho = \frac{p(I_1|G)}{P(I_1)} \) we can write:

\[
p(G|I_1, I_2) = \rho^2 P(G)
\]

having \( n \) images we would have :

\[
p(G|I_1, \ldots, I_n) = \rho^n P(G)
\]

which is obviously a contradiction because having the same image \( n \) times shouldn’t make the final probability different. Therefore we suggest writing the full expression taking into account the dependencies between the images.

\[
p(G|I_1, I_2) = \frac{p(I_1|I_2, G)p(I_2|G)P(G)}{P(I_1|I_2)P(I_2)}
\]

However, it remains a future work to model these dependencies between these cameras. Nevertheless, it must be noted that the effect of this dependence becomes more clear when there is non-uniform distribution of the cameras around the object. Therefore, in order to mitigate this effect, we have used only a fraction of the total number of cameras and tried to chose the same number of cameras in a given viewing angle.
Chapter 5

Ackermann Property of Vehicle Motion

5.1 Introduction

Since we are modeling a car, we could take advantage of the properties of vehicular motions. Ackermann principle is one example of such properties. In the following we explain the Ackermann’s principle and later we explain how we can apply it to our problem to improve the results. The Ackermann principle simply puts constrains on the trajectory of a car.

5.2 The Ackermann Steering Principle

The Ackermann steering principle is the geometry rules that are observed in designing the vehicles in order to achieve good steering performance. This means that almost all vehicles are designed such that at each moment of time their motion can be modeled as a rotation such that their wheels move on concentric circles with different radii. This rotation is around a certain point in the 2d plane of its movement. This point lies on the axle of back wheels (Figure 5.1)

Even when the car is moving on a straight line it could be assumed that the center of rotation is at infinity. If we take a close look at the motion of the car we can derive a relation between the distance of each point from the back axle which we denote as L and the degree of the displacement vector \( \varphi \).

we have:

\[
\sin(\varphi - \theta/2) = L/r;
\]
Figure 5.1: Vehicle Coordinate System

\[ \sin(\theta/2) = \frac{\lambda/2}{r} \]

Therefore we have:

\[ \sin(\varphi - \theta/2) = \frac{2L}{\lambda} \sin(\theta/2) \]

This result has been derived using a different method by Scaramuzza et al in [2].

### 5.3 Ackermann Principle’s Application to This Problem

Let’s assume that we have \( F \) frames and that we have the projection matrix for each of these frames:

\[ P_1 \ldots P_F \]

If we decompose these projection matrices we have:

\[ P = [R] - R \cdot c \]
5.3. Ackermann Principle’s Application to This Problem

Figure 5.2: Relation between the distance to the back axle L and the angle of the displacement vector $\varphi$

where $c$ is the center of the camera and $R$ is the rotation matrix which needs to be applied to points in world coordinate system. In order to find the trajectory of the car, let’s look at the general relation between the car motion and the camera motion:

$$u = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} R_c^T & -R_c^T c \end{bmatrix} \begin{bmatrix} R_v & t_v \end{bmatrix}$$

where $R_c$ is the matrix related to the rotation of the camera, $c$ is the center of the camera, $R_v$ is the matrix related to the rotation of the vehicle and $t_v$ is the displacement of the vehicle. Now if we have the same measurements in the image, that means $u_1 = u_2$, then we have:

$$\begin{bmatrix} R_{c_1}^T & -R_{c_1}^T c_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{v_1} \\ t_{v_1} \end{bmatrix} = \begin{bmatrix} R_{c_2}^T & -R_{c_2}^T c_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_{v_2} \\ t_{v_2} \end{bmatrix}$$

Which means that the same projection matrix could be attributed to the motion of the car, with the same effects being observed. If the position of the car in the world coordinate system at the $n^{th}$ time instant (second frame) is denoted by $X_n$, then we can write:
5. Ackermann Property of Vehicle Motion

\[
X_i = \begin{bmatrix} R_{v_i} & t_{v_i} \end{bmatrix} \begin{bmatrix} X_0 \\ 1 \end{bmatrix} = R_{v_i}X_0 + t_{v_i}
\]

\[
X_{i+1} = \begin{bmatrix} R_{v_{i+1}} & t_{v_{i+1}} \end{bmatrix} \begin{bmatrix} X_0 \\ 1 \end{bmatrix} = R_{v_{i+1}}X_0 + t_{v_{i+1}}
\]

\[
\Delta X_{i+1} = X_{i+1} - X_i = (R_{v_{i+1}} - R_{v_i})X_0 + t_{v_{i+1}} - t_{v_i}
\]

Also for the heading of the car, if we denote the heading vector in time instance \( n \) by \( h_n \) we’ll have:

\[
h_i = R_{v_i}h_0
\]

Then we’ll have:

\[
\text{angle}(\Delta X_{i+1}, h_i) = \text{angle}(h_{i+1}, h_i)/2
\]

\[
\text{angle}((R_{v_{i+1}} - R_{v_i})X_0 + t_{v_{i+1}} - t_{v_i}, R_{v_i}h_0) = (\text{angle}(h_{i+1}, h_i))/2
\]

In order to apply the Ackermann principle, the rotation should be defined as the rotation of the car from one frame to the other. However, the rotation matrices we have denote the rotation of the camera relative to the first frame. In order to extract relative rotations we have:

\[
R_{i:i+j} = R_{i+j}R_i^T
\]

if \( j = 1 \) then this would be the rotation between two consecutive frames, however, since the degree of rotation between two frames is typically very small we use a number between 4 to 7 as \( j \).

5.4 Finding the Natural Coordinate System of the Car

In this section we explain how we find the natural coordinate system of the car. The coordinate system in which we are given the rotation matrices and translation vectors is some arbitrary coordinate system given by the structure from motion pipeline. The natural coordinate system of the car is a coordinate system where the \( x \) axis points towards the orientation of the car, the \( y \) axis is normal to the plane of movement of the car and is hence the axis of rotation for all the rotations the vehicle undergoes. The \( z \) axis is the axis which is parallel to the axis of wheels. Below we first extract the axis of rotation of the car and then move on and propose 3 methods for calculating the \( x \) axis of the car which we call the heading of the car(Figure 5.3).
5.4. Finding the Natural Coordinate System of the Car

5.4.1 The Vertical Component or the Rotation Axis

The $y$-axis of the car could be extracted accurately since this axis is also the axis of rotation of the car. This is the case, because we have the rotation matrix for every frame -or every relative motion. In principle we would be able to find this given only one rotation matrix, however because of noise this estimation wouldn’t be reliable. We, therefore, use all of our rotation matrices to extract the rotation axis robustly.

We know that the axis of rotation is the vector which will remain the same when it undergoes the corresponding rotation. In other words if we denote the rotation axis as $\mathbf{v}$ and the rotation matrix as $R$ we have:

$$R\mathbf{v} = \mathbf{v}$$

Which would be written in eigenvalue formulation as follows:

$$(R - I)\mathbf{v} = 0$$
5. Ackermann Property of Vehicle Motion

That is we will have to find the first eigenvector of this rotation matrix. or more robustly we can stack up all the \( R_i \) matrices and seek the solution to the least square problem:

\[
v = \arg \min_v \left\| \begin{bmatrix} R_1 - I \\ \vdots \\ R_n - I \end{bmatrix} \vec{v} \right\|^2
\]

In Matlab we could easily perform the following:

```matlab
R_all = [];  
for i = 1:length(R)  
    R_all = [R_all; R{i} - eye(3)];  
end  
[s v m] = svd(R_all)
```

5.4.2 Extracting the Heading Vector

Using the center of rotation

The following method has the advantage that it doesn’t depend on the point cloud for its calculations. However it is not in practice suitable because of being highly sensitive to noise. As shown in the figure 5.4 if in the first frame the vehicle rotates around the point \( \mathbf{c}_1 \) and in the second frame the vehicle rotates around the point \( \mathbf{c}_2 \), we are able to find the axis of the wheels by connecting these two points. However we only know the rotation matrices for rotation between the first frame and the \( i \)th frame not between two consecutive frames namely \( i \)th and \( i+1 \)th frames. Therefore we need to find the center of rotation for such a rotation. we have :

if we write it in terms of \( t \) the displacement, we have :

\[
\begin{align*}
X_i &= R_i X_0 + t_i \\
X_{i+1} &= R_{i+1} X_0 + t_{i+1} \\
X_0 &= R_i^T(X_i - t_i) \\
X_{i+1} &= R_{i+1}(R_i^T(X_i - t_i)) + t_{i+1} \\
X_{i+1} &= R_{i+1}R_i^T X_i + t_{i+1} - R_{i+1}R_i^T t_i \\
\end{align*}
\]

That is :

\[
X_{i+1} = R_{i:i+1} X_i + t_{i:i+1} \tag{5.2}
\]
where $R_{i:i+1} = R_{i+1}R_i^T$ and $t_{i:i+1} = t_{i+1} - R_{i:i+1}t_i$.

Now the Ackermann principle states that each segment of the motion is a rotation around some point. If we assume that the center of rotation in $i$th frame is $c_i$, for a point $X_i$ in frame $i$ we can write:

$$X_{i+1} = R_{i:i+1}(X_i - c_i) + c_i$$

we need to compare this to equation 5.2 we can solve for the center $c_i$.

$$c_i = (I - R_{i:i+1})^{-1}t_{i:i+1} \quad (5.3)$$

At this point we are able to find the heading vector. We know that the heading is perpendicular to the vector joining the two consecutive centers namely that

$$(c_{i+1} - c_i)^T h_i = 0$$

and we can also write: $h_i =$, therefore we have

$$(c_{i+1} - c_i)^T R_i h_0 = 0$$
if we stack the left side of the equation we’ll have

\[
\begin{bmatrix}
(c_2 - c_1)^T R_1 \\
(c_3 - c_2)^T R_2 \\
\vdots \\
(c_N - c_{N-1})^T R_N
\end{bmatrix} \quad h_0 = 0_{N-1 \times 1}
\] (5.4)

Therefore, we find \( h_0 \) robustly.

However with real data, this method has shown to be very sensitive to noise. This is the case because with a slight change in the \( t \) value the position of the center of rotation would be different and the vector joining the two vectors would be inaccurate.

**Using the Point Cloud**

The problem we face is that we don’t have the trajectory of the car. What we have are rotation matrices and translation vectors for all of the points. The calculations in the section 5.3 and in particular equation 5.1 show that having these information is not enough and in order to find the trajectory of a point we need to know its position in 3D world coordinates. We use the point cloud to derive the heading in this method. Moreover, from the Ackermann motion it follows that the trajectory of a car would be found, if we knew the position of any point on the back axis. Unfortunately, what we have is a 3D point cloud and we do not know their position with respect to the car. Therefore we turn to the our strong point and that is having a plentitude of frames.

First we note that if a point \( x \) lies on the axis of back wheels and we denote its displacement vector as \( \vec{d}_s \) then we have:

\[
h = R_{\theta_{x_{02}}} \vec{d}_s;
\]

For other points in the point cloud, however, this doesn’t hold. Using the same notations as the ones in the beginning section, and denoting the degree between the heading and the displacement vector for other points by \( \phi \) we have:

\[
h = R_{\phi} \vec{d}_s;
\]

if we plot all the displacement vectors we will notice that there is a fan of vectors with a degrees range 5.5.
5.4. Finding the Natural Coordinate System of the Car

where $\alpha$ is proportional to the length of the vehicle $L$ or the range of the point cloud and the radius of rotation $r$.

\[
\sin(\alpha) = \frac{L}{r}
\]

Therefore if we rotate the fan $-\theta/2$ degrees we are sure that these vectors include the the heading vector. Now we have this fan for the $i$th relative rotation. If we perform the same operation for all the relative rotations and then translate the results to a unit coordinate system, we can we accumulate these fans and we will see that the distribution of vectors has a peak close to the true heading vector of the car. In other words, if this rotation is between frames $i$ and $j$ we have:

\[
h_i \in \bigcup_k R_{-(\theta/2)} d_{ij}
\]

\[
h = R^T_i h_i
\]

The histogram of the degrees in this fan is illustrated in figure 5.6. The one sided histogram is due to the fact that this histogram corresponds to a
5. Ackermann Property of Vehicle Motion

Figure 5.6: Histogram of displacement vectors after being rotated $-\theta/2$ degrees

sequence where the car has only had rotation in one direction. This might cause a bias in the heading towards the direction of rotation. In order to be more precise we need to tighten the bound on the heading in the fan of displacement vectors. One thing to note is that, if we know the direction of the rotation, we would know that the points on the back axel are close to the end of the car, therefore we should weight those vectors more highly. Other thing to note is that in general we could be more sure of our estimate in frames where we don’t have much rotation. therefore the smaller the degree of rotation the more certain we are about the correct direction of heading, and also the closer all the other vectors are to the true heading. Hence we have two weighting coefficients. One is directly on the displacement vector fans where the displacement vectors which we believe are more likely to be the true heading are weighter heavier and the other weighting is uniform for each frame but weights those frames with smaller rotation more.

In order to perform these operations, it is much easier to work in 2D plane rather than 3D space. Moreover we know that the car moves more or less on a plane. We have already calculated the normal to this plane which is the rotation axis. Therefore, we transform all of the matrices and translation vectors onto 2D plane. The translation vectors are easily projected onto the 2D plane. For the rotation matrices, we will have to make the rotation purely around the given axis $a$. This is not the case in most of the frames because
5.4. Finding the Natural Coordinate System of the Car

of noise. First we find the degree of rotation around $a$ by the old rotation matrix which we call $R_{old}$. Then we calculate the new rotation matrix by having the axis of rotation and the degree. To that end, we define $v$ a vector that is orthogonal to $a$. we have:

$$v \perp a$$

$$\vec{v}_R = R_{old} \vec{v}$$

$$\vec{v}_r = \vec{v}_R - (\vec{v}_R^T \vec{a}) \vec{a}$$

Let’s define the angle between the two vectors $\vec{v}_a$ and $\vec{v}$ by $\alpha$. We could then find the rotation matrix $R$ with the rotation axis $a$ and angle $\alpha$:

$$R = aa^T + \cos(\alpha)(I_{3x3} - aa^T) + \sin(\alpha)skew(a)$$

where $skew(a)$ is defined:

$$skew(a) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$
6.1 Introduction

We have evaluated the pipeline by running it on several sequences. Here we tried the pipeline on 3 sequences. In the first one we used a toy car with remote control. For the second one, we have an outdoor scene.

6.2 Synthetic Data

We have used a CAD model and rendered a sequence using Maya software to simulate the outdoor scenario without the complex background. We are also able to readily generate background foreground mask as the rendered images are layered. In order to generate the motion of the car an algorithm to randomly generate valid vehicle motions were designed. Next, these motions were transferred to the CAD object. This is helpful because it provides us with a ground truth. This process was repeated 3 times, the mean error in finding the heading vector is 6.54 degrees. Figure 6.1 shows an example of the generated 3D model compared with the original model.

6.3 Outdoor Scenes

We have tried our pipeline on two different outdoor scenes.

6.3.1 Jimmy’s Pizzeria

This sequence was shot in a junction. The sequence starts with the car moving sideways and it continues with turning and moving away from the cam-
6. Experiments

![Figure 6.1: (a) The CAD BMW M3 model (b) The extracted 3D model](image)

This sequence has 150 images. The images were taken using a Canon High Definition camcorder (Canon LEGRIA HF M306). In order to avoid memory issues the images were down-sampled into 960 x 540 pixel images. The specific car that is under consideration is an Audi A3 car with metallic silver surface.

Challenges

This sequence is taken in outdoor and has therefore all the challenges that outdoor sequences have. The car is not the only object that moves in the sequence, pedestrians as well as other cars in the distance are moving. Also there is a tree which occludes the car in the beginning frames of the sequence which could make both the camera tracking and the background subtraction tasks difficult. The car is very shiny and has a very specular surface which make the tracking and triangulation extremely difficult. The features are not homogeneously distributed and are mostly on the door handles or headlight and there are almost no features on the sides of the car.

Background Modeling

As mentioned earlier the results of the background modeling are consistently putting low probability on certain parts of the car. These regions show either refraction or reflection. In figure 6.3 the background probabilities for a number of images is shown.
6.3. Outdoor Scenes

Figure 6.2: Images from the sequence Jimmy’s

Natural Coordinate System

The results of finding the natural coordinate system of the car is shown in figure 6.5. As it can be seen, calculated heading for the car looks reasonable. However, the true judgement about the correctness of the heading should be based on the final results, since the car is not symmetric at this point. If the natural coordinate system is not correct, then the symmetrization will lead to a surface that is not smooth around the axis of symmetry. The results of symmetrizing the visual hull can be seen in figure 6.4

Probabilistic Visual Hull and the Extracted Iso-Surface

The results are evaluated in two ways. First we put the 3D model into the scene and observe that it is in accordance with the silhouette. The second evaluation is to compare the car with another 3D model that has been produced manually and according to the correct proportions of the real car.
Figure 6.3: Background probabilities for a number of image in the sequence Jimmy’s. It could be observed that some regions on the windows and the surface of the car are in most of the frames assigned low foreground probabilities.

### 6.3.2 Sonnegrassasse Sequence

This sequence is taken with the same camera as Jimmy’s sequence and with the same resolution. The car under consideration is a Volkswagen Fox with a green exterior color.

**Challenges**

This sequence starts with the moving from a distance towards the camera. In the first frames, the car is partially occluded by bushes. This makes the camera calibration difficult, therefore we start the video in the reverse order. This means the beginning frame is where the car is closest to the camera.
Figure 6.4: figures (a) is the probabilistic visual hull for the Jimmy’s sequence. Figure(c) shows the extracted iso-surface figures(b) and (d) illustrate the corresponding visual hull and iso surface after including cameras at symmetric positions.

Figure 6.5: The coordinate axes overlayed on the extracted iso-surface
Figure 6.6: The visual hull setting: Each frame is taken to be a camera viewing the object from a different angle. It could also be observed in this figure, how cameras are unevenly distributed with respect to viewing angle.

Figure 6.7: Overlay of the extracted 3D model for the sequence Jimmy’s on different frames. It can be seen that in most cases it projects nicely onto the car.
6.3. Outdoor Scenes

Figure 6.8: The final 3D model for Jimmy’s sequence

Figure 6.9: Images from the sequence Sonnegstrasse
Figure 6.10: figures (a) is the probabilistic visual hull for the Sonnegstrasse sequence. Figure(c) shows the extracted iso-surface figures(b) and (d) illustrate the corresponding visual hull and iso surface after including cameras at symmetric positions.

**Results**

In figure 6.10c the 3D model in the natural coordinate system is displayed. In figure 6.10d the final 3D model can be seen.

**Probabilistic Visual Hull and the Extracted Iso-Surface**

The computed visual hull is close to the car observed. However due to the slope in the street and the resulting difficulties for calibration of the cameras, the pose estimation is not perfectly correct. In figure 6.11, the overlay of the 3D model onto the frames is illustrated. It could be noticed when the car is far away from the camera the 3D model starts to drift slightly.
6.3. Outdoor Scenes

Figure 6.11: Overlay of the Extracted 3D model of Sonnegstrasse sequence on the frames.

Figure 6.12: The 3D Model for Sonnegstrasse sequence from different viewing angles
Chapter 7

Conclusion and Future work

In this thesis we have dealt with the problem of finding the 3D model of a moving object and in particular a car. This is challenging due to the complexity of the background and the varied distance of the camera to the car in each scene.

We have used background subtraction with mixture of Gaussians to find the background probabilities for pixels of the images. We have also used a structure from motion method to find the poses of the cameras and also a cloud of 3D points corresponding to the car. We have then used the probabilistic visual hull framework to compute occupancy probabilities for each point in the occupancy grid. Since the settings of our problem is different from classical visual hull settings, we had to redefine some of the terms in this formulation to account for varying distance of the cameras from the object which would lead to different projection footprints.

We have also looked into the rotation and translation matrices and estimated the trajectory of the car. We have found the axis of rotation of the car hence the $y$-vector in the natural coordinate system of the car. Also we have found the heading vector of the car, using the Ackermann principle of rotation, hence the $x$-vector of the car. Having found the natural coordinate system of the car we could then use the symmetry cue to put the same set of camera at the other side of the car which we never saw. This way we will have a symmetrized probabilistic visual hull. Consequently we need to find the iso-surface in this probabilistic visual hull that best matches the silhouettes. We have an iterative step that would derive the iso-surface and project it again on the images to see how much the projection differs from the silhouettes.
7.1 Future Work

As explained in section 4.4, the assumption that images are independent is problematic. The problem will show itself in case of inhomogeneous cameras, therefore we tried a heuristic to compensate the introduced bias. However, it would be more mathematically just, if one could model the dependencies of the projected pixels directly. This would mean

\[ p(G|I_1, I_2) = \frac{p(I_1|I_2, G)p(I_2|G)p(G)}{p(I_1|I_2)p(I_2)} \]

and more specifically the two terms: \( p(I_1|I_2, G = 0) \), and \( p(I_1|I_2, G = 1) \) should be defined. Which would mean the probability of observing the image of the second camera given the information of the first camera and the fact that the projected voxel is full.

Another area in which more improvements could be made is having the pipeline run on multiple objects. Using motion segmentation or even user intervention, the number of moving objects or probably a bounding box around them would be provided. Next the pipeline could be run independently for each car. Every time we learn the background excluding only one object, therefore the background probabilities for each object would be different. Having information about the appearance of each model, we might be able to recover from occlusion.

In section sec:cov it was mentioned that the uncertainty on the camera parameters for different frames is assumed equal. It would be more accurate to have those uncertainties for different cameras.
Bibliography


