Improving Action Classifiers with Depth

Semester Thesis

Roger Odermatt
Department of Information Technology and Electrical Engineering
Computer Vision Laboratory

Advisor: Angela Yao
Supervisor: Prof. Dr. Luc van Gool

July 29, 2011
Abstract

The detection and classification of human activities and gestures in video clips has been a popular research topic during the last years. Lately, 3D data has proven to facilitate this task. In addition, the recent availability of depth sensors in normal products has produced many interesting applications for that research.

In this thesis, we use depth and video information from a Microsoft Kinect device in a Hough-voting framework designed for action recognition. The depth information appears to be an invaluable addition to the set of conventional image features, if available for training and testing. However, the goal of this thesis is to use 3D data during training of the classifier, but test it on 2D data only. For the purpose of experimentation and evaluation, different datasets have been recorded with the Kinect.

We present several modifications to the original Hough framework. In a first scenario, an attempt to imitate the depth information is made. The results of two different approaches show that depth imitation fails to considerably improve the average classification performance. In the second scenario, a more refined approach to regularize the training process of random decision trees using depth is adopted. The effects of this balanced use of depth data and normal image data look quite promising. Further, the use of depth as a means of image segmentation is examined. Aside from that, reconstruction of depth images from 2D data is intended and partially realized.

The evaluations of those modifications reach the conclusion that depth information is useful, even if only used for training the action classifier. However, it had to be recognized that the results are solely based on self-recorded datasets, making a fair comparison to other work impossible.
Acknowledgements

I would like to express my gratitude to all people who worked with me during this semester thesis. Especially my supervisor, Angela Yao, for patiently supporting me during the semester thesis. I also want to thank Christian Leistner who’s assistance and competence were a big help and a source of inspiration for me. Finally, let me also mention the people who volunteered for the recording of our datasets. Their commitment is most appreciated.
# Contents

1 Introduction  
1.1 Focus of this Work .................................................. 1  
1.2 Thesis Organization .................................................. 1  

2 Related Work  
2.1 Depth for 3D Body Model ............................................. 3  
2.2 Depth for Silhouette .................................................. 3  
2.3 Depth for Segmentation .............................................. 4  

3 Materials and Methods  
3.1 Setup .................................................................. 5  
3.2 Datasets .................................................................. 5  
3.3 Hough-Voting Framework ............................................ 6  
3.3.1 Training ................................................................. 7  
3.3.2 Testing ................................................................. 9  
3.4 Modifications ............................................................. 9  
3.4.1 Depth Imitation ..................................................... 9  
3.4.2 Depth Regularization ........................................... 11  
3.4.3 Depth Node .......................................................... 12  
3.4.4 Depth Segmentation ............................................ 12  

4 Experiments and Results  
4.1 Depth Imitation .......................................................... 14  
4.2 Depth Regularization ............................................... 16  
4.3 Depth Node ............................................................... 16  
4.3.1 Depth reconstruction ........................................... 18  
4.4 Depth Segmentation ................................................... 18  

5 Conclusion .................................................................. 21  

A Data ........................................................................... 23
List of Figures

3.1 Snapshots of the five actions from the HCI dataset. ................................. 6
3.2 Snapshots of the seven actions from the Office dataset. ......................... 7
3.3 Illustration of the Hough-Voting space. .................................................. 10

4.1 Confusion matrices of the control and depth cases. .............................. 13
4.2 Overview over the average classification performance of the control case, the depth case and the combined case, using depth and image data. .................................................. 14
4.3 Overview over the average classification performance of the control case and the two imitation attempts. .................................................. 15
4.4 Illustration of the depth regularization. .................................................... 16
4.5 Confusion matrices of the depth threshold weight function. .................... 17
4.6 Confusion matrices of the vote weighting with $\sigma(\frac{1}{depth})$. .......... 18
4.7 Two examples of Depth reconstruction .................................................. 19
4.8 Illustration of the depth segmentation. .................................................. 19
List of Figures
List of Tables

4.1 Overview over the selected feature channels distribution . . . . . . . . . . . . . . . . . . . 15
4.2 Overview over different vote weighting functions. . . . . . . . . . . . . . . . . . . . . . . . 17
Chapter 1

Introduction

During the last years, much research has been conducted in the field of activity recognition. The detection and classification of human actions in video clips has a wide range of applications. Apart from intelligent security systems for surveillance or monitoring sick and elderly people, the topic of human computer interaction is growing more and more important. One such alternative computer interface was of particular interest for this thesis, the Kinect.

Built by Microsoft for its Xbox 360 game console and thus meant as an entertainment product, the Kinect is still a very interesting piece of computer vision hardware. The combination of a conventional video camera and a depth sensor enables advanced action recognition and allows remote control of the system with mere hand gestures. As the Kinect is globally available, sold at an affordable price and can be connected to any personal computer, an open community of people interested in developing other free and innovative applications was formed.

However, in the average action recognition scenario, one does not have the valuable depth information at one’s disposal. Thus the goal of this thesis is to improve the performance of an action classifier that works with ordinary video input, by using the additional information exclusively for training. Such an improved classifier could then be used in existing systems without the need to upgrade the camera hardware.

1.1 Focus of this Work

In this thesis we present several modifications to a Hough-voting framework designed for human action recognition [11]. We use depth information during training of the classifier and test it on visual information only. For the purpose of experimentation and evaluation, different datasets have been recorded with the Kinect.

1.2 Thesis Organization

We first give an overview over some related works in Chapter 2. Additionally to the description of the Kinect related software and the self-recorded datasets, we will introduce the Hough-voting framework and
the implemented modifications in Chapter 3. The corresponding experiments and results are presented in Chapter 4. Finally, Chapter 5 concludes the thesis.
Chapter 2

Related Work

This chapter introduces some related research works. Most approaches use depth data for tracking human movements or gesture recognition and are dependent on the depth information during testing. No paper specifically tries to use depth as a means to improve performance of a purely visual classification system.

2.1 Depth for 3D Body Model

Shotton et al’s work [6] is especially noteworthy, because the tracking function of the Kinect device is based directly on the algorithm proposed here. The method quickly and accurately predicts 3D positions of body joints from a single depth image. In their work, they define labels for several body parts. These labels densely cover the human body and can be used to localize particular skeletal joints. The problem of pose estimation can then be solved by classifying the label of every pixel in an image using a randomized decision forest approach. This information is then pooled across pixels to generate proposals for the skeletal joint positions. Using a huge and highly varied training set allowed them to train very deep decision trees without overfitting and achieve state of the art accuracy.

In 2006, Knoop et al [8] also proposed an algorithm for tracking human body movements based on a 3D body model. Especially interesting is the new approach for fusion of different input sensors, i.e. 3D data from Time-of-Flight cameras or stereo reconstruction and 2D video information. Their Iterative Closest Point algorithm calculates the relations between data points and model points to determine the observed activity. The main differences between [8] and [6] are the low performance and the needed initialization process, making it less robust.

2.2 Depth for Silhouette

In this recent work [9], the researchers applied the discrete Hidden Markov Models (HMM) to train and recognize different human activities from binary and depth silhouettes features respectively. First, human activities are represented in the time-sequential depth silhouette features, before the continuous HMM can be applied effectively. In their system, they considered a four-state left-to-right HMM to model the human activities. Each activity was represented by a distinct continuous HMM. To recognize an activity, a feature
vector sequence obtained from the activity image sequence was applied on all trained continuous HMMs to calculate the action with the highest probability.

Another research work [2] from 2011 evaluates the utilization of Radon transformation on depth silhouettes. They show that applying Principle Component and Linear Discriminant Analysis to extract prominent activity features on depth silhouettes leads to improved results over the use of binary silhouettes.

### 2.3 Depth for Segmentation

The main idea in this work [4] from 2007 is that a reliable image segmentation approach can be integrated with very low computational efforts by making use of depth. They used this information to extend existing algorithms for segmentation, object detection and activity recognition efficiently and they planned to measure the quantitative improvements that can be gained this way in the future.
Chapter 3

Materials and Methods

This chapter gives a short overview over the setup used to acquire the image data, the recorded datasets and the Hough-voting framework that served as a basis for the experiments in Chapter 4.

3.1 Setup

There are many useful tutorials about working with the Kinect floating around the internet, e.g. [3] for Ubuntu linux. Here we will elaborate on the most important software packages.

The core component of our Kinect setup is the OpenNI framework [1]. The focus of OpenNI is certifying and improving interoperability of natural interaction devices with other applications. It provides a set of open source APIs that support basic voice recognition, hand gestures and body motion tracking. It further contains a selection of sample applications. Additionally, the SensorKinect module for OpenNI was used, which is a driver for reading and interpreting the sensor input from the Kinect. Another noteworthy module is the NITE module, which can be used on top of the SensorKinect module, to give information about skeletal joints. For recording data, the NiViewer, i.e. a sample application that comes with OpenNI, was modified to save video frames in a raw binary format. Finally, the OpenCV library for computer vision was used for various tasks, e.g. image processing and conversion.

3.2 Datasets

In order to evaluate the performance of our action recognition approach, we need a dataset with video and corresponding depth data. Because of the lack of available datasets containing both modes of data, we decided to self-record suitable datasets. We recorded eight people performing twelve different actions while sitting at a table. Two repetitions for each action were recorded. The twelve actions were separated into the two datasets. The first dataset consists of five typical human-computer interaction (HCI) gestures that involve wide arm movement, making the actions well distinguishable. The HCI dataset includes the actions cheering, clapping, hitchhiking, pointing and waving, as shown in Figure 3.1. The remaining seven actions could be part of a meeting or office scenario and are shown in Figure 3.2. The Office dataset includes casual actions like writing, coughing, checking the time or scratching the head. Additionally, it contains not only
two very subtle actions, which involve only small head movements, i.e. nodding and shaking the head, but also the very individual gesture of being bored.

![Snapshots of the five actions from the HCI dataset.](image)

The depth images have been normalized in a way that the brightest parts correspond to closest proximity of the person, while the background, i.e. depth above 3 meters is completely black. This normalization achieved better results than using the depth information of the whole scene (cf. [2]) and will thus be used for the experiments in Chapter 4.

### 3.3 Hough-Voting Framework

Hough Forests and Hough-Voting was previously introduced for object detection and tracking in [5] and successfully extended for action recognition of single persons [11]. It also showed good results in classifying interactions between two persons or several groups of persons [10]. The original Hough-voting framework for action recognition involves two steps. The first stage includes localization and tracking of the person performing the action and then building normalized action tracks. We can skip this step for our datasets, as the actions are already naturally localized in the center of the image. The second stage includes the actual classification of actions. To find a mapping between training samples and the corresponding votes in a spatio-temporal Hough space, random decision trees are trained. The leaves of the trees form a discriminative
multi-class codebook and share features across all action classes, thus enabling leaves to vote for all classes in a probabilistic manner. We will now explain the concept of the Hough-based voting more closely, focusing first on the training and the testing afterwards.

### 3.3.1 Training

To train a random Hough Forest, a set of training sequences for all action classes is required. Each tree is constructed from a set of patches \( \{ P_i = (I_i, c_i, d_i) \} \), where

\[ P_i \] is a 3D patch (e.g. of \( 16 \times 16 \times 5 \) pixels) randomly sampled from the action tracks.
CHAPTER 3. MATERIALS AND METHODS

\( I_i \) are the extracted features of the patch, i.e. \( I_i = (I^1_i, I^2_i, ..., I^F_i) \in \mathbb{R}^4 \), where each \( I^f_i \) is feature channel \( f \) at patch \( i \) and \( F \) is the total number of feature channels.

c\(_i\) is the patch’s action class or label.

d\(_i\) is a 3D displacement vector from the patch center to the action track center in space and time.

From this set of 3D patches, a tree is built by selecting a binary test \( t \) for every node, starting at the root. The binary tests are comparisons between two pixels from the same feature channel \( f \), at different locations \( p \in \mathbb{R}^3 \) and \( q \in \mathbb{R}^3 \) with a threshold \( \tau \), i.e.

\[
 t_{f,p,q,\tau} (I) = \begin{cases} 
 0 & \text{if } I^f(p) < I^f(q) + \tau \\
 1 & \text{otherwise} 
\end{cases} \tag{3.1}
\]

The binary tests split the training patches depending on the patch appearance into two child nodes. We iterate this process until arriving at the leaf nodes, meaning either the maximum depth of the tree is reached or insufficient patches remain for further splitting. Each leaf node stores the proportion of the patches per action class label \( p_c \) and their respective displacement vectors \( D_c = \{ d_i \}_{c_i = c} \). An optimal binary test minimizes the class label uncertainty or the center offsets uncertainty of the patches in both child nodes. For this purpose, the following two uncertainty measures are used:

- **class uncertainty measure**
  
  \[
  U_1(A) = -|A| \cdot \sum_c p_c \ln(p_c) \tag{3.2}
  \]

  where \( A = \{ P_i \} \) is a set of patches, \( |A| \) is the number of patches in set \( A \) and \( p_c \) is the proportion of patches with label \( c \) in set \( A \).

- **center offset measure**
  
  \[
  U_2(A) = \sum_i ||d_i - \overline{d_A}||^2 \tag{3.3}
  \]

  where \( \overline{d_A} \in \mathbb{R}^3 \) is the mean offset vector of set \( A \).

To find such an optimal split, a pool of binary tests with random values of \( f, p, q \) and \( \tau \) is generated and tested at each non-leaf node. Hence, the binary test which scores the highest potential information gain \( \Delta H \) according to one of the two uncertainty measures is selected, i.e.

\[
 \Delta H = -\frac{|A_l|}{|A_l| + |A_r|} U(A_l) - \frac{|A_r|}{|A_l| + |A_r|} U(A_r) \tag{3.4}
\]

where \( A_l \) and \( A_r \) are the left and right subsets of the training patches and \( U(A) \) the offset or class uncertainty measures.

By randomly alternating between minimizing class or center offset uncertainty, the leaves contain patches that tend to have low variation in label and offset, thus vote with high confidence during the testing.
3.3.2 Testing

For classifying and localizing actions, patches from test action tracks are densely sampled and passed through each tree of the Hough forest. When a patch arrives at a leaf, votes proportional to $p_c$ are cast for the spatial and temporal location of the action and the action class, according to a 3D Gaussian Parzen window estimate. The votes from all patches are integrated into a 4D Hough accumulator. For a formal description we refer to [11]. The Hough accumulator is then marginalized across the spatial dimensions into a 2D Hough space, where the global maximum indicates class label and time of an action track. This process is illustrated in Figure 3.3.

3.4 Modifications

In order to provide informative results about the generalization capabilities of an action classifier, all experiments in Chapter 4 have been performed using cross-validation with two cross-folds. Two repetitions of each action of four subjects were used for training, while the other four subjects were used for testing the classifier and vice versa. This gives us an average classification performance, which we use to compare the effects of the different depth related modifications to the original Hough-voting framework. The following parameters were used:

- Image resolution of $160 \times 120$ pixels
- 3D patch size of $16 \times 16 \times 5$ pixels
- Maximum tree depth size of 16 and minimum number of patches to continue splitting of 10
- Set of 4 trees for each cross-fold
- 1000 training patches per sequence, i.e. 8000 for each action class

The following set of image features or feature channels has been used:

- Lightness $L$ and color components $a$ and $b$ of the Lab color space
- Second order frame difference $Idd = frame_i - frame_{i-2}$
- Values of the $x$ and $y$ components of the gradient $Ix$ and $Iy$

Those parameters and image features have been successfully applied in the past and will allow us to observe the effects that the following depth modifications cause.

3.4.1 Depth Imitation

The first modification is dedicated to the idea of imitating the depth, or more precisely, imitating the trees that are trained, if depth could be used for testing. We want to select other features that split the training patches similar to the optimal depth split. If we can build an identical tree from image features only, in theory, the performance should be closer to the depth case. For this we thought of two different ways, a patch-wise and a distribution-wise approach.
Figure 3.3: Illustration of the Hough-Voting space for a sequence showing the action cheer, where (a) shows the output of the Hough voting framework, (b) the accumulation in time for the cheer class, (c) the accumulation in x for the cheer class and (d) the 2D space in time and action label with the highlighted global maximum, which indicates the recognized action.

**Patch-wise**

We investigate in a first step for each patch $p_i$, if the patch is split to the left or the right side of a node. After finding the optimal binary test $t_{depth}$ for the depth channel, we now try out binary tests $t$ on the other feature channels and evaluate with a score $V(t(A))$, i.e.

$$V(t(A)) = \frac{1}{|A|} \sum_i 1(t(p_i) = t_{depth}(p_i))$$  \hspace{1cm} (3.5)
where $A = \{p_i\}$ is a set of patches, $|A|$ is the number of patches in set $A$ and $1(\cdot)$ is an function, which indicates if the patch would be split to the same side of the node for both binary tests. The test with the highest score is selected and the split performed accordingly. Note that the actual side, i.e. left and right, does in fact not matter, thus a score of 0% is treated equal to one of 100% in the actual implementation.

**Distribution-wise**

The Hough-voting uses the patch distributions of the leaves to cast votes. Thus instead of focusing on the individual patches, there is a more natural way to imitate depth. A tree with similar leaves as the depth tree is expected to cast similar votes, thus resulting in improved performance. The natural way to describe the divergence of two probability distributions $p^{\text{color}}$ and $p^{\text{depth}}$ over the same probability space is their cross entropy, which is defined as

$$H(p^{\text{color}} || p^{\text{depth}}) = -\sum_c p^{\text{color}}_c \ln(p^{\text{depth}}_c)$$

(3.6)

where $c$ denotes the class label.

Again, we first run the binary tests on the depth channel and calculate the optimal patch distribution $p^{\text{depth}}_c$ for each action label $c$. The task of finding the binary test on image channels, which minimizes the divergence between the two probability distributions now corresponds to minimizing their cross entropy [7].

**3.4.2 Depth Regularization**

If we take a look at the depth images of our datasets in Section 3.2, we realize that the areas with different depth values are the most noticeable. For example, the pointing hand close to the camera or the areas along the silhouette of the person, provide the most useful information concerning action recognition. We now use this hypothesis to influence the training in a more subtle way, by adding a factor $\Delta^{\text{depth}}$ to the quality measure for optimal splitting, i.e.

$$\Delta H^* = (1 - \alpha) \Delta H + \alpha \cdot \Delta^{\text{depth}}$$

(3.7)

where $\Delta H$ is the information gain of the node.

The parameter $\alpha$ steers the influence that the $\Delta^{\text{depth}}$ has during splitting. Since we want to make use of this depth differences in the depth channel, we penalize a binary test that does not compare pixels of different depth, but at the same time we still maximize the information gain. Of course, many different definitions of $\Delta^{\text{depth}}$ are possible. Let $\delta$ be the difference of the depth values at locations $p$ and $q$, i.e. the pixels we compare with our binary test $t$. We will compare the results of $\Delta^{\text{depth}}$ defined as:

- **absolute depth difference measure**

$$\Delta^{\text{depth}} = \frac{\sum_i |\delta_i|}{|A|}$$

(3.8)
or as depth difference rate measure

$$\Delta_{\text{depth}} = \frac{\sum \mathbf{1}(\delta_i \neq 0)}{|A|}$$

where $|A|$ is the number of all patches $p_i$ in set $A$ and $\mathbf{1}(\cdot)$ is an indicator function.

### 3.4.3 Depth Node

The modifications presented thus far use depth information to change the way a random decision tree is trained. This led to improved performance results when done right. However, since we do have the depth information available for the training, maybe we can train a tree in a way that allows speculations on the depth information of the testing data. What we need is a way to store the depth information of the training data in the trees. For this purpose we introduce a third type of tree node, called the depth node.

Thus, while sampling patches for the Hough training, in addition to the action label and center offset, we attribute each patch with its depth information. For simplicity, we limit this information to a single number, e.g. the average depth of a patch. In addition to voting for the spatio-temporal location and label of the action, this makes it possible to vote for depth in a pseudo depth-space.

In order to vote for depth with high confidence, the patches with same depth have to be kept together during training. Similar to the center offset uncertainty we introduce the third uncertainty measure for depth:

$$U_3(A) = \sum_i ||\text{depth}_i - \text{depth}_A||^2$$

where $\text{depth}_A$ is the arithmetic mean depth of the patches of set $A$.

This measure is then minimized in the depth nodes.

### 3.4.4 Depth Segmentation

Having depth data helps with segmentation, as shown in [4]. The normalized depth images enables perfect segmentation for the HCI and Office datasets. Hence, training patches can be divided into foreground and background patches by simply measuring the patches depth value. We want to find out, if background patches hold valuable information about the action or if they are just clutter.
Chapter 4

Experiments and Results

This chapter gives an overview over the effects that the different depth related modifications to the original Hough-voting framework have on the average classification performance.

For that reason, we need a starting point, i.e. a control case. This means that we first apply the Hough-voting using only the above low level feature channels for training and testing. An average classification performance of 44% for the HCI and 19% for the Office dataset was achieved. The corresponding confusion matrices are shown in Figure 4.1, where it can be seen that some actions are very hard to recognize. The low average performance can be attributed to the small amount of training data and also the lack of direction during recording. In other words, the actors were at the liberty to perform the actions very individually.

![Confusion matrix](image)

Figure 4.1: Confusion matrices of the control case for (a) HCI and (b) Office using the standard image features for action classification. Average classification performance of 44% (HCI) and 19% (Office) has been achieved.

The next logical step is to investigate the performance of the depth feature channel. The overview in
Chapter 4. Experiments and Results

Figure 4.2 illustrates that the depth information is more distinctive than the color channels. Interestingly, the combined case, where depth and color channels are available for training and testing, performs worse than the case using depth only. If all channels are available, apparently not only the depth channel is selected during training, or the bars would look identical. This is no contradiction, but the result of a generalization error. It is entirely possible that the color feature channels provide higher information gain for the training data. Or to be precise, overfitting occurs. In any case, depth is a valuable feature channel. What we intend to do with the following experiments, is to improve the performance over the control case by using the depth feature channel exclusively during training, while testing only on the other feature channels.

![Figure 4.2: Overview over the average classification performance of the control case, the depth case and the combined case, using depth and image data.](image)

4.1 Depth Imitation

The first experiments are dedicated to the idea of imitating the depth according to equation 3.5. Unfortunately, we could not reach a perfect patch-wise imitation score at all nodes in our experiments. Instead we observed that the early splits, which involve all or most of the training patches, usually score worse than the splits close to the leaves. This can be expected, as it is obviously hard to imitate depth with for example a color channel, because of the initial amount of various different cloth textures and background colors. If we weight the split similarity with the number of patches involved, we measure a total similarity score of around 85% for the two datasets. Despite this quite high similarity, the overall performance actually drops, as shown in Figure 4.3. The cause of this is likely the less similar splits near the tree root, and those early deviations lead to an entirely different tree, i.e. the distribution of patches in the tree leaves produces very different votes during testing. Also, if we take a look at the distribution of selected channels...
in the control case and compare it to this experiment (cf. Table 4.1), we can see an significantly increased proportion of the $b$ color channel and the horizontal gradient. It is not surprising that color may not be the best way to imitate depth in general.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>18.25</td>
<td>11.9</td>
<td>17.05</td>
<td>18.05</td>
<td>10.95</td>
<td>16.55</td>
</tr>
<tr>
<td>$a$</td>
<td>16.95</td>
<td>15.8</td>
<td>16.3</td>
<td>17.6</td>
<td>14.2</td>
<td>16.95</td>
</tr>
<tr>
<td>$b$</td>
<td>17.35</td>
<td>23.75</td>
<td>16.35</td>
<td>18.15</td>
<td>25.45</td>
<td>16.9</td>
</tr>
<tr>
<td>$Idd$</td>
<td>17.35</td>
<td>18.6</td>
<td>16.45</td>
<td>16.95</td>
<td>15.8</td>
<td>16.8</td>
</tr>
<tr>
<td>$Ix$</td>
<td>14.95</td>
<td>19.35</td>
<td>17.6</td>
<td>14.5</td>
<td>21.25</td>
<td>16.05</td>
</tr>
<tr>
<td>$Iy$</td>
<td>15.15</td>
<td>10.6</td>
<td>16.25</td>
<td>14.75</td>
<td>12.35</td>
<td>16.75</td>
</tr>
</tbody>
</table>

Table 4.1: Overview over the share of selected feature channels for the control case and the two different imitation approaches expressed as a percentage.

The results of the next experiment are shown in Figure 4.3. The performance of the distribution-wise imitation approach lies very close to the control case. Table 4.1 shows that the amount selected gradient features increased slightly at the expense of all other channels evenly. Given the properties of the subjects performing the actions, this more balanced approach proves to generalize much better. In case of the Office dataset the performance improves actually. The fact that depth only performs much better, however, shows that we can not easily substitute depth information with gradient or color features.

![Figure 4.3](image_url)

Figure 4.3: Overview over the average classification performance of the control case and the two imitation attempts.
CHAPTER 4. EXPERIMENTS AND RESULTS

4.2 Depth Regularization

The results for both variations of $\Delta_{depth}$ and the whole range for variable $\alpha$ are shown in Figure 4.4. We observe a largely positive effect on the Office dataset, but also the performance on the HCI dataset can be slightly improved at low values of $\alpha$. Both datasets show bad results for $\alpha = 1$. The results also show that the absolute depth difference measure and the depth rate measure are equally qualified for our task.

![Figure 4.4: Average classification performance for the regularization with depth. Note that $\alpha = 0$ is the control case, variation 1 is defined in equation 3.8 and variation 2 in equation 3.9.](image)

It can further be seen that this approach influences especially those nodes, where many patches have depth information. However, the nodes that split only patches with no discriminative depth information, e.g. from the background in our case, are not influenced at all.

4.3 Depth Node

The depth information obtained from the depth node is applied to the classification process in different ways. We first attempt to weight the 4D Hough votes with depth in different ways. The goal is to find out, which patches are the most informative according to depth. For example, if we believe that patches farthest from the camera are most informative, we could multiply the votes with their high depth value.

Table 4.2 comprises a non-exhaustive list of different weighting functions that were factored into the votes. The results show that the mere sorting of patches according to average depth already effects the performance slightly in either direction. What also can be observed is that for the HCI dataset, weighting has a detrimental effect on performance while having the opposite effect on the Office dataset. The confusion matrices in Figure 4.5 clarify what happens in the case where depth is used as a threshold. It can clearly be seen that those actions with lowest depth, i.e. the subject’s arms are closest to the camera, get more votes. For the HCI dataset this is a serious problem, as the cheer action was already monopolizing votes in the control case. The bad performance of the HCI dataset with thresholding can be explained with the unique character of the cheer action. Cheering occupies much more background than the other actions.
CHAPTER 4. EXPERIMENTS AND RESULTS

<table>
<thead>
<tr>
<th></th>
<th>HCI</th>
<th>Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>control case</td>
<td>43.75</td>
<td>18.75</td>
</tr>
<tr>
<td>sort only</td>
<td>42.5</td>
<td>19.6429</td>
</tr>
<tr>
<td>$\sigma(\frac{1}{d})$</td>
<td>40</td>
<td>28.5714</td>
</tr>
<tr>
<td>$\sigma(d)$</td>
<td>32.5</td>
<td>23.2143</td>
</tr>
<tr>
<td>$d$</td>
<td>36.25</td>
<td>25.8929</td>
</tr>
<tr>
<td>$d^2$</td>
<td>42.5</td>
<td>25</td>
</tr>
<tr>
<td>$\frac{1}{d}$</td>
<td>20</td>
<td>14.2857</td>
</tr>
<tr>
<td>$\begin{cases} 0 &amp; \text{if } d = 0 \ 1 &amp; \text{otherwise} \end{cases}$</td>
<td>23.75</td>
<td>27.6786</td>
</tr>
</tbody>
</table>

Table 4.2: Overview over the average performance of a selection of vote weighting functions. Note that depth $d$ here denotes the grayscale value of the normalized depth images, i.e. $d = 0$ would be the black background. Also note that if $d = 0$, no vote is cast. $\sigma(\cdot)$ is the sigmoid function.

This results in disproportionally many patches of the cheer action that have depth. The effect of taking the background votes away can also be seen in section 4.4. The average performance for the Office dataset, on the other hand, increased considerably. On closer examination, one can see that the subtle actions involving no low depth values, e.g. nodding or shaking, are never recognized correctly, while the other actions actually perform much better.

In Figure 4.6 we see how the results change if we multiply the votes with $\frac{1}{\sigma(d)}$. This means we weight actions with high depth higher now. As expected, the more subtle actions are now recognized, at
least to a certain degree. Not surprising, as the $\sigma(\text{depth})$ case actually shows the same trend as Figure 4.5.

### 4.3.1 Depth reconstruction

We also attempted to reconstruct a depth image of the testing images by voting into the pseudo depth-space instead of voting for the action label. Figure 4.7 shows the results of some experiments using trees specifically trained to restore depth information. These trees were trained using much more but also smaller patches. As we can see, the reconstructed depth images are roughly comparable with the real depth images. Depth reconstruction should be invaluable. Suppose that original and reconstructed depth image looked nearly the same. We then would no longer need the depth feature channel, because in a first stage, the depth could just be reconstructed. Feeding the reconstructed depth channel into the classifier afterwards, i.e. a classifier trained with actual depth data, should consequently lead to the best possible results.

### 4.4 Depth Segmentation

We wanted to find out, whether background patches hold valuable information or not. Figure 4.8 shows that for our two datasets, the effect of segmentation has great influence on the average performance. The case with only foreground patches performs worse than the case with only background patches for the HCI dataset. Thus, segmentation further increased the overfitting for the cheer action. The good result of using only background patches is difficult to explain, but proves that background patches hold valuable information for classification. The results of the Office dataset are more comprehensible. The performance is actually better than the control case for the most part. The best results are achieved with a mix of different
(a) In this example the minimum depth of a patch was stored during training.

(b) In this example the maximum depth difference of a patch was stored during training.

Figure 4.7: Depth reconstruction

Figure 4.8: Average classification performance for the segmentation with depth. Note that the number of foreground and background patches add up to a total of 1000 patches. Note that the dashed lines indicate the control cases.

Of course, the usefulness of the background is very dataset dependent. For example when classifying patches. For the HCI dataset this optimum is close to the case of just randomly selecting patches, as it is usually done. The Office dataset though particularly benefits from using more foreground patches than usual.
sports videos, the background already narrows the possible action down, e.g. playing football instead of gymnastics. Since our datasets have a static background, where certain background parts are visible or hidden depending on the subject’s size and the performed action, it contains less, but still some information. This knowledge has at least one useful application, i.e. the actual implementation can be more efficient, enabling the training with more useful patches and reducing the already limiting memory demand.
Chapter 5

Conclusion

The goal of this semester thesis was to improve an existing action classifier, using 3D data during training, but testing on 2D data only.

We showed that depth is the most discriminative feature channel for our datasets. We then presented several modifications to the original Hough framework.

The first modification made an attempt to imitate the depth information. The results showed that depth imitation fails to considerably improve the average classification performance in case of the patch-wise approach. However, the distribution-wise approach actually performed nearly identically or even slightly better than the control case. In conclusion, there is still no definite feature channel suitable to perfectly substitute the depth information in the general case. Thus instead of forcing this similarity, without regard to the two uncertainty measures originally used, the subsequent modifications took a less radical approach to use depth.

The second and more refined approach adopted the regularization of the Hough training process using depth. The effects of this balanced use of depth data and normal image data looked quite promising. Thus it seems that depth does indeed help however much.

Further, the use of depth as a means of image segmentation was examined. It has been shown that deciding upon a fixed foreground-background patch ratio can already improve the performance.

The introduction of the depth node and the effect of depth weighted voting was shown to be heavily dependent on the action. Therefore it is not a good way to improve our action classifier for general actions. Aside from that, reconstruction of depth images from 2D data was intended and partially implemented. However, the outlined future application is expected to be very hard if not impossible to realize.

In summary, it can be stated that the presented modifications to the Hough-voting framework resulted in an increased performance on average. However, it can not be excluded that the depth information is just preventing overfitting to some degree. Thus our datasets may not be well suited to draw a final conclusion on the value of depth data in general.
Appendix A

Data

- This DVD contains the modified source code of the Hough-voting Framework for action recognition, some MATLAB files to set up and evaluate experiments as well as the software used for recording and converting datasets. The HCI and Office datasets are also included.
Bibliography


