Hand segmentation with random forest on RGB-D data

handed in
MASTER'S THESIS

B.Eng. Dominik Zech

born on the 29.06.1987
living in:
Christoph-Probst-Str. 16
80805 Munich
Tel.: +49 163-5106521

Chair of
AUTOMATIC CONTROL ENGINEERING
Technical University of Munich

Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss
Univ.-Prof. Dr.-Ing. Dirk Wollherr

Supervisor:	Prof. Ph.D. Dongheui Lee, M.Sc. Shile Li
Start:	01.08.2016
Intermediate Report:	30.11.2016
Delivery:	09.03.2017

TECHNISCHE UNIVERSITÄT MÜNCHEN
LEHRSTUHL FÜR STEUERUNGS- UND REGELUNGSTECHNIK
UNIV.-PROF. DONGHEUI LEE

July 15, 2016

M A S T E R ' S T H E S I S
for
Dominik Zech
Student ID 03657259, Degree EI

Hand segmentation using random forest on RGB-D data

Problem description:

This project aims to segment a human hand from an RGB-D image in a hand-object interaction scenario. One approach for hand segmentation is using random forest classifier [1]. In previous random forest based methods, RGB alone based classifier [3] and depth alone based classifier [2] have been proposed. However, limitations of both RGB alone method and depth alone method exist, for example it is hard to distinguish skin colored background from the human hand using RGB alone method. And by using depth alone, valuable color information is lost. To the best of our knowledge, no RGB and depth combined classifier for human hand segmentation task exists. Perhaps one of the reasons is that the ground-truth data is not available for RGB-D data. In this project, we aim to train an RGB-D combined classifier for hand segmentation tasks, especially focused on hand-object interaction scenarios.

Tasks:

- Literature study about hand segmentation methods using random forest.
- Collect ground truth data for hand segmentation.
- Implement the training framework for random forest method.
- Evaluation of segmentation accuracy using different low-level features for both RGB and depth data.

Bibliography:

[1] A Criminisi and J Shotton. Decision forests for computer vision and medical image analysis. pages 273–293, 2013.
[2] Byeongkeun Kang, Kar Han Tan, Hung Shuo Tai, Daniel Tretter, and Truong Q. Nguyen. Hand segmentation for hand-object interaction from depth map. 2016.
[3] Cheng Li and Kris M Kitani. Pixel-level hand detection in ego-centric videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3570–3577, 2013.

Supervisor:	M. Sc. Shile Li
Start:	01.08.2016
Intermediate Report:	30.11.2016
Delivery:	16.03.2017

(D. Lee)
Univ.-Professor

Abstract

The increasing relevance and importance of hands as high-bandwidth input devices for machines and the advantages of demonstrating grasping motions to robots with a human hand demand for a method to automatically distinguish the hand from objects. In this thesis we propose a pixel-level based classification scheme using random forest and low-level color as well as depth features. Incorporating both color and depth information raises the issue that respective data sets are not readily available. Therefore, this thesis proposes a feasible method to label data sets containing both color and depth information. This method is then applied to collect two data sets. The training data set contains 3420 images showing 38 different desktop scenes and the verification data set contains 922 images showing 16 scenes. All scenes contained in the data sets show a cluttered desktop with randomly arranged objects. The data sets are used to train and compare a variety of random forest classifiers employing different feature vectors. The vectors differ in the type of features contained. Specifically one vector contains depth features only, one vector contains color features only, one vector contains all features and one vector contains the 60 best features. The comparison shows that applying the classifier on the region of interest results in the highest performance and efficiency. Employing depth and color features together results in a more robust classifier in contrast to classifiers relying on either depth of color information. Moreover, the comparison demonstrates that the size of the feature vector can be reduced to 60 features without causing a significant drop in the segmentation quality. Further analysis shows that post-processing the classification output reduces the noise and results in smoother contours of the hand in the segmentation mask. The evaluations revealed that the data sets collected have to be improved for future work. This conclusion suggests that increasing the number of examples with realistic grasps and decreasing the amount of examples of the hand above the desktop without grasping an object could drastically improve the segmentation result.

Contents

Chapter 1

Introduction

The segmentation of hand and skin has been researched for several decades. There are lots of applications where hand segmentation is applied, for example in hand tracking, in gesture recognition or in object modeling in robotics. The increased computational power of consumer computers allows for more complex input strategies and information processing. Thus, interaction or communication between a robot or in general any machine and humans is aimed to be more generic and intuitive by making use of hands as high bandwidth input devices.

Principally, there are two types of hand-machine communications, one involving the hand and the other involving hand-object interaction. First, let us have a look at some examples which typically do not involve interaction with objects. There already are systems using the hand or body as input device for machines, for example games for *Microsoft*'s *Xbox* with *Kinect* sensor. However, researchers aim to use hands as high-bandwidth input devices, thus improving the quality of hand-machine communication. A reliable hand-machine communication could make remote controls for television redundant and in addition it would make the interaction more intuitive. Similarly, the upcoming trend of VR (Virtual Reality) would benefit from hand controllable applications. Second, let us have a look at hand-object interactions in the following. In robotics, learning by demonstration is used to teach a robot how to execute a task which includes learning from observing humans doing a task. Hand-machine communication has benefits when it comes to generation of three-dimensional object models, as well. One could think of applications where a robot needs to learn the model of unknown objects, for example to track and interact with the objects. Another scenario could be a rapid replication system, where a 3D-printer is used to replicate an object in short time. For both applications a human operator could rotate the object in front of a set of sensors, for example a camera, and the machine extracts the object's 3D model from the sensor data. Currently, researchers also work on approaches to model the hand in hand-object scenarios with the focus to teach artificial hands how to grasp objects.

In general, hand-object interaction imposes a more complex problem, at least on hand modeling, due to strong occlusions. Although occlusions do occur in scenarios without objects as well, they are of another type. In this case the hand does only occlude parts of itself, for example a thumb placed on the palm and covered by the remaining four fingers. Occlusions in hand-object scenarios include both self-occluded ones and occlusions caused by objects which are grasped. Current hand modeling approaches just use a single depth camera. The use of a depth camera does not just remove the problem of different skin colors but also makes the RoI (Region of Interest) detection for hand-only scenarios easier. In the current hand modeling approaches there is no need for an additional skin segmentation step after the RoI extraction, because the proposed approaches do not consider hand-object interaction. However, hand-object interaction scenarios require a system to segment the hand, because the RoI still contains the handled object. The segmentation is mostly a preprocessing step; thus, approaches with little computational expenses are searched. A reliable and robust hand segmentation method eases the exclusion of irrelevant pixels and keep only relevant pixels.

This thesis deals with the problem of hand segmentation in RGB-D (Red, Green, Blue and Depth) images. The focus is on hand-object interaction, but the proposed framework and the captured training data set does not exclude hand-only scenarios. Recently, RF (Random Forest) classifiers have been used for pixel-wise skin detection with either color or depth information. However, using either color or depth information excludes important segmentation cues. Especially, relying only on color makes the distinction of human skin and skin colored objects difficult. Further, depth features might struggle if the object's shape is similar to the shape of the hand or skin. To the best of our knowledge there is no approach combing both color and depth information to segment hands or skin with a single classifier in images or videos. One reason for this might be the missing data set for training. In this thesis, an approach to label the data in a feasible manner is first described and then used to collect two data sets. The data is then used to train a RF classifier with only depth features, only color features, all features and a feature subset containing only the most important features, in order to compare the classification performance of the different feature configurations. However, we expect the approach using both color and depth information to achieve the best results because the color and shape characteristics evaluated together provide a better cue for skin segmentation than using either color or depth information.

In Chap. 2 the theoretical background to this thesis is given. The technical approach describing the scheme for data collection, feature selection and parameterization of the classifier is explained in Chap. 3. The experiments and their results are presented in Chap. 4. Chapter 5 discusses the experimental results and finally, in Chap. 6 the conclusion and outlook for future works are given.

1.1 Related Work

Tackling the problem of hand segmentation in videos or images has been done for a long time solely based on the color information; however, the recently available consumer RGB-D cameras allow to exploit color independent depth information as well. The approaches, whether for hand modeling or segmentation, exploiting solely depth information look promising; moreover, they encourage to seek for other ways utilizing depth information.

Nonetheless, let us begin with traditional approaches of hand segmentation using only the color information. Our intuition tells us that for hand segmentation the skin color can be a powerful cue. Therefore, Vezhnevets et al. conducted a survey on pixel-based skin color detection methods [VSA03]. Kakumanu et al. extended the investigations on color based skin detection by including more recent approaches [KMB07]. The latter compared the advantages of different color space representations, different skin-color classifiers and illumination adaptation approaches. Kakumanu et al. conclude that the choice of the best color space does not only depend on the type of classifier but also on the amount and quality of training data. In particular, histogram-based Bayes classifiers show good results when trained with a large data set, while NN (Neural Network) and mixture models achieve a comparable performance with a small data set. According to their results, non-linear color space transformations are computationally expensive and thus not adequate for real-time applications. Also, they rebut that leaving out the intensity component of the color space, in order to be more robust against illumination changes, reduces the skin classification accuracy. However, both surveys [VSA03, KMB07] do not include investigations on RF classifiers which is a popular approach in recent papers. In [LK13, SSK$^+$13, WA15, KTT$^+$16], the scientists use RF-based classifiers for different tasks because RFs have shown their speed and effectiveness. Moreover, RFs are fast if implemented on a GPU (Graphics Processing Unit) [Sha08].

The approaches which exploit depth information obtain the information from the depth sensor of a RGB-D camera. The recent development of consumer RGB-D cameras such as *Microsoft Kinect* or *Asus Xtion* makes it easy to incorporate the depth information of a scene because they capture an RGB (Red, Green and Blue) image and a dense depth map in real-time. Shotton et al. employ the spatial information of the scene to recognize and label different body parts of humans [SSK$^+$13]. From the labeled body parts they infer the joint positions. They do not account for interaction with objects in their set-up; nonetheless, the choice of features and classifier types also can be used in case of hand-object interaction. The feature set contains a bundle of easily computable differences of depth-invariant locations in the depth map. They choose those features in order to reduce computational complexity; whereas, with a higher computational cost more sophisticated shape features could be used [BMP02]. In [WA15], Wan and Aggarwal are interested in tracking objects which are grasped and moved by a human hand. They exploit the information of a RGB-D camera with egocentric perspective. However, compared to the approach in [SSK+13], they do not use features from the depth map to classify the object or segment the hand. Their approach consists of a foreground-background segmentation at the beginning followed by a skin detection step of the foreground which leaves a mask solely containing the object to be tracked. They use a method based on the histogram of depth values to segment the foreground from the background. A realistic assumption for egocentric camera perspectives is that there is a gap in depth values between foreground and background. The authors of [WA15] do not exploit the depth information in the skin segmentation part. Their skin detection method is only based on color and is inspired by [LK13]. There are also approaches to estimate the model of the human hand, for example [SKR+15, OWL15] propose methods to perform modeling of a single hand in real-time. Although both approaches address the problem by exploiting different machine learning schemes, both of them use the input of a single depth camera. The results presented in both works show the possibilities offered by a dense depth map and explain the trend of exploiting this information for real-time modeling of human hands in hand-only scenarios.

In order to find a method for hand segmentation in RGB-D images the hand modeling approaches in [SKR+15, OWL15] seem to be computationally too expensive. Moreover, both approaches do not account for hand-object interaction and thus do not consider occluded parts of the hand by objects. The approaches in [LK13, KTT+16] are proposed in the context of hand-object interaction and are more suited for a lightweight two-class pixel classification system. Both approaches employ a RF classifier using, at least partly, low level features. Li and Kitani [LK13] propose a sophisticated pixel-level detection scheme using the color information. The evaluation of their method is done on a public data set and on their own data set which consists of three different egocentric videos. The authors propose a feature set consisting of low-level features as well as high-level descriptors such as SIFT (Scale-Invariant Feature Transform). The low-level features comprise the color information of the pixel and its surrounding pixels represented in different color spaces along with textural information obtained by a Gabor filter bank. Kang et al. propose a method to segment the hand in hand-object interaction scenarios by only exploiting depth information [KTT+16]. They collected a ground truth data set containing 25 000 images by wearing a blue glove and thus their ground truth data does not contain color information. Furthermore, their data set consists of simple desktop scenes containing only the object in the hand. However, they do not perform background segmentation through a histogram method of the depth map, as done in [WA15]. Kang et al. use RFs as classifiers for the different configurations, the RFs do have a size of 10 trees and a maximum tree depth of 20. The authors train several single and double stage RF classifiers and analyze those set-ups using five different objects.

Chapter 2

Theoretical Background

This thesis deals with hand segmentation on pixel-level in RGB-D images by assigning a class label to each pixel. There are two class labels in our case. The pixels showing no skin and the pixels showing skin. The corresponding classes are named as non-skin pixels and skin pixels. Therefore, we are dealing with a two-class classification problem. One of the key points for a successful classification scheme is the choice of features. Therefore, we provide the theoretical background of features obtained from different color spaces in this chapter. Additionally, this chapter sheds light on how to extract features representing the textural characteristics of the color image. Further, we briefly discuss the depth information from the RGB-D camera because we include depth features in our approach. Thus, we give a short introduction to SL (Structured Light) depth cameras. Moreover, theoretical background of the chosen classification algorithm RF is given.

2.1 Color Spaces

Visual information is one of the most important features for humans to interact with the world. Especially the color contains useful information about things in sight and eases interaction with the environment. In robotics and computer vision, researchers want to teach computers or machines to interact with the real world the same way as humans do. The technology to capture color images is highly developed. In the digital world the colors in images are mostly represented by the three primary colors red, green and blue which is called RGB color space. However, there are other representations and the most suitable depends on the application. Different color spaces were defined to overcome problems in the RGB color space or to ease pattern recognition by searching for a color representation which creates well separated clusters of different color classes. In skin detection, researchers evaluate the advantages and disadvantages of different color spaces with respect to classification performance. A good color representation for skin detection collects the skin color in easily distinguishable clusters in the color space. Thereby, the classification is simplified and the results tend to be better.

In this work three different color spaces are used: the RGB, the HSV (Hue, Saturation and Value) and the CIELAB color spaces. Whereas, the most common and well-known color space is RGB. Using different color spaces may help to improve the classification results because the number of features is increased and the set of skin colors form better distinguishable clusters. In the following sections the color spaces which are used in this thesis are presented.

2.1.1 RGB Color Space

RGB is the most popular color space. The way colors are generated on display units is directly related to the RGB color space. The colors are represented by an addition of the primary colors red, green and blue; thus, it is called an additive color space. The intensity of each primary color can be adapted independently and results simultaneously in a change of tone, saturation and intensity of the color. The most common way to represent digital images in RGB uses 8 bit per primary color; therefore, 256^3 different colors can be represented which is approximately 16.8 million colors. The colors in the RGB space all lie within a cube because there are three primary colors and all of them have the same range $[0, C_{\max}]$; whereas, $C_{\max}$ describes the maximum intensity of the color. Figure 2.1 sketches the RGB cube for $C_{\max} = 1$. In Tab. A.1 the values of the marked color points in the cube and the intensity values of the primary colors are given. The axes of the system are the primary colors red (R), green (G) and blue (B). The connection line from **S** to **W** via **K** contains the different gray shades [BB09].

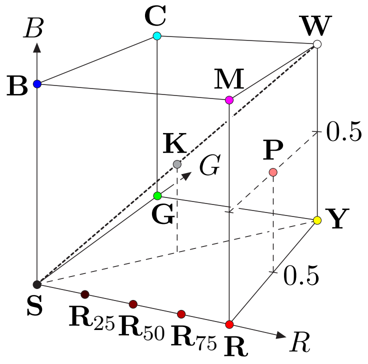


Figure 2.1: The cube sketches the distribution of various colors in the RGB color space [BB09, Chap. 12, p. 310]. The RGB values of the different color points are given in Tab. A.1.

On the one hand, RGB is a very easy understandable representation because it directly relates to the generation in digital displays. On the other hand, the visual perception of the colors does not directly relate to its digital representation. Thereby, a shift of a point inside the space of the RGB color space leads to a very different visual perception of the change in color. Similarly, the change in intensity in the RGB space is not linearly connected to the perceived change. A shift in any direction changes simultaneously the color's tone, saturation and intensity. As a result, manual adjustment of colors in the RGB color space is difficult and not intuitive [BB09].

2.1.2 HSV Color Space

The HSV color space in contrast to the RGB color space represents explicitly color characteristics which are important for subjective perception. As a result, manual adjustment of colors is simplified which makes the HSV color space very popular in image processing applications [BB09]. A change in either H, S or V respectively results in a change either in tone, saturation or intensity.

According to [BB09], the mathematical conversion from RGB to HSV is shown in the following. First, the components $R, G, B \in [0, C_{\text{max}}]$ are used to define the two auxiliary variables

$$C_{\text{high}} = \max(R, G, B) \qquad \text{and} \qquad C_{\text{low}} = \min(R, G, B)\,, \tag{2.1}$$

where R, G and B are the intensities of the primary colors red, green and blue. Second, saturation

$$S_{\text{HSV}} = \begin{cases} \frac{C_{\text{high}} - C_{\text{low}}}{C_{\text{high}}} & \text{if } C_{\text{high}} > 0 \\ 0 & \text{else} \end{cases} \tag{2.2}$$

and intensity

$$V_{\text{HSV}} = \frac{C_{\text{high}}}{C_{\text{max}}} \tag{2.3}$$

are defined. If $(C_{\text{high}} - C_{\text{low}}) = 0$, then the color is a shade of gray $(R = G = B)$, resulting in $S_{\text{HSV}} = 0$. Therefore, the tone of color or hue H_{HSV} is undefined because shades of gray are located on the V_{HSV}-axis. Third, the computation of the tone of color H_{HSV} is done, if $(C_{\text{high}} - C_{\text{low}}) > 0$. The tone of color is defined as

$$H' = \begin{cases} B' - G' & \text{if } R = C_{\text{high}} \\ R' - B' + 2 & \text{if } G = C_{\text{high}} \\ G' - R' + 4 & \text{if } B = C_{\text{high}}\,, \end{cases} \tag{2.4}$$

where the normalized colors are defined as

$$R' = \frac{C_{\text{high}} - R}{C_{\text{high}} - C_{\text{low}}}\,, \quad G' = \frac{C_{\text{high}} - G}{C_{\text{high}} - C_{\text{low}}} \quad \text{and} \quad B' = \frac{C_{\text{high}} - B}{C_{\text{high}} - C_{\text{low}}}\,. \tag{2.5}$$

The values of H' are within the interval $[-1, 5]$; therefore, this value is normalized to obtain

$$H_{\text{HSV}} = \frac{1}{6} \cdot \begin{cases} (H' + 6) & \text{if } H' < 0 \\ H' & \text{else}\,. \end{cases} \tag{2.6}$$

The resulting range of values for the three components $H_{\text{HSV}}, S_{\text{HSV}}$ and V_{HSV} is $[0; 1]$.

The three-dimensional space is shown in Fig. 2.2. As can be seen, the HSV color space describes the colors inside a cylinder. Whereas, the colors on the V-axis are the different shades of gray with black (**S**) located in the origin and white (**W**) on the opposite side of the cylinder. The saturation of the color can be adjusted along the S-axis. The tone of color is changed by adapting the H-value which corresponds to a rotation around the V-axis. For comparison of RGB and HSV color space, the indicated color points in Fig. 2.2 and Fig. 2.1 describe the same color points, except for **K**. The values of the indicated color points are given in Tab. A.1.

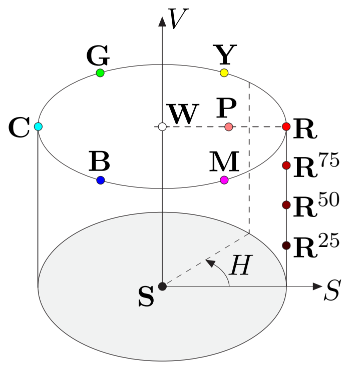


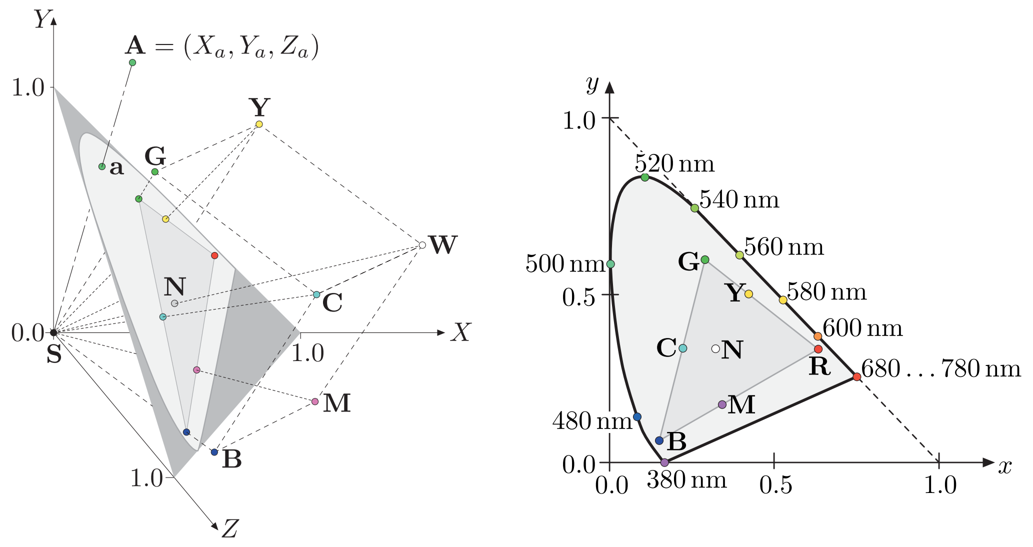
Figure 2.2: The cylinder sketches the distribution of various colors in the HSV color space [BB09, Chap. 12, p. 326]. The HSV values of the different color points are given in Tab. A.1.

2.1.3 CIELAB Color Space

In 1931 the CIEXYZ color system was standardized by the CIE (International Commission on Illumination)[1] and forms a basis for almost all colorimetric color spaces, such as the CIELAB color space. First, the CIEXYZ color space is introduced, because the transformation of CIELAB is done from the CIEXYZ color space. The imaginary primary colors X, Y, and Z of the CIEXYZ color space are chosen, such that all visible colors lie within the subspace of the positive axes. The space of the visible colors looks like a sugarloaf with its tip in the origin of the coordinate system. However, the imaginary primary colors do not lie within the visible colors. The RGB color space can be linearly transformed into CIEXYZ. Constant relative

[1] Commission Internationale d'Eclairage

changes in the coordinates, of both RGB and CIEXYZ space, lead to a non-linear change in the perceived color [BB09].



(a) CIEXYZ color space (b) CIE chromaticity diagram

Figure 2.3: Diagram of CIEXYZ color space and the corresponding chromaticity diagram [BB09, Chap. 14, p. 366]. The values of the marked color points are given in Tab. A.2.

In Fig. 2.3(a) the distorted RGB cube is shown inside the CIEXYZ color space. Whereas, the cube's black corner is located in the origin of the XYZ coordinate system. The plane $X + Y + Z = 1$ inside the color space is called chromaticity diagram and is usually drawn in the projection onto the x, y-plane, as can be seen in Fig. 2.3(b). This diagram is usually used to simplify the representation of the CIEXYZ color space. Thus, each color point $\mathbf{A} = (X_a, Y_a, Z_a)$ is represented with its chromaticity value $\mathbf{a} = (x_a, y_a, z_a)$. Whereas, $\mathbf{a}$ describes the intersection of the line $\overline{\mathbf{SA}}$ with the plane $X + Y + Z = 1$. The representable physical colors of the RGB color space are shown inside the triangle of the chromaticity diagram in Fig. 2.3(b). The monochromatic colors are located along the edge of the chromaticity diagram. The corresponding wavelength of the monochromatic colors from purple to red (380 nm to 780 nm) can be seen in the figure as well. The color values of denoted color points in Fig. 2.3 are given in Tab. A.2. The color point **N** describes the neutral point which does not contain any color; thus, contains the projection of all different shades of gray from black to white [BB09].

As mentioned above, the problem of the CIEXYZ color space is that a constant shift of color points in the space causes a differently perceived change of color. The perceived color change may be larger or smaller compared to other colors, depending on the color shifted. Therefore, researchers define a color space, called CIELAB or CIEL*a*b* which matches coordinate changes to perceived color changes. As a result a constant shift of any color point in CIELAB results in a constant perceived color change. However, CIELAB is an approximation and does not achieve this requirement perfectly [HI16, BB09].

In the following the conversion policy from RGB to CIELAB is given, according to [BB09, Poy97]. First, the RGB color has to be transformed into the CIEXYZ color space. This is achieved by a linear coordinate transformation. The matrix M_{RGB} describes the transformation from RGB-values into XYZ-values.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = M_{\mathrm{RGB}} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.7}$$

Next, the XYZ-values are converted into the LAB-values. The auxiliary function

$$f_1(c) = \begin{cases} c^{1/3} & \text{if } c > \epsilon \\ \frac{\kappa \cdot c + 16}{116} & \text{if } c \leq \epsilon \end{cases} \tag{2.8}$$

is defined to simplify the notation, where $\epsilon \approx 0.008856$ and $\kappa = \left(\frac{29}{3}\right)^3$. The color values of CIELAB are computed by

$$\begin{aligned} L^* &= 116 \cdot f_1\left(\frac{X}{X_{\mathrm{ref}}}\right), \\ a^* &= 500 \cdot \left(f_1\left(\frac{X}{X_{\mathrm{ref}}}\right) - f_1\left(\frac{Y}{Y_{\mathrm{ref}}}\right)\right) \quad \text{and} \\ b^* &= 200 \cdot \left(f_1\left(\frac{Y}{Y_{\mathrm{ref}}}\right) - f_1\left(\frac{Z}{Z_{\mathrm{ref}}}\right)\right), \end{aligned} \tag{2.9}$$

where the asterisk (*) indicates that the values have perceptual properties. Furthermore, L^* is called the lightness, a^* spans the range of red-green colors and b^* spans the range of yellow-blue colors. The variables $X_{\mathrm{ref}}, Y_{\mathrm{ref}}$ and Z_{ref} denote the reference white point in the CIEXYZ color space. As a reference point for the transformation the point D_{65} ($[X_{\mathrm{ref}}, Y_{\mathrm{ref}}, Z_{\mathrm{ref}}] = [0.95047, 1.0, 1.08883]$) is usually chosen.

2.2 Image Texture

In pattern recognition for vision tasks, researchers look for ways to extract features from images or videos in a way humans do. Visual information is processed amongst others, in the visual cortex of mammalian brains. Therefore, the visual cortex system has been analyzed by researchers in order to understand the processing of the visual information collected by the eyes in the mammalian brain. The cells processing the information are called cortical cells. In the 1980s, Marčelja [Mar80] and Daugman [Dau85] compared in their publications the response of simple cortical cells with the response of so-called Gabor filters. Marčelja focussed solely on the one-dimensional case. Daugman on the other hand extended the analysis to the two-dimensional case and discusses the choice of filter parameters regarding the cell characteristics. Both scientists conclude that responses of cortical cells can be modelled well with the Gabor filters.

In the following, we want to give some background information and theory for Gabor filters. In 1946, the Hungarian engineer Dennis Gábor published a new method to analyze information containing signals in communication. He focusses on finding a more suitable representation for a signal which contains both time and frequency variables. His goal is to maximize the information transmitted in a given time over a restricted channel. An information containing signal is characterized by changing both in time and in frequency. Signals of limited duration can only be analyzed in the frequency domain with a certain inaccuracy and vice versa. The inaccuracy for example is inversely proportional to the duration. From that, Gábor deduced constraints which have to be fulfilled in order to minimize the uncertainty in both time and frequency [Gab46]. Gábor's transformation function can be adapted to signal analysis to establish a filter function, called the Gabor filter [KKK06]. In 1978 Granlund [Gra78] proposed a method to describe pictorial features. Surprisingly, Granlund's result corresponds to the Gabor filters in 2-D. The notation we use is according to the OpenCV 2.4 implementation, for comparison see [Gab46, KKK04, KKK06]. The two-dimensional Gabor filter function can be written as

$$\begin{aligned} h[x,y] &= \exp\left(-\frac{1}{2\sigma^2}\left(x'^2+\gamma^2 y'^2\right)\right)\,\exp\left(\mathrm{j}\frac{2\pi}{\lambda}x'\right)\,, \\ x' &= x\cos\theta + y\sin\theta\,, \\ y' &= -x\sin\theta + y\cos\theta\,, \end{aligned} \tag{2.10}$$

where $x, y \in [-L, -L+1, \ldots, L-1, L]$ and $L \in \mathbb{N}^*$. Furthermore, λ denotes the wavelength of the central spatial frequency, θ is the orientation of the major axes of the Gaussian, σ describes the smoothing parameter along the major axis and γ is the spatial aspect ratio of the Gaussian envelope. The minor and major axes are perpendicular to each other. The definition in (2.10) does not include a normalization constant. However, the inclusion of a normalization constant is not necessary in our case.

In the following the general shape and the effects of the design parameters are shown. The Gabor filters are complex filters and the real and imaginary part of a Gabor filter kernel are shown in Fig. 2.4. On the other hand in Fig. 2.5 the real part of the kernel and the magnitude of the filter kernel in the frequency domain are shown. One can see that the frequency response of the Gaussian envelope is shifted in the frequency domain. The shift corresponds to the frequency of the complex exponential function. Figure 2.5 is used as a reference to show the influence of the design parameters θ, σ, λ and γ. The complex exponential can be imagined as a transverse wave; therefore, let us denote its direction of propagation as the direction of the kernel. In the case of $\theta = 0$ the direction of the kernel is the same as the x-axis.

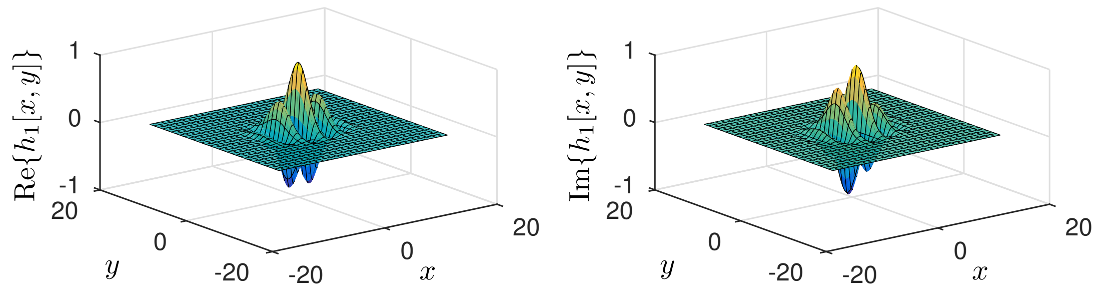


Figure 2.4: Real (l) and imaginary part (r) of the reference Gabor filter kernel. ($\theta = 0°$, $\sigma = 2.8$, $\lambda = 3.5$, $\gamma = 1.25$)

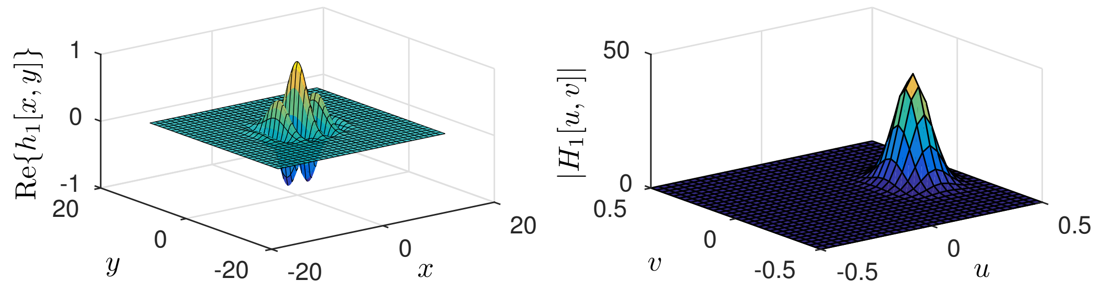


Figure 2.5: Real part (l) and magnitude of the frequency response (r) of the reference Gabor kernel. ($\theta = 0°$, $\sigma = 2.8$, $\lambda = 3.5$, $\gamma = 1.25$)

The influence of the orientation θ is as expected and rotates the kernel around the filter center. The angle between the direction of the filter kernel and the x-axis is described by θ. Similarly, the frequency response of the filter is rotated around the origin, as well by the angle θ. Figure 2.6 shows the real part and the magnitude of the frequency response of the rotated version of the reference kernel by 135°.

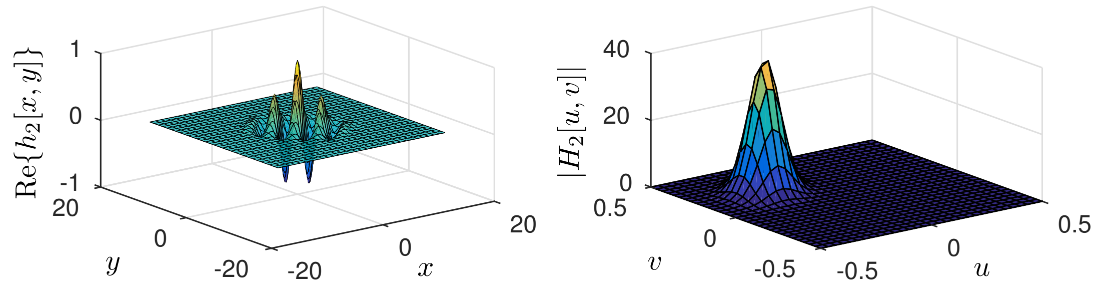


Figure 2.6: Real part (l) and magnitude of the frequency response (r) of the rotated Gabor kernel. ($\theta = 135°$, $\sigma = 2.8$, $\lambda = 3.5$, $\gamma = 1.25$)

The smoothing parameter σ varies the sharpness of the Gaussian envelope. Increasing σ leads to a wider and less sharp impulse response while simultaneously making the frequency response of the kernel sharper. This means increasing the certainty in one domain increases the uncertainty in the other domain. Figure 2.7 shows the changes for an approximately doubled smoothing parameter.

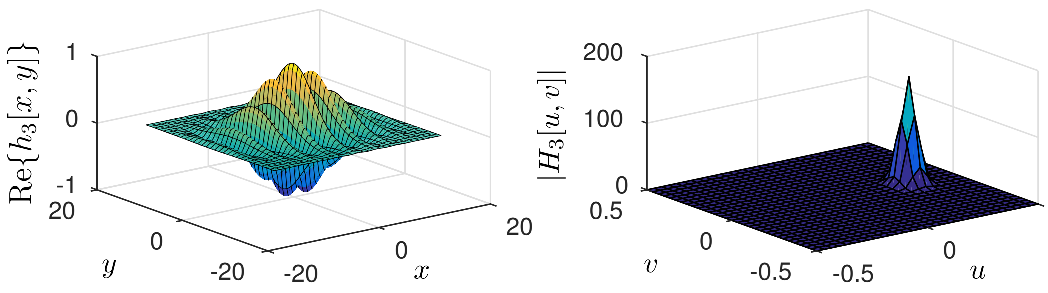


Figure 2.7: Real part (l) and magnitude of the frequency response (r) of a Gabor kernel with higher smoothing. ($\theta = 0°$, $\sigma = 5.7$, $\lambda = 3.5$, $\gamma = 1.25$)

A variation of the wavelength shifts the frequency response along the direction of the kernel in the frequency domain. The reduction of oscillations in the spatial domain shifts the frequency response towards the origin in the frequency domain. Similarly, for an increase of oscillations the frequency response is shifted towards the higher frequencies. The result of increasing the wavelength compared to the reference kernel in Fig. 2.5 can be seen in Fig. 2.8.

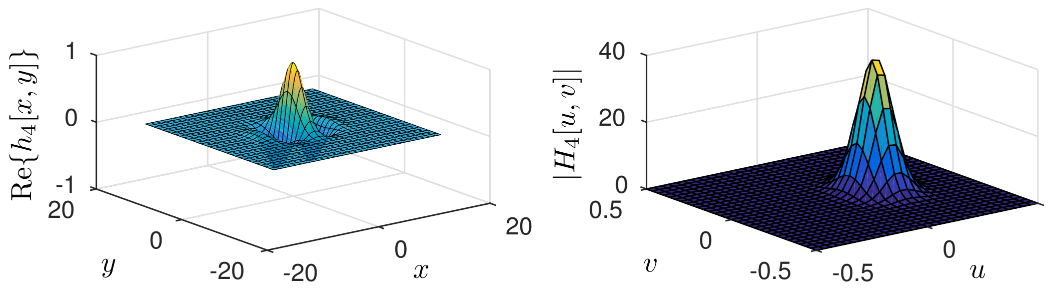


Figure 2.8: Real part (l) and magnitude of the frequency response (r) of a Gabor kernel with a different wavelength. ($\theta = 0°$, $\sigma = 2.8$, $\lambda = 7$, $\gamma = 1.25$)

Changing the spatial aspect ratio to a value $\gamma \neq 1$ changes the contour lines of the Gaussian envelope from circles to ellipses. The major axis of the Gaussian envelope in the spatial domain is identical to the direction of the kernel, for $\gamma > 1$. Whereas, for $0 < \gamma < 1$, the minor axis of the envelope and the direction of the kernel are identical. In Fig. 2.9 the influence for an increased spatial aspect ratio can be seen. In particular the envelope of the frequency response is sharper along the major axis in the spatial domain. This means the frequency selectivity is higher in the direction of the major axis in the spatial domain.

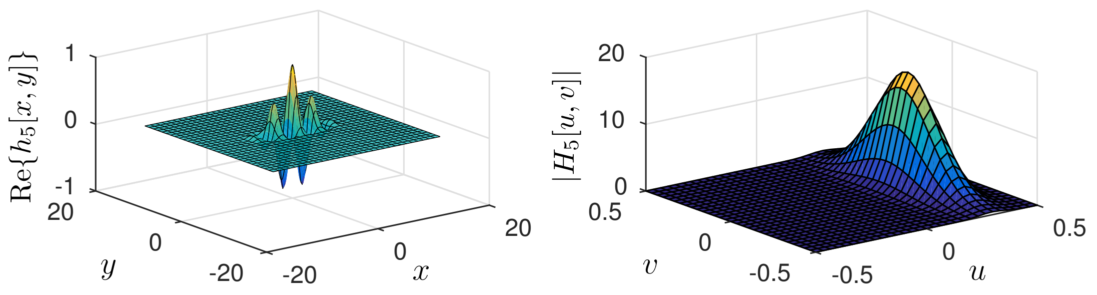


Figure 2.9: Real part (l) and magnitude of the frequency response (r) of a Gabor kernel with a higher spatial aspect ratio. ($\theta = 0°$, $\sigma = 2.8$, $\lambda = 3.5$, $\gamma = 3$)

In order to extract discriminative features from the images a proper filter design is needed. Furthermore, exploiting the information of a single kernel will not extract the entire information. Therefore, most researchers apply a Gabor filter bank on the input image. Intuitively one would design the filter bank in a way that the combination of filters covers the entire frequency domain from 0° to 360°. However, the area from 180° to 360° does not give additional information for real valued input signals. The derivation for this property is given in App. B. Thus, the filter bank is designed such that the area in frequency domain from 0° to 180° is covered. The design of the filter bank is given in Chap. 3.2.1.

Gabor filters have emerged to one of the leading approaches to extract texture information from images in order to classify different types of textures [BF07]. The reason for this appears to be the similarity of the impulse response of a Gabor filter kernel compared to that of cortical cells in mammalian eyes. As well as, the optimality of Gabor filters in the spatial and frequency domain [KKK04].

2.3 Structured Light Depth Camera

There are passive and active methods to compute the depth map of a scene. A stereo camera system is a passive system and similar to the mammalian visual system because two cameras are used to estimate the depth. The estimation is done by finding point correspondences in both images which is computationally expensive and they usually fail on unicolored scenes. On the other hand, active methods are used in ToF (Time-of-Flight) and SL depth cameras. The working principle of a ToF depth camera is the RADAR (Radio Detection And Ranging) principle, while SL depth cameras apply computational triangulation methods to estimate the disparity or depth. This section discusses only SL depth cameras because a SL depth camera is used as sensor for our approach. SL depth cameras use a light projector to illuminate the scene with a known light pattern, note that the light pattern must not be a diffuse light source. Usually, an infrared light projector is used because the human eye cannot perceive such light patterns. In the following a short introduction to the working principle is given. Furthermore, the problems or non-idealities are discussed. Zanuttigh et al. give a more detailed introduction to SL cameras, while we restrict ourselves to a short summary of [ZMDM+16].

2.3.1 Working Principle

SL depth cameras apply triangulation methods as in passive stereo systems. The name of SL depth cameras implies the application of a light projector which illuminates the scene with a known pattern. The *Primesense* design, applied in *Microsoft*'s *Kinect* and *Asus*' *Xtion*, consists of two cameras and a projector. The projector uses infrared light to project the pattern onto the scene. One of the cameras is a dedicated infrared camera and the other is a standard RGB camera. All three components are calibrated in order to align the estimated depth values with the RGB image. Zanuttigh et al. describe the calibration process in detail. In the following we consider the interaction between projector and infrared camera, both together form a rectified stereo system.

Figure 2.10 shows a schematic of a rectified stereo system to illustrate the estimation of the depth. However, it is a simplified version. The image frame of the projector is displayed on the right, while the camera frame is shown on the left. The optical centers are indicated by $\mathbf{c}_{\text{cam}}$ for the camera and $\mathbf{c}_{\text{proj}}$ for the projector. The correspondences of the pattern can be found in both images. In a rectified stereo system the disparity can be estimated by the column positions in both images. Assume the points $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ represent three different scenes with different depth values. The blue dots in the camera frame indicate, how the correspondence appears in a different pixel for each situation. The shift is only present in horizontal direction due to the geometrical structure of the system.

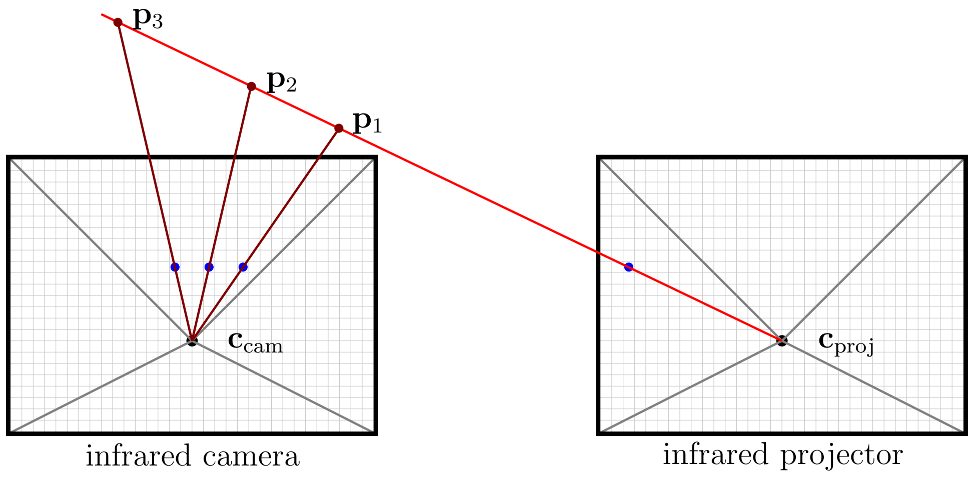


Figure 2.10: Illustration of the depth estimation process in SL depth cameras. The figure shows how the point correspondences change for different depths.

The uniqueness of the projected light pattern in a SL depth camera can be designed by the four different signal multiplexing methods wavelength, range, temporal and spatial multiplexing. The SL depth cameras with the *Primesense* design use spatial multiplexing. In spatial multiplexing the same pattern is projected onto the scene at each time. The pattern's uniqueness ensures that finding the point correspondences is easy but robust. The correspondences are used to estimate the depth of the space points visible in the image.

2.3.2 Problems and Non-Idealities

In this section the problems of SL depth cameras are described. The geometrical set-up of the cameras and the non-idealities of camera and projector cause limitations of the system.

The result of a SL depth camera is influenced by perspective distortions of the projected light pattern. The camera captures a distorted version of the projected light pattern in regions of slanted or curved surfaces. The projector and camera themselves are not ideal imaging systems; thus, the captured light intensity is not equivalent to the emitted intensity of the projector.

Depending on the scene the corresponding ideal depth map contains discontinuities which can cause invalid depth measurements in the estimated depth map. In this context invalid means, that the depth at the respective position cannot be estimated. These invalid depth measurements are caused by occlusions. In particular, the projectors light pattern is not projected on the surface which is captured by the infrared camera, because the perspective onto the scene is different for infrared projector and infrared camera. Therefore, the camera can perceive certain areas of the scene, while the projector cannot illuminate them and vice versa. The problem is shown in Fig. 2.11 in the first row. The images from left to right are RGB image, infrared image and depth map. The black pixels in the infrared image show areas not illuminated by the projector.

Other reasons for invalid depth measurements are grey-level distortions and external illumination. Grey-level distortions are caused by certain reflectance properties of the surface materials. The color of an object influences the absorption of light for example black objects absorb more light than white objects. Therefore, the intensity of the infrared pattern reflected from black objects is generally lower as from white objects. Figure 2.11 shows this problem in the second row, the intensity of the reflected infrared pattern depends on the scene's color. In addition, external illumination such as light bulbs or natural light, act as noise on the projected light pattern because the captured light is the sum of reflected light from external illumination and projected light pattern. This may lead to clipping of intensity values in the captured infrared image, due to saturation effects, in certain scene areas and cause invalid depth measurements.

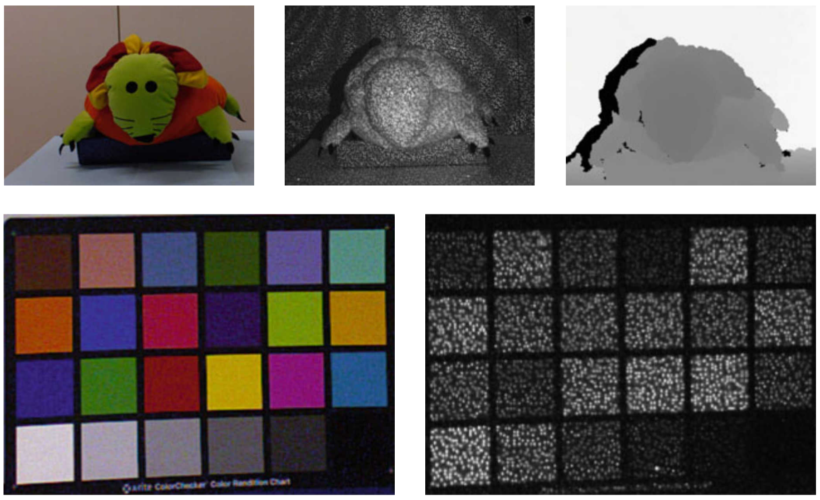


Figure 2.11: The figure shows illustrations of two problems occurring with SL depth cameras [ZMDM$^+$16, Chap. 2, p. 66]. The first row shows the reason for invalid pixels due to occlusions and the second row show grey-level distortions. The first row shows from left to right the RGB image, the infrared image and the depth map. In the second row displays the RGB image on the left and the infrared image on the right.

2.4 Classification Algorithm

In this thesis we use the machine learning algorithm RF (Random Forest). As the name implies, the algorithm consists of a set of decision trees forming a forest. Breiman et al. profoundly deal with classification and regression trees in their book [BFSO84] which is one of the earliest works on decision trees and simultaneously considered as very worthwhile in the community. Amid and Geman [AG94, AG97] have been the first using a set of different decision trees to form an ensemble. Breiman has been inspired by their work and proposes a new procedure to introduce randomness into the trees [Bre01]. Recently, RFs have been getting popular because they have proven to provide a good estimation of the true result, even to unseen data. This characteristic is called generalization.

This section deals with decision trees and ensembles of them. However, we restrict this section to the special case of two-class classification problems, because the proposed approach in this thesis is a two-class classification scenario. Section 2.4.1 provides the mathematical background to decision trees. Further, Sec. 2.4.2 describes the generalization to randomized ensembles of decision trees, called RF.

2.4.1 Binary Decision Tree

In this section a short theoretical introduction to binary decision trees [BFSO84, CS13] is provided. As the name implies, a decision tree contains various decisions.

In the case of binary decision trees each decision has one of two outcomes. Once all decisions have been made a class is assigned to the data point, based on all outcomes. The decision tree uses a set of features extracted from the raw data point to make the decisions. In computer vision the features are extracted from images or videos which form the raw data set. Thus, the features from the raw data have to be extracted in advance of training or classification. In the context of pixel-level classification the data to be classified is for example the image or video and the data point describes a single pixel.

Definitions

In Sec. 2.4 we define the images used for training and classification as raw data set. Consequently, each raw data point describes a single pixel of the image. Further, a data point describes the set of features of a pixel. Accordingly, a data set is the set of data points.

In general, a decision tree is a graph with a hierarchical structure and may not contain loops, an example is shown in Fig. 2.12. In binary decision trees, there are two types of nodes: the internal nodes and the leaf nodes. The internal node at the highest hierarchical level is called root node and the other internal nodes are called split nodes. In Fig. 2.12 the internal nodes are drawn as circles, while squares indicate leaf nodes. Furthermore, the decision tree is executed in a top down manner, starting at the root node and finishing at the leaf node. Each internal node has two children, a left and a right one, for example the root node's left child is node 1 and the right child is node 2.

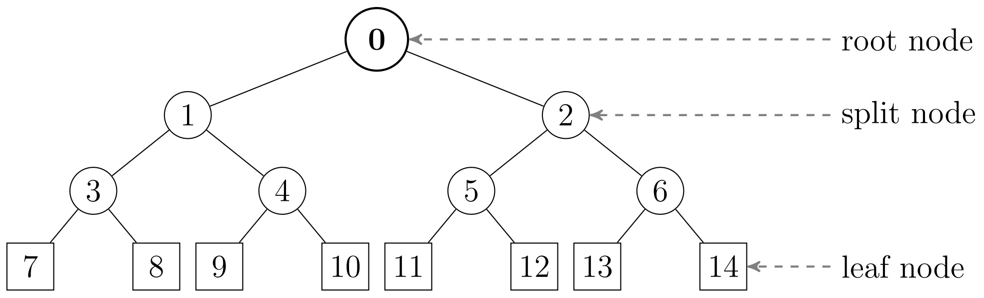


Figure 2.12: General structure of a decision tree. According to [CS13, Chap. 3, p. 9].

The data points describe a set of features or characteristics which are described as the feature vector

$$\mathbf{v} = (v_1, v_2, \ldots, v_d)\,, \tag{2.11}$$

where $v_i \in \mathbb{R}$ represents the value of the features and d is the dimension of the vector. The dimension d is equal to the number of features used for classification and training. In most cases d is a large number and calculating an optimal split function for the entire feature set is not feasible. Thus, each internal node selects a small subset of all features. In our work only one feature is selected at each node. This selector function is formulated as

$$\boldsymbol{\phi}(\mathbf{v}) = x_\phi \in \mathbb{R}\,, \tag{2.12}$$

where $\phi \in [1; d]$ describes the number of the feature selected. The selector (2.12) is different for each internal node.

Each internal node is described by a model of a split function. In this chapter we restrict ourselves to split functions in only one dimension because that is the type of split functions we use in the experiments in Sec. 4. The split function selects a feature with (2.12) and then performs a split on that feature based on a threshold operation. The parameters of the model are described as $\boldsymbol{\theta} = (\boldsymbol{\phi}(\mathbf{v}), \tau)$, where $\boldsymbol{\phi}(\mathbf{v})$ is the selector from (2.12) and τ describes the threshold of the inequality. Therefore, the split function with binary output is formulated as

$$h(\mathbf{v}, \boldsymbol{\theta}) = [\,\boldsymbol{\phi}(\mathbf{v}) > \tau\,]\,, \tag{2.13}$$

where $[\,\cdot\,]$ returns 0 if the inequality is false and 1 if it is true. In [CS13] more general definitions for linear split functions as well as non-linear split functions can be found.

Tree training is performed in a supervised manner; thus, a labeled raw data set is needed. The data set used for training is called the training set and similarly, the data points are called training points. A training point is defined as a pair $(\mathbf{v}, c)$ containing the feature vector $\mathbf{v}$ and the class label c. In this thesis we only use two classes and therefore, $c \in [0, 1]$. In order to simplify the notation we define $\mathcal{S}_j$ as the set of training points which are available at the jth split node. Furthermore, $\mathcal{S}_j^{\mathrm{L}}$ and $\mathcal{S}_j^{\mathrm{R}}$ denote the children of split node j. The naming of the nodes is done in a breadth-first order which starts at 0 for the root node. This definition of notation results in the following properties for binary trees [CS13]:

$$\mathcal{S}_j = \mathcal{S}_j^{\mathrm{L}} \cup \mathcal{S}_j^{\mathrm{R}}\,, \quad \mathcal{S}_j^{\mathrm{L}} \cap \mathcal{S}_j^{\mathrm{R}} = \emptyset\,, \quad \mathcal{S}_j^{\mathrm{L}} = \mathcal{S}_{2j+1}\,, \quad \mathcal{S}_j^{\mathrm{R}} = \mathcal{S}_{2j+2} \tag{2.14}$$

The training sets $\mathcal{S}_j^{\mathrm{L}}$ and $\mathcal{S}_j^{\mathrm{R}}$ are deduced from the set $\mathcal{S}_j$ by

$$\begin{aligned} \mathcal{S}_j^{\mathrm{L}}(\mathcal{S}_j, \boldsymbol{\theta}) &= \{(\mathbf{v}, c) \in \mathcal{S}_j \mid h(\mathbf{v}, \boldsymbol{\theta}) = 0\}\,, \\ \mathcal{S}_j^{\mathrm{R}}(\mathcal{S}_j, \boldsymbol{\theta}) &= \{(\mathbf{v}, c) \in \mathcal{S}_j \mid h(\mathbf{v}, \boldsymbol{\theta}) = 1\}\,, \end{aligned} \tag{2.15}$$

where c describes the class label.

Classification

The classification process for a two class classification problem with a binary decision tree and split functions along a single dimension of the feature space is described in this section.

The raw data point to be classified is represented by its characteristics via the feature vector $\mathbf{v}$, defined in (2.11). This data point is injected into the decision tree at the root node, where the split function $h(\mathbf{v}, \boldsymbol{\theta}_0)$ directs the data point to the left or right child of the root node depending on the outcome of the function. Assuming that the data point is directed to node j, then the next node is determined by the split function $h(\mathbf{v}, \boldsymbol{\theta}_j)$ of node j. This is done until the data point reaches a leaf node. The leaf node gives an estimation of the class the data point belongs to. In case of classification, only the nodes are processed which are reached by the data point [CS13].

In Fig. 2.13 an example of a binary decision tree is shown. The decision tree is a very simplified version to find out if one needs to buy groceries. Starting from the top and following the branches according to the answer one ends up in one of the leaf nodes. The leaf nodes indicate what action has to be done: either buying groceries or not buying groceries. In this case the data point refers to the current situation and observations, the questions at each internal node represent the split rules and the actions describe the class.

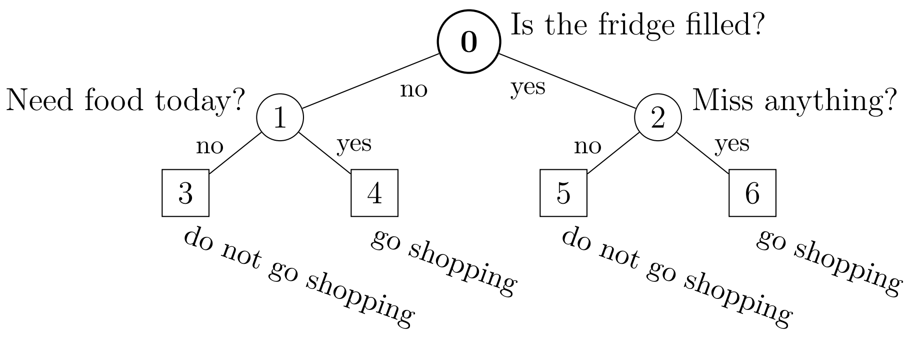


Figure 2.13: Example of a decision tree to find out if one need to buy groceries.

Tree Training

The problem illustrated in Fig. 2.13 is easy to understand because humans have a good understanding of the process. However, if the problem is more complex or if there is no experience with the problem, finding a well functioning tree is very difficult. Similarly, for large numbers of characteristics there are too many possibilities and it is not obvious to find the best split rules. Nowadays, creating a decision tree is done by machine learning techniques. These techniques require a representative training set and deduce suitable split rules based on a statistical analysis of the training set. In the following, the technique to train random decision trees is presented where each tree in the RF is trained randomly.

The key element of the training process is the understanding how the algorithm computes the best split. The computation of the split model parameters is an optimization problem which is solved by means of an energy model. There are different definitions of energy models available, more details can be found in [BFSO84]. In order to train a binary decision tree for classification problems amongst others the commonly known metrics entropy and information gain can be used. We chose entropy and information gain to illustrate the optimization problem because these metrics are well understood. The definition of the Shannon entropy for the training set $\mathcal{S}$ is

$$H(\mathcal{S}) = -\sum_{c\in\mathcal{C}} p(c) \log\left(p(c)\right) , \tag{2.16}$$

where $\mathcal{C}$ is the set of class labels and c denotes the label. The empirical distribution of training points in $\mathcal{S}$ is described by $p(c)$. Consequently, $p(c)$ is the relative amount of classes within the training set $\mathcal{S}$. The information gain I describes the energy model and is given as

$$I = H(\mathcal{S}) - \sum_{i\in\{\mathrm{L},\mathrm{R}\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i) , \tag{2.17}$$

where $|\cdot|$ describes the cardinality of the set. The training phase aims at finding the best parameters of the split rule by maximizing the information gain. Thereby, the uncertainty of the classes in the training sets $\mathcal{S}_j^{\mathrm{L}}$ and $\mathcal{S}_j^{\mathrm{R}}$ is reduced. As defined in (2.13) the parameter set $\boldsymbol{\theta}_j$ describes the split rule or test function at node j. Thus, the optimization problem is formulated as

$$\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta}\in\mathcal{T}} \{I(\mathcal{S}_j, \boldsymbol{\theta})\} , \tag{2.18}$$

where $\mathcal{T}$ describes the set of all possible parameters $\boldsymbol{\theta}$. The split of the training set into the subset of the left and right child is done with (2.15). The parameters $\boldsymbol{\theta}$ which split the training data best are assigned to node being optimized.

The entire training set appears only in the root node. The cardinality of the training set $\mathcal{S}_j$ decreases towards the leaf nodes of the tree. This means the training set of node j is larger than of node $2j + 1$ and $2j + 2$. Therefore, it makes sense to stop the training if a certain number of training points fall below a defined threshold. In this work three termination conditions to stop the growing of the decision tree are used:

- Maximum tree depth
- Minimum number of training points required at a node to perform a split
- All data points in a node belong to the same class

If none of the aforementioned conditions is fulfilled, then the training is continued and if one or more conditions are fulfilled, then the training is stopped and the node becomes a leaf node. The characteristic of a leaf node is that it does not split the training points anymore and thus, the class represented most in the training points $\mathcal{S}_j$ can be assigned to it.

2.4.2 Random Forest

A RF (Random Forest) is an ensemble of decision trees which are trained in a randomized manner. In this section the design parameters of a RF, the specialties of a randomized forest training and the variable importance metric is explained.

The tree growing termination criteria and the number of trees inside the forest are important design parameters. The termination criteria for tree growing were discussed in the previous section. Here, we focus on the size of the forest. Fortunately, the number of trees is not a problematic parameter. Breiman [Bre01] shows that the number of trees does not impose overfitting of the RF. According to Breiman, more trees inside the RF do not worsen the classification result but adding more trees does not improve the classification result. As a result finding a reasonable number of trees is more important with regard to efficiency of the classifier.

Randomized Forest Training

In a RF every tree is trained separately but there are differences in the tree training compared to the previous section. These differences can be called randomization of the training process. Again, we restrict the description in order to match the training process used in Chap. 4. The randomness is introduced into the RF training with the two following methods:

- Bagging and
- RNO (Random Node Optimization).

Both methods can be applied on their own as well as in combination. The introduction of randomness into each tree leads to different trees which is favorable. Obviously, training several trees without randomization will result in the same tree and thus, will not improve the classification result or generalization property. Randomizing the tree training results in distinct trees which will lead to different estimations helping to improve the generalization.

Bagging is the method of training each tree with a different subset of the training set. Whereas, the subset is randomly drawn from the entire training set. Let us define this subset as $\mathcal{S}_0^l$, where l denotes the index of the currently trained tree. According to [CS13], Bagging has two advantages. First, Bagging has shown to improve the generalization property of the forest. Second, using a smaller subset of the entire training set reduces the training time. The subset $\mathcal{S} \setminus \mathcal{S}_0^l$ is called the OOB (Out-of-Bag) data, which can be used for an OOB error estimate or for calculating the variable importance.

RNO describes the technique to randomize the parameters of the split rules. The extent of randomization in the split rules is part of the forest design. In [CS13] a more general discussion can be found; however, we will focus on the type of RNO used in the experiments in Chap. 4. From the mathematical point of view the optimization problem (2.18) is adopted to

$$\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}_j} \{ I(\mathcal{S}_j, \boldsymbol{\theta}) \} , \tag{2.19}$$

where $\mathcal{T}_j$ is the set of randomly chosen split rule parameters used in the optimization process at node j. In our case, for one dimensional test functions (2.13) there are two possibilities to randomize the node optimization. First the feature number to optimize on can be randomly drawn. Second the threshold values can be drawn randomly as well. In our training scheme we randomize the feature number but search for the optimal threshold value. At each node a set of selector functions (2.12) are randomly drawn and for each selector the optimal split is computed. At the end the configuration with the best split performance is chosen and assigned to node j.

Variable Importance

The definition used to compute the variable importance in some of the experiments in Chap. 4 is described in the following. This measure reflects the power of the feature to change the classification outcome. The computation of the variable importance is performed with the OOB data. The variable importance of the tree is computed as soon as the training process for a tree has finished. The procedure is as follows.

1. The OOB data is used to determine the number of correct classification results n^{correct} of the OOB data set for tree number l.
2. The features are randomly permuted. The permutation process is executed on each feature dimension separately. The values of feature m in the OOB data set are randomly shuffled and used to determine the number of correct classification results n_m^{correct}.
3. The raw variable importance is given by the difference $w_{m,l}^{\text{imp}} = n^{\text{correct}} - n_m^{\text{correct}}$, where $\mathbf{w}^{\text{imp}} = [w_{1,l}^{\text{imp}}, w_{2,l}^{\text{imp}}, \ldots, w_{229,l}^{\text{imp}}]$.

This procedure is executed for each feature m of tree l and the result is stored in vector $\mathbf{w}_l^{\text{imp}} \in \mathbb{R}^d$. The variable importance of the forest is then calculated by

$$\mathbf{w}^{\text{imp}} = \frac{\sum_{l \in \mathcal{L}} \mathbf{w}_l^{\text{imp}}}{\| \sum_{l \in \mathcal{L}} \mathbf{w}_l^{\text{imp}} \|_1} , \tag{2.20}$$

where $\mathcal{L}$ denotes the set of trees available in the forest.

Chapter 3

Technical Approach

In Chap. 2 the theoretical background of different color spaces, the Gabor filter, depth maps and RFs are described. Consequently, this chapter explains the technical details of the chosen approach in this thesis. First, Sec. 3.1 deals with the collection of two labeled data sets which are used for training and verification of the RFs. Second, in Sec. 3.2 provides background to the arrangement and reasoning behind the used feature set. Next, in Sec. 3.3 the parameterization of the RF is discussed. Finally, Sec. 3.4 shows the scheme for RoI extraction without available hand position.

3.1 Ground Truth Estimation

The RF requires a training phase in order to adapt to the statistical properties of the application. In case of supervised training, a data set must be available which contains both the data used to infer the result and the result itself. Often such a training data set is referred to as the ground truth. The approach chosen combines parts of the approaches in [LK13, KTT$^+$16], meaning both color and depth information are used. This poses a problem because, to the best of our knowledge, there is no available labeled data set combining both color and depth information of an ego-centric camera perspective showing hand-object interactions. The scientists Kang et al. used a colored glove to label the data [KTT$^+$16] which is not possible in this approach because using a colored glove removes all color information of the hand. Considering these implications, a new data set has to be collected and labeled. A vast amount of data is needed for training and thus it is not possible to manually label the entire data set. Therefore, to collect the data in a feasible amount of time a new concept of data collection is proposed. The collected training data set consists of 3420 images showing 38 different desktop scenes. Additionally, a verification data set comprising 922 images from 16 different desktop scenes is collected. As the names imply the training data set is used for training of the RF and for finding the hyper parameters. In contrast, the verification data set is only used to test the performance of the trained classifiers.

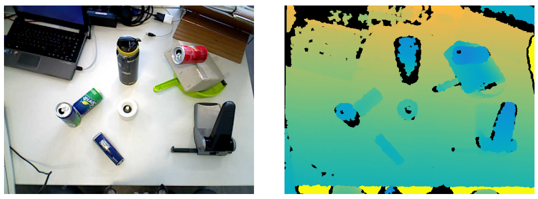
3.1.1 Collection Concept

The data set is divided into sessions with each session being a subset of the entire data set. Each session contains a reference frame and training frames. Specifically, a reference frame contains both a color image and a depth map showing the scene without hand. The training frames on the other hand consist of the color image, depth image and the generated mask. Note that only the scenes of training frames show the hand. The data set contains only frames and not the entire video file, in order to reduce the amount of data. Moreover, this ensures having a lot of highly varying scenes in the data set.

A central assumption in this work is an ego-centric camera perspective involving actions of the right hand. The chosen approach to collect data exploit the difference in the depth map of training and reference frame. Thus, the following assumptions or restrictions are made:

- The images are recorded with an ego-centric camera perspective. Furthermore, the camera position remains static for each session.
- The desktop scene has to be kept static for each session, i.e. all objects in the scene remain at the same position and must not move.
- The scene shows a general desktop scene with randomly arranged objects on the desk but the scene changes for each session. Figure 3.1(a) shows one of the captured desktop scenes.

In the following the process of collecting data is described. Again, the desktop scene changes from session to session; thus, there is no fixed set of objects. However, some objects appear more frequently in the data set and some have been covered with skin-like prints, more detail is given in Sec. 3.1.2. During each session roughly 40 to 120 training frames are taken. We do not want to restrict the set-up to hand-object interaction. Therefore, approximately half of the frames show the hand without object interaction while the other half does. In this case hand-object interaction means the objects placed on the desktop are grasped in different ways without moving the object. The data set contains a huge variety of different grasps for each object. Figure 3.1 shows exemplarily the reference frame and one training frame of a session in the data set. The top row of Fig. 3.1 shows the reference frame, consisting of color image and depth map. Further, the bottom row depicts the collected training frame with color image, depth map and generated mask. The mask in Fig. 3.1(e) shows the labels assigned to each pixel. The meaning of the pixel color is described in the following:


(a) Color image (rf) (b) Depth map (rf)

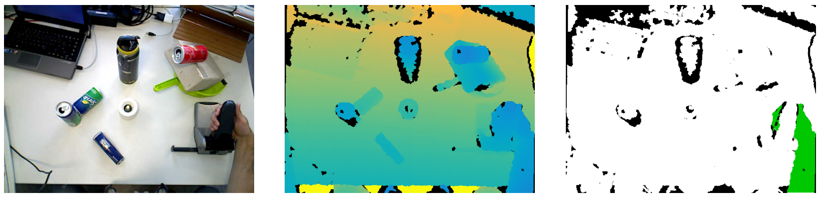

(c) Color image (tf) (d) Depth map (tf) (e) Mask (tf)

Figure 3.1: In the top row the reference frame (rf), which shows the scene without hand. The bottom row exemplarily shows a training frame (tf) from the same session. The depth maps are colored in order to give a better impression of the depth. The green pixels in the mask show skin, the white ones non-skin and the black ones result from invalid depth measurements.

Black pixels denote pixels with an invalid value in the depth map of either reference frame or training frame. Section 2.3 gives more insight on invalid depth measurements. The value of black is 0.

Grey (Green) pixels denote skin pixels. The value of the grey shade is 128.

White pixels denote background pixels. The value of white is 255.

In the following we illustrate the process of mask generation. Therefore, let us denote the depth map of the reference frame as

$$\mathbf{D}^{\mathrm{rf}} = \{D^{\mathrm{rf}}_{mn} \mid 0 \le m < M, 0 \le n < N\} \tag{3.1}$$

and similarly, the depth map of the ith training frame as

$$\mathbf{D}^{\mathrm{tf},i} = \{D^{\mathrm{tf},i}_{mn} \mid 0 \le m < M, 0 \le n < N\} . \tag{3.2}$$

The color images and depth maps are of size $M \times N$. Furthermore, let us assume that the reference frame is captured as well as the color image and depth map of the ith training frame. The labeling process is based on the difference in the depth maps. The difference $\Delta\mathbf{D}^i$ is defined as

$$\Delta\mathbf{D}^i = \mathbf{D}^{\mathrm{rf}} - \mathbf{D}^{\mathrm{tf},i} = \{\Delta D_{mn} \mid 0 \le m < M, 0 \le n < N\} , \tag{3.3}$$

because this definition assures a positive difference at pixel positions which are part of the hand in the training frame. Note that the depth measurements are noisy; thus, it is not possible to detect hand pixels with $\Delta \mathbf{D}^i \geq 0$. Additionally, the assumption is made that the depth maps are preprocessed, meaning that invalid depth measurements are represented by a value smaller or equal to the threshold τ_{inv}. The rule for assigning values to the mask $\mathbf{M}$ is defined as

$$M^i_{mn} = \begin{cases} 0 & \text{if } D^{\text{rf}}_{mn} \leq \tau_{\text{inv}} \text{ or } D^{\text{tf},i}_{mn} \leq \tau_{\text{inv}} \\ 128 & \text{if } \Delta D^i_{mn} \geq \tau \text{ and } D^{\text{rf}}_{mn} > \tau_{\text{inv}} \text{ and } D^{\text{tf},i}_{mn} > \tau_{\text{inv}} \\ 255 & \text{if } \Delta D^i_{mn} < \tau \text{ and } D^{\text{rf}}_{mn} > \tau_{\text{inv}} \text{ and } D^{\text{tf},i}_{mn} > \tau_{\text{inv}}\,, \end{cases} \tag{3.4}$$

where the superscript i denotes the number of the training frame, the subscripts m, n denote the element of the corresponding matrix, τ denotes the noise threshold and τ_{inv} denotes the threshold for invalid depth values. Once $\mathbf{M}^i$ is created all grey blobs are searched in $\mathbf{M}^i$, the number of blobs is denoted with J. For each blob $j = [0, 1, \ldots, J-1]$ an area measure is calculated. The area measure of the ith training frame and the jth blob is described as

$$C^{ij} = \sum_{\chi(m,n) \in \mathcal{L}^j} \frac{1}{D^i_{mn}}\,, \tag{3.5}$$

where $\mathcal{L}^j$ denotes the set of all pixel positions $\chi(m, n) = [m, n]$ of the jth blob. The blob with the largest area measure of all J blobs is chosen and denoted as blob j_0.

$$j_0 = \max_j \left(\{C^{ij} \mid j = [0, 1, \ldots, J-1]\} \right) \tag{3.6}$$

The data set also include grasps with occluded parts of the hand and it is possible that the finger tips and the hand are not connected in one blob due to occlusions, compare Fig. 3.1(c). Therefore, in the next step the shortest distance d^{j0} from blob j_0 to the jth blob with $A^j \geq A_{min}$ is calculated. Whereas, A^j denotes the area of the jth blob. In this case, shortest distance refers to the shortest distance between the contours of blob j_0 and blob j. The set of blobs which are kept is

$$\mathcal{O} = \{o \mid o = j_0 \text{ or } o = j\,\forall j \mid d^{j0} \leq d_{\text{min}}\}\,. \tag{3.7}$$

The set $\mathcal{O}$ contains all blobs which are part of the hand. In order to obtain the final mask the mask $\mathbf{M}$ needs to be updated because $\mathbf{M}$ obtained by (3.4) contains blobs not showing skin due to noise. The elements M^i_{mn} of the final mask are obtained by

$$M^i_{mn} = \begin{cases} 0 & \text{if } D^{\text{rf}}_{mn} \leq \tau_{\text{inv}} \text{ or } D^{\text{tf},i}_{mn} \leq \tau_{\text{inv}} k \\ 128 & \text{if } \chi(m,n) \in \mathcal{L}^j \forall j \in \mathcal{O} \\ 255 & \text{else}\,. \end{cases} \tag{3.8}$$

The definition of the area measure in (3.5) is made in order to make the priority of the pixel dependent on the inverse depth. Thus, blobs closer to the camera are more likely to be chosen as pixels containing hand. This prioritization is based on the assumption that the hand is usually an object closest to the camera in an ego-centric scenario.

After generation of the mask with the aforementioned algorithm, a manual inspection of the masks is done. The inspection is necessary because during the collection process some of the objects are moved creating non-skin blobs. Those falsely detected hand regions must be removed in order to obtain a data set of decent quality. The detection of faulty masks is done by comparing the color image with the corresponding mask. The correction is done by an image processing program, such as GIMP (GNU Image Manipulation Program). Figure 3.2 shows examples for non-skin blobs due to noise (in the first row) and an accidentally moved object (in the second row). Whereas, the images from left to right respectively show the RGB image, the ground truth mask before the cleaning step and the ground truth after the cleaning step.

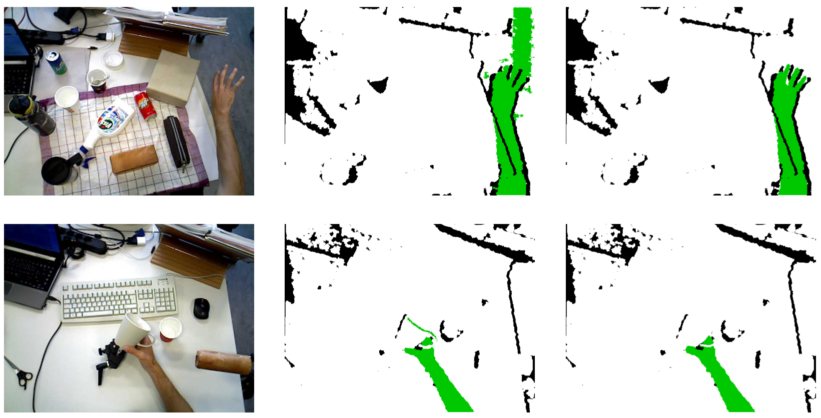


Figure 3.2: The figure illustrates the effect of cleaning the ground truth masks. In the first row an example for non-skin blobs due to noise is shown, in the second row an example for accidentally moved objects. The images from left to right respectively show the RGB image, the ground truth before the cleaning step and the ground truth after the cleaning step.

3.1.2 Objects in the Data Set

The data set is not collected with a specified set of objects but with randomly selected objects for each session. However, some of the objects frequently appear in the data sets. A selection of those objects is shown in Fig. 3.3. The objects in Fig. 3.3(e) and Fig. 3.3(f) are covered with skin-colored paper. This decision is made so that the classifier learns to distinguish between skin-colored objects and the human hand.

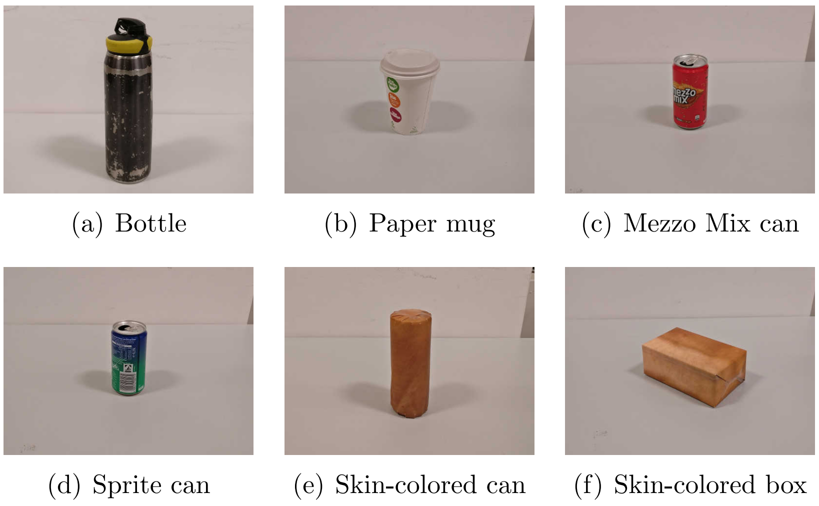

(a) Bottle (b) Paper mug (c) Mezzo Mix can
(d) Sprite can (e) Skin-colored can (f) Skin-colored box

Figure 3.3: A subset of the objects regularly appearing in the data set.

3.1.3 Implementation Details

The ROS (Robot Operating System) provides a very simple interface to the RGB-D camera *ASUS Xtion*. Therefore, the program used for data collection is written in `C++` as a ROS node. The ROS distribution *indigo* is used. The program stores frames with every keystroke and for every session recorded one reference frame is stored. The `C++` library OpenCV 2.4 is used to perform all image processing tasks needed, such as finding contours of blobs and filling contours with a specified value.

The algorithm in Sec. 3.1.1 defines and uses a number of constants to generate the mask. The values used for collecting the data set are shown in Table 3.1.

Name	Value	Unit	Short description
τ	9.9	mm	Threshold for difference of depth maps
τ_{inv}	10.0	mm	Threshold for indicating invalid depth measurement
d^{j0}	50.0	pixel	Shortest distance between contour of blob j and j_0
A_{min}	50.0	pixel^2	Minimum area of child blobs
M	450.0	pixel	Height of the color image, depth map and mask
N	598.0	pixel	Width of the color image, depth map and mask

Table 3.1: The table shows the chosen set of parameters for the data labeling approach. The parameters are described in Sec. 3.1.1.

3.2 Feature Selection

The selection of features is based on the approaches in [LK13, KTT$^+$16]. This thesis combines features from both color and depth images. In the following the reasons for selecting the proposed feature set are given, starting with features from color information and concluding with the features from the depth map.

3.2.1 Features from Color Information

Li and Kitani investigated a feature vector consisting of 498 features extracted from color information [LK13]. The features can be divided into two types, the low-level and high-level features. In (3.9) the feature vector of Li and Kitani is shown.

$$\begin{aligned}\mathbf{v} = [\ & \underbrace{\underbrace{v_1 \dots v_{75}}_{\text{RGB}}\ \underbrace{v_{76} \dots v_{150}}_{\text{HSV}}\ \underbrace{v_{151} \dots v_{225}}_{\text{CIELAB}}\ \underbrace{v_{226} \dots v_{273}}_{\text{Gabor}}}_{\text{low-level features}} \\ & \underbrace{\underbrace{v_{274} \dots v_{309}}_{\text{HOG}}\ \underbrace{v_{310} \dots v_{437}}_{\text{SIFT}}\ \underbrace{v_{438} \dots v_{453}}_{\text{BRIEF}}\ \underbrace{v_{454} \dots v_{485}}_{\text{ORB}}\ \underbrace{v_{486} \dots v_{498}}_{\text{superpixels}}}_{\text{high-level features}}\ ]\end{aligned} \tag{3.9}$$

The low-level features comprise the color information itself and the texture of the image; whereas, the high-level features are built from gradient descriptors. Li and Kitani show in [LK13] that the low-level features are the most discriminative features in their approach and that the high-level features are important for difficult decisions. Considering the computational expense of the high-level features, only the low-level features are chosen in our approach. Furthermore, the features from the depth map give additional differentiation and therefore may make the high-level features redundant.

Color Spaces

Li and Kitani [LK13] do not only consider the color values of the pixel to be classified in fact they take the surrounding pixels into account as well. This is reasonable because if the surrounding pixels are skin pixels the likelihood of the center pixel being a skin pixel is very high. Thus, the consideration of neighboring pixels is adopted to our feature vector. The same holds for the number and type of color representations. Li and Kitani found that a patch of 5×5 pixels give the best results but such a patch size adds up to 225 features in case of three different color representations. In Sec. 3.2.2 we choose to use 100 depth features in our proposed feature vector. Considering the number of depth features, a patch size of 3×3 pixels is chosen in order to avoid having too much color features in the vector. Table 3.2

shows the different color representations contained in the feature vector and the number of features.

Representation	Patch size	Number of values per pixel	Number of features
RGB	3×3	3	27
HSV	3×3	3	27
CIELAB	3×3	3	27
		In total	81

Table 3.2: Number of features from different color spaces used in our approach. The table shows the chosen patch size, the number of components per pixel and the total number of features for each color space.

The construction of the feature vector describing the color spaces is explained in the following. The feature vector contains the features from three different color spaces RGB, HSV and CIELAB. Let $\mathbf{I}_{m,n}^{\text{cspace}}$ return the color triplet at pixel position $\mathbf{x} = [n, m]^{\text{T}}$

$$\mathbf{c}^{\text{cspace}} = \mathbf{I}_{m,n}^{\text{cspace}}, \tag{3.10}$$

where "cspace" denotes the type of color space. The row and column index of the image are described respectively by m and n. The color triplet of color space "cspace" is described by the vector $\mathbf{c}^{\text{cspace}}$. With (3.10) the feature vector of a single color space can be formulated as

$$\mathbf{v}_{\mathbf{I},\mathbf{x}}^{\text{cspace}} = \left[\mathbf{I}_{m-1,n-1}^{\text{cspace}}, \ldots, \mathbf{I}_{m-1,n+1}^{\text{cspace}}, \ldots, \mathbf{I}_{m+1,n-1}^{\text{cspace}}, \ldots, \mathbf{I}_{m+1,n+1}^{\text{cspace}}\right], \tag{3.11}$$

Consequently, the feature vector of the color spaces is constructed as

$$\mathbf{v}_{\mathbf{I},\mathbf{x}}^{\text{color}} = \left[\mathbf{v}_{\mathbf{I},\mathbf{x}}^{\text{RGB}}, \mathbf{v}_{\mathbf{I},\mathbf{x}}^{\text{HSV}}, \mathbf{v}_{\mathbf{I},\mathbf{x}}^{\text{CIELAB}}\right]. \tag{3.12}$$

Image Texture

The size of the filter bank used to obtain the texture related features of the image is equal to the defined size in [LK13]. The filters used in the filter bank are Gabor filters. Gabor filters are discussed in Sec. 2.2 from a theoretical point of view. There are 24 Gabor filters in the filter bank. They differ in scale and filter orientation. Specifically, three different scales are used and each scale has eight different orientations. Furthermore, the real and imaginary part of the filters are used. The aforementioned choices result in a total of 48 features and a feature vector with 48 elements is generated. Bianconi and Fernández statistically evaluate the influence of Gabor filter parameters on the classification performance [BF07]. They run several experiments generating the features with different sets of parameters and statistically evaluate the influence of the parameters. The design choice is influenced by their findings.

As a short remainder the Gabor filter can be designed by the following parameters:

- Filter orientation θ
- Wavelength λ
- Standard deviation σ of the Gaussian envelope
- Spatial aspect ratio γ
- Filter size $2L+1$

The wavelength and the standard deviation change for each scale. In order to ensure precise notation, a subscript ij is added to the parameters, where $i \in [1,2,3]$ denotes the scale and $j \in [1;8]$ denotes the orientation. Note, that the smallest scale refers to $i=3$ and the smallest filter orientation refers to $j=1$.

Appendix B shows that the filter results for real input signals are the complex conjugate in the orientation range of $[0;\pi[$ and $[\pi;2\pi[$. Therefore, the filter orientations in the range of $[0;\pi[$ are sufficient. For an equal spacing of the orientations within that range an angular displacement of 22.5° is given for 8 different orientations. Consequently, the calculation rule for the orientation is given as

$$\theta_{ij} = \frac{\pi(j-1)}{8}\,, \qquad j = [1,2,\ldots,8]\,, \tag{3.13}$$

which holds for all i.

The smallest wavelength is chosen such that the highest frequency of the half-peak magnitude contour lines touches the Nyquist frequency (0.5). The smallest wavelength is used for the filter kernels at the smallest scale and is given by

$$\lambda_{\text{min}} = \left(\frac{1}{2} - \frac{\sqrt{\log_{\text{e}} 2}}{\sqrt{2}\pi\sigma_{\text{start}}}\right)^{-1}\,. \tag{3.14}$$

The definition in (3.14) can be easily derived by the following procedure. First, setting the magnitude of the Fourier transformation of the Gabor filter kernel (2.10) for $\theta = 0$ to the half-peak value. Second, finding the semiaxis along the x-axis. Finally, subtracting half of the semiaxis from the Nyquist frequency which results in the expression in (3.14). The smallest wavelength λ_{min} given the remaining values can be derived. The spacing is chosen to satisfy a half-octave frequency sampling and is formulated as

$$\lambda_{ij} = \sqrt{2}^{(3-i)}\lambda_{\text{min}}\,, \qquad i = [1,2,3]\,, \tag{3.15}$$

which holds for all i. If the scaling factor $\sqrt{2}$ is changed to 2 one uses octave frequency sampling.

The standard deviations σ_{ij} are chosen such that there is superposition between adjacent filters of the same filter orientation. The smallest value for the smoothing parameter is chosen to be equal to 2, which is denoted by σ_{start}. The remaining values σ_{ij} are given by

$$\sigma_{ij} = \sigma_{\text{start}}\sqrt{2}^{(3-i)}, \qquad i = [1, 2, 3], \tag{3.16}$$

which holds for all j.

The spatial aspect ratio γ, as σ_{ij}, is chosen such that adjacent filters superimpose each other. Additionally, the spatial aspect ratio γ is chosen which is constant for each scale and orientation. However, γ describes the overlap of adjacent kernels of constant scale in the frequency domain. The value of the spatial aspect ratio is set to 1.25 in order to obtain sufficient overlap of the half-peak contour lines in the frequency domain.

The last parameter used to design the filter kernel is L which is set to 15, in order to reduce the computational cost of image filtering, since each image has to be filtered 48 times. This choice of L results in a filter kernel size of 31×31 pixels.

The presented filter design results in the coverage of the frequency domain sketched in Fig. 3.4. The blue numbers, corresponding to j, indicate the orientation angle. Whereas, the line type of the contour describes the scale. The dotted line corresponds to $i = 1$, the dashed line to $i = 2$ and the solid line to $i = 3$.

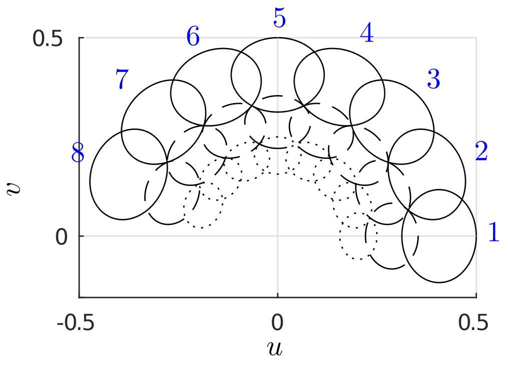


Figure 3.4: The plot shows the half-peak contour lines of Gabor filter bank in the frequency domain. The blue numbers indicate the different orientations j and the different line types of the contours represent the different scales i. Whereas, the dotted line corresponds to the largest scale, the solid line corresponds to the smallest scale. The axes u and v are the normalized frequencies.

The design above is chosen according to the findings in [BF07]. Bianconi and Fernández concluded from their comparison that half-octave frequency sampling performs better than octave frequency sampling and the smallest wavelength should be chosen according to (3.14). Moreover, they found that σ should be at the lowest level to achieve better results. Furthermore, their results show that superposition of adjacent filter kernels in the frequency domain results in better classification performances.

The feature vector comprising the features of the color texture is

$$\begin{aligned}\mathbf{v}_{\mathbf{I},\mathbf{x}}^{\text{texture}} = \big[\,&\mathrm{Re}\{\mathbf{G}_{00}(\mathbf{x})\}, \mathrm{Im}\{\mathbf{G}_{00}(\mathbf{x})\}, \dots, \\ &\mathrm{Re}\{\mathbf{G}_{ij}(\mathbf{x})\}, \mathrm{Im}\{\mathbf{G}_{ij}(\mathbf{x})\}, \dots, \\ &\mathrm{Re}\{\mathbf{G}_{38}(\mathbf{x})\}, \mathrm{Im}\{\mathbf{G}_{38}(\mathbf{x})\}\big]\,,\end{aligned} \tag{3.17}$$

where $\mathbf{G}_{ij}(\mathbf{x})$ describes the filter output for the Gabor filter with the configuration ij. The parameter i describes the scale and j describes the filter orientation. The color image $\mathbf{I}$ is converted to a greyscale image $\mathbf{B}$. Thus, the filter output is given as

$$\mathbf{G}_{ij} = \mathbf{B} ** \mathbf{h}_{ij}\,, \tag{3.18}$$

where $\mathbf{h}_{ij}$ is the complex Gabor filter kernel for configuration ij and $**$ describes the two dimensional convolution. Figure 3.5 shows the output of the filter bank for orientation $j = 1$ and all scales. The first column shows the greyscale image of the example and the columns two, three and four respectively display the outputs for scales $i = 1, 2$ and 3. The filter outputs show which parts in the image have texture and which are homogeneously colored.

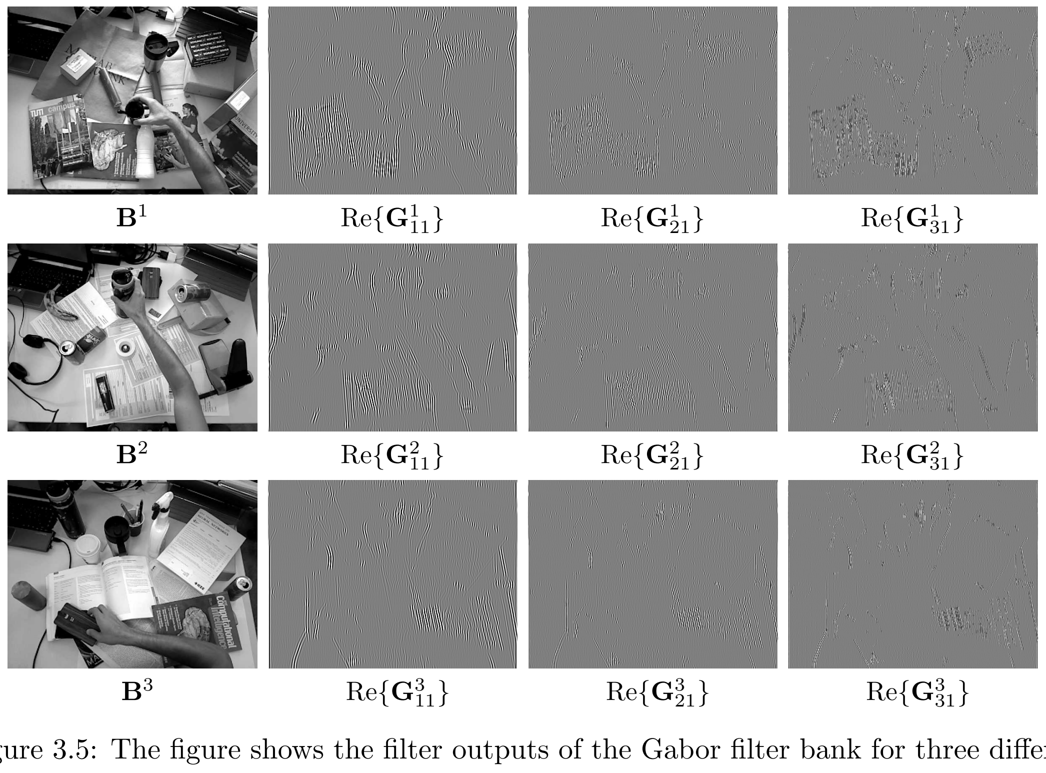

$\mathbf{B}^1$
$\mathrm{Re}\{\mathbf{G}_{11}^1\}$
$\mathrm{Re}\{\mathbf{G}_{21}^1\}$
$\mathrm{Re}\{\mathbf{G}_{31}^1\}$
$\mathbf{B}^2$
$\mathrm{Re}\{\mathbf{G}_{11}^2\}$
$\mathrm{Re}\{\mathbf{G}_{21}^2\}$
$\mathrm{Re}\{\mathbf{G}_{31}^2\}$
$\mathbf{B}^3$
$\mathrm{Re}\{\mathbf{G}_{11}^3\}$
$\mathrm{Re}\{\mathbf{G}_{21}^3\}$
$\mathrm{Re}\{\mathbf{G}_{31}^3\}$

Figure 3.5: The figure shows the filter outputs of the Gabor filter bank for three different examples. The figures shows the result for all three scales i and the orientation $j = 1$ which corresponds to 0°. The first column shows the greyscale image of the example. The columns two, three and four respectively show the output for $i = 1, 2$ and 3.

3.2.2 Features from Depth Map

The depth features are used in [SSK$^+$13, KTT$^+$16] and the method Kang et al. employ in [KTT$^+$16] serves as example in our approach. Shotton et al. and Kang et al. use depth features formulated as

$$f_k(\mathbf{D}, \mathbf{x}) = d_{\mathbf{D}} \left(\mathbf{x} + \frac{\mathbf{u}_k}{d_{\mathbf{D}}(\mathbf{x})} \right) - d_{\mathbf{D}} \left(\mathbf{x} + \frac{\mathbf{v}_k}{d_{\mathbf{D}}(\mathbf{x})} \right) , \tag{3.19}$$

where $d_{\mathbf{D}}(\mathbf{x})$ returns the depth value of $\mathbf{D}$ at pixel position $\mathbf{x} \in \mathbb{R}^2$. The vectors $\mathbf{u}_k \in \mathbb{R}^2$ and $\mathbf{v}_k \in \mathbb{R}^2$ describe a displacement of pixel position $\mathbf{x}$. Both vectors describe the displacement in pixels at a depth of one meter; thus, the unit of the vectors is [pixels · m]. The total number of depth features is chosen to be 100. Therefore, $k \in [1, 2, \ldots, 100]$ is used to indicate the depth feature number. The function $d_{\mathbf{D}}(\mathbf{x})$ returns the maximum depth in the depth map $\mathbf{D}$ if the depth at $\mathbf{x}$ is smaller than 10 mm or if $\mathbf{x}$ is outside the image boundaries. The vector

$$\mathbf{v}_{\mathbf{D},\mathbf{x}}^{\text{depth}} = \left[f_1(\mathbf{D}, \mathbf{x}), f_2(\mathbf{D}, \mathbf{x}), \ldots, f_{100}(\mathbf{D}, \mathbf{x}) \right] \in \mathbb{R}^{100} , \tag{3.20}$$

collects the depth features for depth map $\mathbf{D}$ and pixel position $\mathbf{x}$ from (3.19).

Figure 3.6 visualizes how the depth features are calculated. The displacement vectors in pixel are calculated and are shown as the black arrows in the figure. The feature is then calculated by subtracting the depth at the red circle from the depth at the red cross. Thus, the depth feature represents the difference in depth between the beginning and the end of the red line.

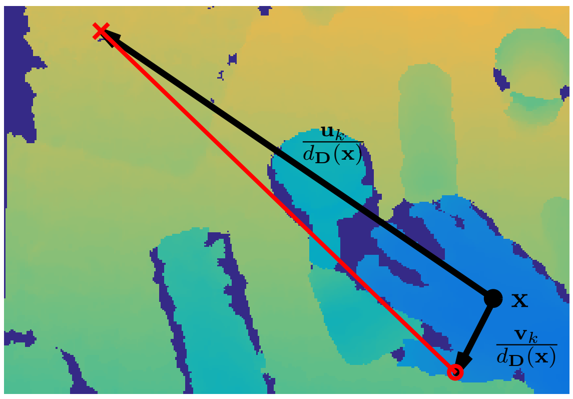

$\frac{\mathbf{u}_k}{d_{\mathbf{D}}(\mathbf{x})}$
$\mathbf{x}$
$\frac{\mathbf{v}_k}{d_{\mathbf{D}}(\mathbf{x})}$

Figure 3.6: The scheme in the figure visualizes the applied depth features. The depth at the red circle is subtracted from the depth at the red cross and the difference is used as depth feature. The black dot denotes the pixel $\mathbf{x}$ which is characterized by the depth feature while the black arrows represent the randomly drawn displacements normalized by the depth at $\mathbf{x}$.

The components of the two dimensional displacement pair $(\mathbf{u}_k, \mathbf{v}_k)$ are randomly drawn, where either $\mathbf{u}_k$ or $\mathbf{v}_k$ are set to $[0, 0]$ for half of the displacement pairs. The range of the components of the displacement vectors in physical measure of distance is $[-0.4\,\mathrm{m}; 0.4\,\mathrm{m}]$. In order to get the range of the displacement vectors the physical range has to be multiplied by the focal length $f = 525\,\mathrm{pixel}$. Therefore, the displacement vectors are randomly drawn from the range $[-210\,\mathrm{pixel}\cdot\mathrm{m}; 210\,\mathrm{pixel}\cdot\mathrm{m}]$. In Sec. 3.3.2 a comparison of three different distributions is given. The three distributions are:

1. Uniform distribution in the range of $[-210\,\mathrm{pixel}\cdot\mathrm{m}; 210\,\mathrm{pixel}\cdot\mathrm{m}]$. The patch size for randomly drawing depth features is: $0.8\,\mathrm{m} \times 0.8\,\mathrm{m}$.
2. Normal distribution with a standard deviation $\sigma_{70} = 70\,\mathrm{pixel}\cdot\mathrm{m} \mathrel{\hat{\approx}} 0.13\,\mathrm{m}$. The patch size where 68 % of the features are drawn from is: $0.27\,\mathrm{m} \times 0.27\,\mathrm{m}$.
3. Normal distribution with a standard deviation $\sigma_{35} = 35\,\mathrm{pixel}\cdot\mathrm{m} \mathrel{\hat{\approx}} 0.07\,\mathrm{m}$. The patch size where 68 % of the features are drawn from is: $0.13\,\mathrm{m} \times 0.13\,\mathrm{m}$.

The effect of the distribution of the displacement vectors can be seen in Fig. 3.7. The black dot indicates the pixel position for which the depth features are generated. As in Fig. 3.6 the start and the end of the red lines indicate where the difference in the depth map is calculated. It can be seen that the features in Fig. 3.7(a) are uniformly distributed across the entire image. This implies that a lot of the features do not describe the shape of the hand. Figure 3.7(b) shows that for Normal distributed displacement vectors the spatial area covered by the features is closer to the point which is classified. Consequently, in this case we believe that the shape of the hand is described better. The spatial locations of the depth features for displacement vectors drawn from a Normal distribution with σ_{35} can be seen in Fig. 3.7(c). The third configuration should represent the shape of the hand best; therefore, a comparison of the different configurations is done in Sec. 3.3.2.

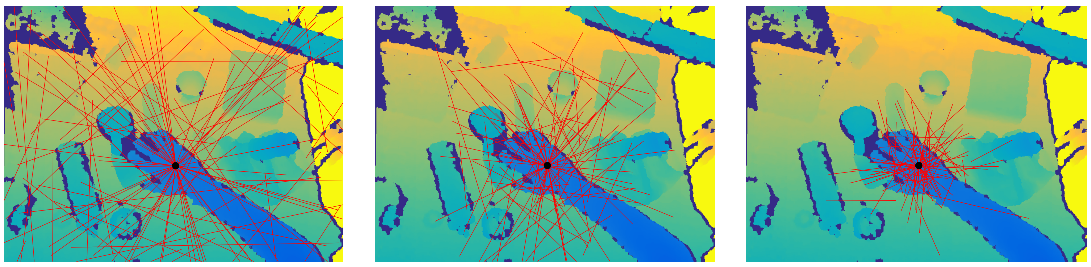

(a) Uniform distribution (b) Normal distribution (σ_{70}) (c) Normal distribution (σ_{35})

Figure 3.7: Comparison of the spatial distribution for different types of depth features. The difference in the depth features results from the choice of the random distribution of the displacement vectors. Three distributions are compared and each red line represents one depth feature. The depth features shown are computed to characterize the pixel indicated by the black dot.

The displacement vectors are generated in advance of the experiments and are fixed for all experiments. Thereby, comparability between the experiments is ensured.

3.2.3 Proposed Feature Vector

The proposed feature vector can be seen in (3.21) and is constructed by concatenating (3.12), (3.17) and (3.20). This feature vector has 229 dimensions and is generated for each training or data point $\mathbf{x}$ to be classified.

$$\begin{aligned}\mathbf{v}^{\text{proposed}}_{\mathbf{I},\mathbf{D},\mathbf{x}} &= \left[\mathbf{v}^{\text{color}}_{\mathbf{I},\mathbf{x}}, \mathbf{v}^{\text{texture}}_{\mathbf{I},\mathbf{x}}, \mathbf{v}^{\text{depth}}_{\mathbf{D},\mathbf{x}}\right] \\ &= \Big[\underbrace{\underbrace{v_1 \dots v_{27}}_{\text{RGB}} \; \underbrace{v_{28} \dots v_{54}}_{\text{HSV}} \; \underbrace{v_{55} \dots v_{81}}_{\text{CIELAB}}}_{\text{81 color features}} \\ &\qquad \underbrace{v_{82} \dots v_{129}}_{\text{48 features from image texture (Gabor filter)}} \; \underbrace{v_{130} \dots v_{229}}_{\text{100 depth features}} \Big]\end{aligned} \tag{3.21}$$

3.3 Random Decision Forest Design

The design of the classifier is crucial for its performance. Section 3.3.1 gives some background to specifics with respect to the implementation of the RF, mostly describing the chosen training scheme. In Sec. 3.3.2 the choice of the hyper parameters, as well as the reasoning for it, are described. Moreover, the reasons for the choice of the depth feature type, the RF size and the choice of the tree depth are given.

3.3.1 Implementation

The software framework for the experiments is written in `C++` and the image processing modules as well as the machine learning algorithm are part of the OpenCV 2.4 library. The software has a terminal interface and allows to start a training process as well as a classification process for existing data. However, it is important to note that the implementation is not real-time compliant because the RF implementation does not take advantage of the parallel processing on the GPU. Sharp et al. show that RFs can be implemented on GPUs to make the classification process faster [Sha08].

The RF is configured as classifier for two classes. Tree growth is terminated if less than 10 training points reach the node, if the maximal tree depth is reached or if all training points at the node have the same class. The training of the RF is finished as soon as the defined forest size is reached.

Training Scheme

The classifier training is done with the training data set (Sec. 3.1). However, not every pixel inside the images is used for training. In the training images the non-skin pixels outnumber the skin pixels by a significant factor. Therefore, the training uses a fraction of all available pixels but per image the number of skin and non-skin pixels is the same. In each image 1000 data points per class are extracted which sums up to 2000 data points per image. The data points used for training are uniformly spread throughout the images. In total, around 6.8 million data points are available for training. This training set has been fixed at the beginning because for each experiment at least one configuration has been trained. Fixing the training set simplifies the comparison between the different configurations. To sum up, less than 1 % of all available training data points are used for training but the performance metric is calculated on all pixels in the training data set.

The depth map of the RGB-D image may contain invalid measurements which are indicated by the value 0, as described Sec. 2.3. Such data points or pixels within the training data set are not used for training. However, data points whether in the training or verification data set with invalid depth measurement are interpreted as non-skin pixels for calculation of performance metrics.

Every training point is described by a vector of 230 values, 229 features and a class label (920 B). The available training set contains around 6.8 million training points. Thus, the training set occupies roughly 6.3 GB of fast memory. The entire training set together with the overhead of the RF training does not fit into fast memory. Therefore, the trees are trained separately and each tree is trained with 60 % of the training set. The training points used for training are drawn uniformly.

3.3.2 Hyper Parameters

In this section, the reasons for the choice of the RF hyper parameters are laid out. The hyper parameter evaluation is done based on the depth features alone. First, the size of the RF is evaluated. The number of trees is not a critical parameter and too many trees to not cause overfitting but only increase the computational cost. Second, the type of depth features is evaluated. Third, different tree depths are compared in order to find a sufficient one. The computation of the performance metrics in this section is based on the entire training data set containing 3420 images.

RF Size

The number of trees in the RF should be chosen reasonably in order to reduce the computational cost of the classification process. Thereby, the efficiency of the RF for training and classification is improved. The investigation is done with a RF which classifies solely based on the depth features. The depth features are distributed according to a Normal distribution with the standard deviation of σ_{35} and the tree depth is set to 20. This section evaluates the performance for different sized RFs. The biggest RF contains 210 decision trees which are divided into groups of 10 trees. Each group has one vote and votes for either skin or non-skin, i.e. there are 21 groups and votes. A skin pixel is classified if more than half of the groups vote for skin. For example in case of an RF consisting of three groups, a skin pixel is classified if more than 1.5 respective votes are present. Collecting the classification results in a matrix leads to the segmentation mask which is compared to the ground truth mask to compute the performance metric.

The performance metric is calculated for different numbers of votes. There are 21 votes available; thus, 21 different RF sizes can be evaluated. For each RF size several RFs can be constructed by combining different groups. There are 21 possibilities to construct an RF containing one group or 10 trees as well as for an RF containing 20 groups or 200 trees. The number of permutations for RF sizes of 20 to 190 trees is a lot larger than 21. Considering the computational expenses of the evaluation we restrict ourselves to use only 21 different permutations. Thereby, for each RF size 21 values of the performance metric are obtained. The mean and standard deviation of these values are computed.

The result of the evaluation is shown in Fig. 3.8 using the F_1-score which is defined as

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \,. \tag{3.22}$$

The solid blue line shows the mean value of the F_1-score for different RF sizes in steps of 10 trees. The error bars indicate the standard deviation for the different sizes. The standard deviation of the results is low for each RF size. This indicates that each group of trees has approximately the same performance while still being different. The graph shows two more interesting characteristics. First, the performance of the RF for increasing size shows a saturating behavior, see the dashed grey lines. From the graph it is obvious that a RF size of 50 to 60 trees is sufficient. Second, the blue line shows strong oscillations; whereas, for even numbers of votes the performance is better than for odd numbers of votes. Surprisingly, a RF of just 20 trees or 2 groups seems to outperform all 20 other RF sizes. The reason for this is based on the fact that small RFs with an even number of groups (even number of available votes) show a better performance because the detection threshold is larger than 50 % in reality.

The oscillations of the graph in Fig. 3.8 occur due to a skin detection threshold larger than 50 %. In the following we give a short explanation of this effect. Considering two available votes, there are three possible outcomes: no vote for skin, 1 vote for skin and 2 votes for skin. The threshold to detect a skin pixel of a RF which has two votes is 1, i.e. there is just one case where the RF classifies a pixel as skin: all

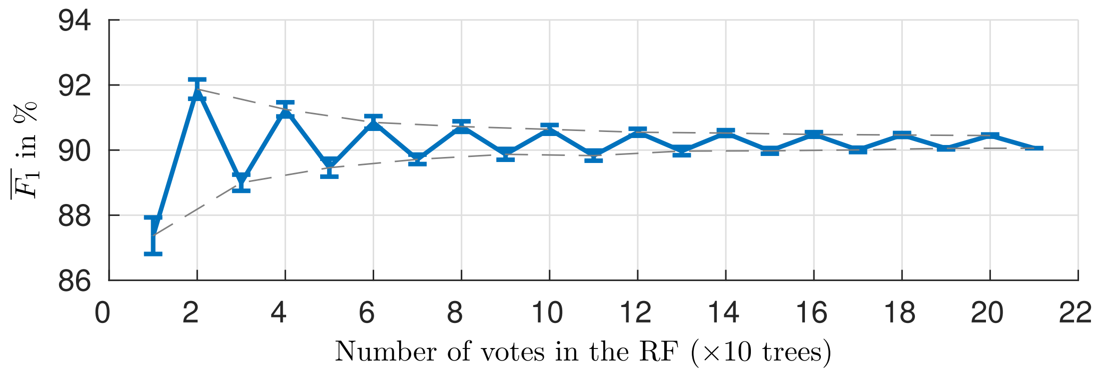


Figure 3.8: The figure shows the average F_1-score for differently sized RFs using depth features only. The blue line indicates the mean F_1-score for different RF sizes in steps of 10 trees. The error bar indicates the standard deviation for the different RF sizes. The dashed grey line indicates the envelope.

votes say skin. Therefore, the real threshold is 66.7 % instead of 50 %. Thus, the detection threshold t_{skin} is given as

$$t_{\text{skin}} = \frac{\lfloor \frac{n_{\text{votes}}}{2} \rfloor + 1}{n_{\text{votes}} + 1} \cdot 100\,\%\,, \tag{3.23}$$

where $\lfloor \cdot \rfloor$ denotes the floor function and n_{votes} describes the number of votes available in the RF. Figure 3.9 shows the detection threshold for classifying skin. Comparing this figure with Fig. 3.8 results in a similar picture and explains the behavior in F_1-scores between even and odd number of votes.

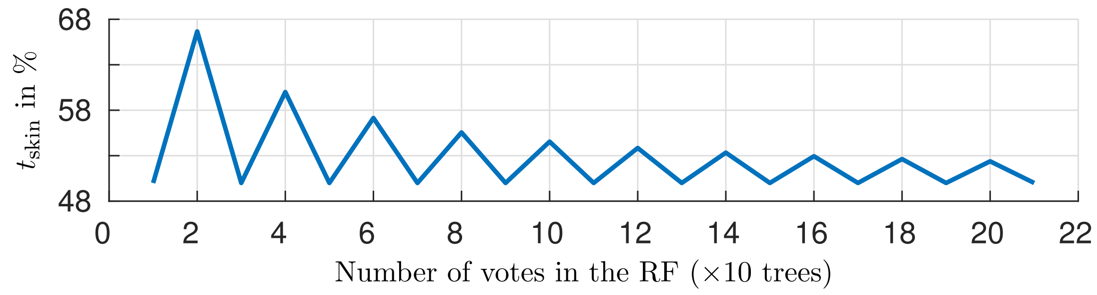


Figure 3.9: The graph shows the change of the detection threshold for different RF sizes. Each group of 10 trees has a vote and the virtual skin detection threshold is 50 % of the available votes. The oscillations occur due to oscillating behavior of the actual t_{skin}.

This investigation indicates that the trade-off between performance and computational cost is reasonable for RFs with approximately 60 trees. Furthermore, the result of the RF size evaluation shows that a detection threshold for skin of 50 % is not optimal. Therefore, the optimal threshold is evaluated in one of the subsequent sections. Further, to adapt the threshold more precisely one vote per tree is generated. We assume that the optimal RF size is not strongly dependent on the features used and we do not have to evaluate the size for every configuration. Besides, our RF is considerably larger than the one used in [LK13] or [KTT+16] even though Li et al. employ a larger feature set.

Type of Depth Features

In this evaluation three different RF are trained, each containing 60 trees with a tree depth of 20. The training is done with depth features only. The difference between the RFs is that the depth features are extracted using different spatial distributions (Sec. 3.2.2). In this evaluation we employ a vote for each tree not only one vote per 10 trees because we want to be able to adjust the detection threshold more precisely for upcoming evaluations and experiments. The detection threshold t_{skin} for this evaluation is again set to 50 %.

Tab. 3.3 depicts the results of the different depth feature configurations. All data points available in the training data set are used to compute the F_1-scores. Note, that less than 1 % of the entire set is used for training. The table shows that if the depth features are spatially distributed according to the Normal distribution with a standard deviation of σ_{35}, they perform best. Furthermore, it is obvious that with increasing spatial restriction of the depth features the performance can be improved. This indicates that more spatially restricted features better reflect the hand shape. Considering the result of the comparison and the prospective to restrict the segmentation process to a RoI, the Normal distribution with the smallest standard deviation is chosen.

Type	Uniform	Normal σ_{70}	Normal σ_{35}
F_1-score	85.4 %	86.9 %	88.4 %

Table 3.3: F_1-score for different types of depth features. The applied skin detection threshold t_{skin} is 50 %. For the computation of the metrics each tree of the RFs with 60 trees has a vote. The skin detection threshold is set to 50 %.

Depth of Trees

In this section four different tree depths 10, 15, 20 and 25 are compared. In this evaluation a broader spectrum of skin detection thresholds is considered in order to compare the performance of the different RFs not only for $t_{\text{skin}} = 50\,\%$. Each RF contains 60 decision trees and each decision has one vote. The spatial distribution of the depth features is according to the Normal distribution with σ_{35}. The main impact of the parameter tree depth is its sensitivity to overfitting according to [CS13].

Figure 3.10 shows the mean F_1-score over all pixels in the training data set for different tree depths. The deeper the tree is, the better the F_1-score gets. The improvement in the maximal F_1-score with approximately 10 % is highest if the tree depth is increased from 10 to 15. Increasing the tree depth by 5 up to 20 increases the F_1-score by approximately 5 %. The increase from 20 to 25 is negligible. Furthermore, 1 % of the data in the set has been used for training meaning that the set is not completely independent of the set used for training. Thus, we assume that overfitting of the training data does not cause the increase in F_1-score for a tree depth of 25 compared to a tree depth of 10. The reason for this assumption is that the number of unknown pixels outnumber the pixels used for training by far. Therefore, overfitting to the training pixels would cause an improvement in F_1-score for training pixels but would decrease the F_1-score for all other pixels. As a result, the average F_1-score would drop.

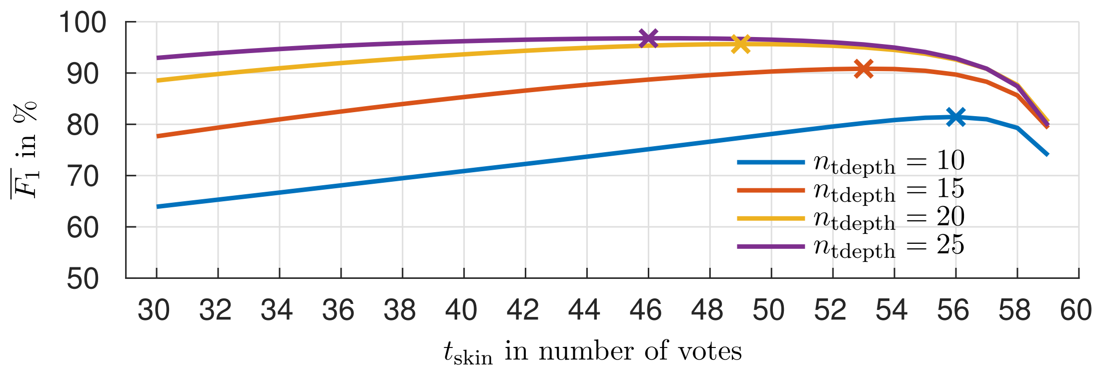


Figure 3.10: Average F_1-score $\overline{F_1}$ for different skin detection thresholds and four different tree depths. The detection threshold t_{skin} is expressed in number of votes. The cross in the lines mark the maximal F_1-score of the corresponding tree depth.

Table 3.4 shows the training and classification time for the different tree depths. The training times in the table result from the training scheme described in Sec. 3.3.1. The classification times displayed in the table yield from classifying an RGB-D image with a height of 450 pixel and a width of 598 pixel. Intuitively not only should it take more time to train deeper trees also classification should take longer. This understanding is confirmed in Tab. 3.4.

Tree depth	10	15	20	25
Training time of RF	2 h 2 min	2 h 45 min	3 h	3 h 25 min
Classification time per image	$\sim$ 3.6 s	$\sim$ 5.6 s	$\sim$ 6.8 s	$\sim$ 7.1 s

Table 3.4: Training and classification times of RF with different tree depths. The training time denotes the time needed to train the entire RF and the classification time is the time needed to classify an entire 450 × 598 RGB-D image.

Li et al. [LK13] choose a maximum tree depth of $n_{\text{tdepth}} = 10$; whereas, Kang et al. [KTT$^+$16] pick $n_{\text{tdepth}} = 20$. The evaluation of the tree depth in this section shows that $n_{\text{tdepth}} = 20$ improves the results compared to $n_{\text{tdepth}} = 10$. Moreover, the increase in F_1-score for a tree depth of 25 compared to a depth of 20 is negligible, while training and classification would consume more time. Therefore, we decide to employ $n_{\text{tdepth}} = 20$.

Skin Detection Threshold

The evaluation of the RF size shows that a detection threshold of 50 % is not the optimal choice. Therefore, this section investigates the performance with different skin detection thresholds t_{skin}. A skin detection threshold of 50 % shows a performance worse than the one of a higher threshold. In this evaluation the threshold in terms of votes is increased from 30 to 59 in order to detect the optimal threshold value. As in the previous evaluations the RF contains 60 trees with a tree depth of 20, while each tree votes either for non-skin or skin. The spatial distribution of the depth features originates from the Normal distribution with σ_{35}. Three different RFs are compared one using depth features only, one using color features only and one using all features.

Figure 3.11 shows the result said evaluation. The solid blue line in the figure shows the performance of the classifier employing the depth features only, while the red line shows the results for color only and yellow for using all features. As can be seen for all configurations the best performance is achieved for considerably higher thresholds than 50 % or 30 votes. The adjustment of the skin detection threshold improves the performance by around 6 % to 12 %. The best results for the configurations are collected in Tab. 3.5, while in Fig. 3.11 the optimal thresholds are indicated with a cross. The table shows the optimal threshold in number of votes, the F_1-score for the optimal skin detection threshold and the F_1-score for a threshold of 50 %. In Chap. 4 thresholds are applied on the classification result in order to compare the performance of the different classifiers. Those thresholds are obtained by the same method as in this section. However, for each experiment and each classifier the threshold has to be evaluated again based on the training data set. The adjustment of the threshold improves the average F_1-score by approximately 7 % to 12 %.

Type	only depth features	only color features	all features
F_1-score (best)	96.7 %	68.5 %	94.7 %
F_1-score (50 %)	90.5 %	57.7 %	88.8 %
t_{skin} in votes	49 votes	56 votes	53 votes

Table 3.5: Results of threshold evaluation with different feature combinations. The table shows the best average F_1-score and for a t_{skin} of 50 %.

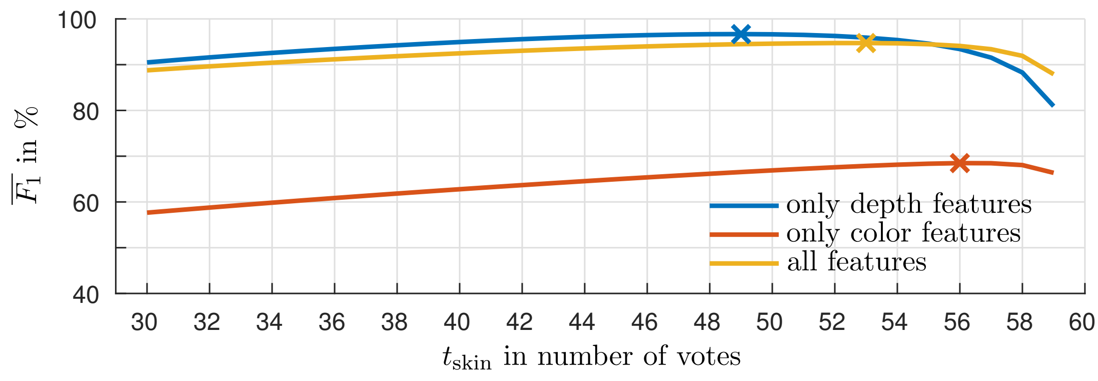


Figure 3.11: Average F_1-score $\overline{F_1}$ for different skin detection thresholds and feature combinations. The detection threshold t_{skin} is expressed in number of votes. The cross in the lines mark the maximal F_1-score for each feature combination.

Final Design

The final design of the different RFs for the experiments in Chap. 4 is summarized in this section. The design is chosen based on the outcomes of the hyper parameter evaluations in Sec. 3.3.2. We do not verify the impact of an alternative choice of hyper parameters, instead we assume that the choice of the RF size, the type of depth features and the depth of trees does not influence the preceding evaluations. The RF design we choose is summarized in Tab. 3.6.

Parameter	Choice
Bagging	60 % of all training points randomly drawn to train 10 trees
RNO	A set of $\sqrt{n_{\text{features}}}$ is drawn to optimize each node
RF size	60 trees
Depth features	Normal distribution with $\sigma_{35} \approx 0.07\,\text{m}$
Tree depth	$n_{\text{tdepth}} = 20$
Threshold	Only depth features: $t_{\text{skin}} = 49$ votes Only color features: $t_{\text{skin}} = 57$ votes All features: $t_{\text{skin}} = 53$ votes

Table 3.6: Parameters of the final RF design

3.4 Region of Interest Extraction

This section explains the scheme for the RoI extraction in the collected training and verification data sets. It is a simple and efficient way to extract the RoI from ground truth data without available hand position. This step is necessary because during the data collection phase no hand tracker was used to obtain an initial estimate of the hand position. The hand position could be estimated from the hand motions in previous frames or with the *Kinect* skeletal tracker [SKR+15].

The data has been collected from the ego-centric perspective while using the right hand only. However, the following scheme can also be applied if the left hand is used instead of the right hand. In case of the right hand and a labeled data set, it is reasonable to assume that the arm appears in the bottom right area of the RGB-D images, see for example Fig. 3.12. This assumption is the start of our RoI extraction scheme. Basically, a RoI search is applied twice on the same mask.

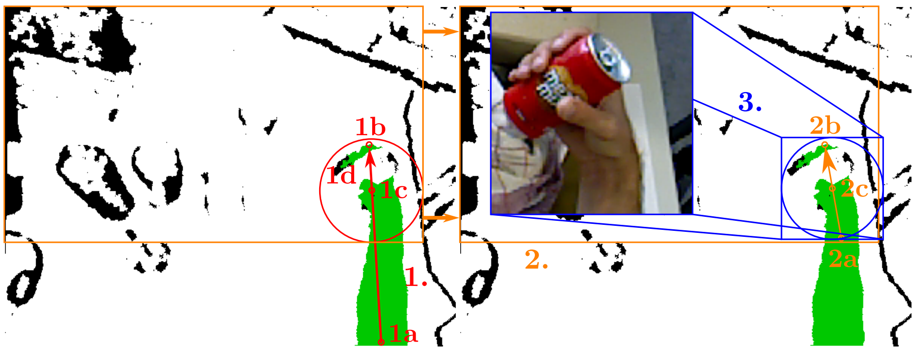


Figure 3.12: Visualization of the proposed RoI extraction scheme. The figure shows the same mask twice next to each other. First, estimate initial RoI from mask (red) and define new search area. Second, find second RoI estimate in defined search area (orange). Third, extract the final RoI (blue).

Figure 3.12 shows the process of the RoI extraction process. In the figure one can see the same ground truth mask next to each other to make the method clearer. In the following, The RoI extraction process is described in detail. The numbers in the figure correspond to the numbers in the following enumeration the letters indicate single steps. The pixels of the mask are addressed in the usual way which corresponds to the cell addressing of matrices. Therefore, the pixel in row m and column n is denoted as $\mathbf{x} = [n, m]^{\mathrm{T}}$. The size of the extracted RoI is $0.2\,\mathrm{m} \times 0.2\,\mathrm{m}$ and the radius of the RoI's incircle is $r_{\mathrm{RoI}} = 0.1\,\mathrm{m}$.

1. The red colored items in Fig. 3.12 show the first step. The RoI extraction process is applied on the entire mask, at the beginning. Specifically the following intermediate steps are executed:

(a) The *start point* $\mathbf{x}^{\text{start}}$ is found by searching for the hand pixels with the highest row index $m_{\max}$ and average the corresponding column indices n to obtain n_{avg}. Consequently, the *start point* $\mathbf{x}^{\text{start}} = [m_{\max}, n_{\text{avg}}]^{\mathrm{T}}$.

(b) The *end point* $\mathbf{x}^{\text{end}}$ is found by searching for the hand pixel with the largest Euclidean distance to $\mathbf{x}^{\text{start}}$.

$$\mathbf{x}^{\text{start}} = \underset{\mathbf{x}\in\mathcal{X}_{\text{skin}}}{\arg\max}\{\|\mathbf{x}^{\text{start}} - \mathbf{x}\|_2\}\,,$$

where $\mathcal{X}_{\text{skin}}$ describes the set of all skin pixels in the mask.

(c) The *centroid* of the RoI is assumed to be located on the connecting line between $\mathbf{x}^{\text{start}}$ and $\mathbf{x}^{\text{end}}$. The diameter of the RoI in pixels depends on the distance between the camera and the skin. The depth of the skin is approximated by the average depth of all skin pixels.

$$\overline{d}_{\text{skin}} = \frac{1}{|\mathcal{X}_{\text{skin}}|}\sum_{\mathbf{x}\in\mathcal{X}_{\text{skin}}} d_{\mathbf{D}}(\mathbf{x})\,,$$

where $d_{\mathbf{D}}(\mathbf{x})$ returns the depth at pixel position $\mathbf{x}$ and $|\cdot|$ describes the cardinality of the set. Thus, the *centroid* $\mathbf{x}^{\text{centroid}}$ is given as

$$\mathbf{x}^{\text{centroid}} = \mathbf{x}^{\text{end}} - 0.9\cdot\frac{r_{\text{RoI}}f}{\overline{d}_{\text{skin}}}\left(\frac{\mathbf{x}^{\text{end}} - \mathbf{x}^{\text{start}}}{\|\mathbf{x}^{\text{end}} - \mathbf{x}^{\text{start}}\|_2}\right),$$

where f is the focal length and the factor 0.9 is chosen to ensure a gap of 1 cm or 1 % of the RoI radius between the border of the RoI and the closest skin pixel.

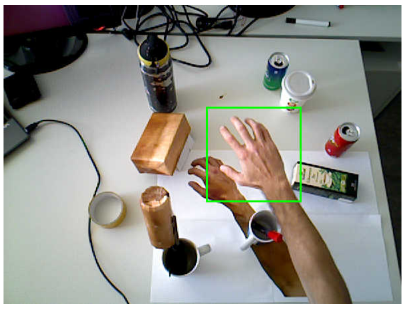
(d) The circular RoI is determined by centering a circle at $\mathbf{x}^{\text{centroid}}$. The radius of the circle is $r = \frac{r_{\text{RoI}}f}{\overline{d}_{\text{skin}}}$ pixel.

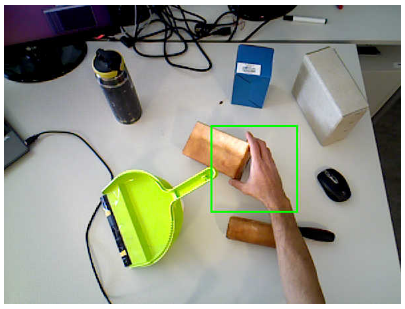
2. This step is indicated by the orange colors in Fig. 3.12 and improves the RoI extraction result of step 1. First, the circular RoI from intermediate step 1d is used to extract a new search area. This area is given by spanning a rectangle from $\mathbf{x}^{\text{origin}} = [0,0]^{\mathrm{T}}$ to $\mathbf{x}^{\text{rb}} = \mathbf{x}^{\text{centroid}} + r\cdot[1,1]^{\mathrm{T}}$. Inside this area, called mask $\mathbf{M}^{\text{new}}$, the intermediate steps 1a to 1c are executed; however, the set $\mathcal{X}_{\text{skin}}$ contains all skin pixels $\mathbf{x}$ inside $\mathbf{M}_{\text{new}}$. Furthermore, let us denote the extracted *centroid* and radius of the circular RoI from $\mathbf{M}_{\text{new}}$ with respectively $\mathbf{x}_2^{\text{centroid}}$ and r_2.

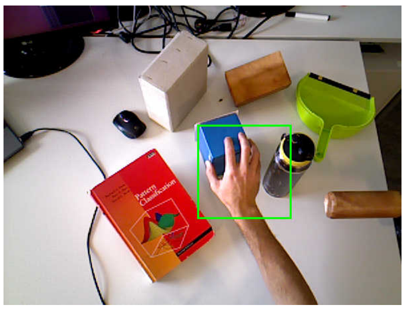
3. In the previous step $\mathbf{x}_2^{\text{centroid}}$ and r_2 are extracted, in this step both values are used to extract the RoI from the original mask and not from $\mathbf{M}_{\text{new}}$. This effect can be seen in Fig. 3.12. The newly computed average depth $\overline{d}_{\text{skin}}$ in step 2 can be different from the one computed in intermediate step 1c because the hand itself may be closer to the camera than the arm. Similarly, $\mathbf{x}^{\text{centroid}}$ moves

down or to the right. In order to get the complete RoI, it is extracted from the original mask. However, for certain RGB-D images in the data set this scheme fails. Therefore, in this step two more intermediate steps which are not shown in Fig. 3.12 are performed. First, the skin pixel with the smallest row index m and the skin pixel with the smallest column index n are searched. If one of those two skin pixels is not located in the quadratic RoI then it is moved accordingly to the top or to the left.

In Fig. 3.13 four different RGB images out of the verification data set are shown. Each image shows a different desktop scene, different grasps as well as different objects. In Fig. 3.13(a) the hand without interacting with an object can be seen; whereas, the remaining three images show hand-object interactions. In Fig. 3.13(b) and Fig. 3.13(d) skin colored objects are grasped, while in Fig. 3.13(c) hand interaction with a blue box is shown. Figure 3.13 shows that the described scheme in this section for RoI extraction works well and can be used to find a suitable RoI to compensate for the missing information of the hand position. However, this scheme only works for a labeled data set captured from an ego-centric camera position showing the right hand. The scheme works for left hand as well but it has to be modified accordingly.


(a) No hand-object interaction


(b) Hand-object interaction


(c) Hand-object interaction

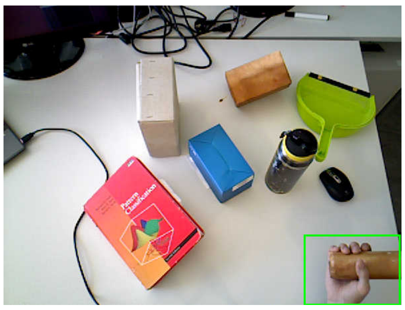

(d) Hand-object interaction

Figure 3.13: The figure shows four different examples of the verification data set from different scenes. The extracted RoIs with proposed scheme is shown by the green rectangle in each image.

Chapter 4

Experimental Results

This chapter presents the results of four experiments. Each experiment is introduced by describing the details. The results are compared and interpreted with the F_1-score from (3.22) as performance metric. The ground truth masks contain invalid pixels due to invalid depth measurements during the ground truth estimation. These invalid mask pixels are excluded from the computation of the F_1-score because they distort the result if we assign them to either non-skin or skin pixels. Thereby, the F_1-score is computed by taking into account only valid pixels. The standard parameterization is given in Sec. 3.3. The experiments compare the classification performance of RFs with different feature vectors such as a vector containing depth features only, color features only, or depth and color features together. Thus, for each experiment a set of different RFs is trained. In the following, necessary notations are introduced to ease the discussion of this chapter. Henceforth, we denote a RF as f_b^a, where $a \in [\text{depth}, \text{color}, \text{both}, \text{subset}]$ describes the feature vector used for training and classification. Whereas, *depth* shows that the vector contains depth features only, *color* reflects the usage of color features only, *both* means that the feature vector contains the full feature set of depth and color features and *subset* stands for a feature vector containing the 60 best features. The feature subset contains both depth and color features. The type of the experiment is denoted by $b \in [\text{EI}, \text{RoI}, \text{PP}]$, where *EI*, *RoI* and *PP* relate to "entire image", "region of interest" and "post-processing". We define a set of RFs per experiment and the set depends on the type of experiment. A set contains four different RFs at maximum. The set is defined as

$$\mathcal{F}_b = \left[f_b^{\text{depth}}, f_b^{\text{color}}, f_b^{\text{both}}, f_b^{\text{subset}}\right], \tag{4.1}$$

however $b = \text{PP}$ is not a valid description for a RF because post-processing does not change the RF. The RFs are used to classify the entire image or the image's RoI and each pixel is separately processed. For each pixel the number of trees inside the RF voting for skin are summed up, the resulting number is in the range of $[0, 1, 2, \ldots, 60]$. This number is stored at the same pixel position in a matrix which is called probability map and defined as

$$\mathbf{P} = \left\{ P_{mn} \mid P_{mn} \in [0, 1, \ldots, 60], 0 \leq m < M, 0 \leq n < N \right\}, \tag{4.2}$$

where M and N describe height and width of the images. The resulting probability maps of the different RFs and experiments are collected in the set

$$\mathcal{P}_b = \left[\mathbf{P}_b^{\text{depth}}, \mathbf{P}_b^{\text{color}}, \mathbf{P}_b^{\text{both}}, \mathbf{P}_b^{\text{subset}} \right], \tag{4.3}$$

where the different sets do not have to contain all feature vector types a. The mask used for segmentation is obtained from the probability map by thresholding and is called segmentation result. The segmentation result is defined as

$$\mathbf{R} = \{ R_{mn} \mid R_{mn} = [0, 1], 0 \leq m < M, 0 \leq n < N \} \tag{4.4}$$

and the elements of $\mathbf{R}$ are obtained by

$$R_{mn} = \begin{cases} 0 & \text{if} \quad P_{mn} \leq t_{\text{skin}} \\ 1 & \text{if} \quad P_{mn} > t_{\text{skin}} \,. \end{cases} \tag{4.5}$$

The segmentation results identify which pixels in the image are classified as skin and which as non-skin. The obtained segmentation results of the different feature vector types are collected in the set

$$\mathcal{R}_b = \left[\mathbf{R}_b^{\text{depth}}, \mathbf{R}_b^{\text{color}}, \mathbf{R}_b^{\text{both}}, \mathbf{R}_b^{\text{subset}} \right]. \tag{4.6}$$

The set does not have to contain segmentation results of all feature vector types in each experiment, similar as for (4.3). The used thresholds for the thresholding operation in (4.5) are collected in the set

$$\mathcal{T}_b = \left[t_b^{\text{depth}}, t_b^{\text{color}}, t_b^{\text{both}}, t_b^{\text{subset}} \right], \tag{4.7}$$

while each threshold $t_b^a \in [0, 1, \ldots, 60]$.

In the description of the experiments we often refer to terms such as *threshold evaluation*, *chosen threshold*, *chosen F_1-score* and *optimal F_1-score*. Our understanding of the aforementioned terms is defined in the following.

Threshold evaluation: A threshold evaluation describes the process of computing the average F_1-score for each possible threshold value across the entire data set. The thresholds at which the average F_1-score is computed are in the range of $[0, 1, \ldots, 60]$. In general, a threshold evaluation is performed on each data set for each RF, if needed.

Chosen threshold: The chosen threshold is the threshold which results in the best average F_1-score obtained by a threshold evaluation on the training data set. The chosen threshold is different for each RF; therefore, a threshold evaluation on the training data set is performed for each RF.

Chosen F_1-score: This term refers to the average F_1-score obtained for the chosen threshold on the verification data set or another data set which is not the training data set. The chosen F_1-score is called $\overline{F}^{a,b}_{1,\text{chosen}}$.

Optimal F_1-score: This term refers to the maximal F_1-score achievable on the verification data set or another data set which is not the training data set. The value is obtained with a threshold evaluation and is called $\overline{F}^{a,b}_{1,\text{opt}}$.

The presentation of the experiment results is structured as follows. First, Sec. 4.1 shows the segmentation results achieved for the RF set $\mathcal{F}_{\text{EI}}$ which is trained on the entire image. Next, Sec. 4.2 compares the performance of $\mathcal{F}_{\text{RoI}}$ with the result of $\mathcal{F}_{\text{EI}}$ inside the RoI. The RFs $\mathcal{F}_{\text{RoI}}$ are trained on the RoI. Third, the RFs $\mathcal{F}_{\text{RoI}}$ are evaluated on different data sets which show extreme cases. The experiments one to three do not apply any post-processing and the segmentation results shown are raw results. Finally, experiment four shows the effects of a post-processing step.

4.1 Experiment 1: Entire Image

The RF set $\mathcal{F}_{\text{EI}}$, see (4.1), is trained using the entire image and the set of segmentation results $\mathcal{R}_{\text{EI}}$, see (4.6), is generated. However, both sets do not contain the elements of the feature subset, $f^{\text{subset}}_{\text{EI}}$ and $\mathbf{R}^{\text{subset}}_{\text{EI}}$. The configuration of the RFs is as described in Sec. 3.3.

Table 4.1 summarizes the results of the threshold evaluation performed on the verification data set. However, the chosen thresholds $\mathcal{T}_{\text{EI}}$ are obtained by a threshold evaluation on the training data set and are given in Tab. 3.5. The F_1-score $\overline{F}^{a,\text{EI}}_{1,\text{opt}}$ denotes the maximum average F_1-score which can be achieved on the verification data set. Whereas, $\overline{F}^{a,\text{EI}}_{1,\text{chosen}}$ denotes the F_1-score achieved on the verification data set if the threshold set $\mathcal{T}_{\text{EI}}$ is applied. The comparison between optimal and chosen threshold shows that the scheme for obtaining $\mathcal{T}_{\text{EI}}$ provides a good hint of the optimal threshold. Furthermore, the F_1-scores $\overline{F}^{a,\text{EI}}_{1,\text{chosen}}$ shows that

- if color features only are employed the worst segmentation performance is obtained.
- if depth features only are used the F_1-score is more than 10 % higher than for color features only.
- if both feature types are applied the best segmentation performance is obtained.

The threshold investigation on the entire image shows that our proposed feature vector (3.21) achieves the best performance.

Type	$a = \text{depth}$	$a = \text{color}$	$a = \text{both}$
$\overline{F}^{a,\text{EI}}_{1,\text{opt}}$	85.0 %	72.4 %	86.6 %
$\overline{F}^{a,\text{EI}}_{1,\text{chosen}}$	83.8 %	72.4 %	86.5 %
t^a_{EI}	49 votes	56 votes	53 votes

Table 4.1: The average F_1-scores $\overline{F}^{a,\text{EI}}_{1,\text{opt}}$ and $\overline{F}^{a,\text{EI}}_{1,\text{chosen}}$ obtained from a threshold evaluation are listed in the table. The values are generated with RF set $\mathcal{F}_{\text{EI}}$. The thresholds t^a_{EI} are obtained by the threshold evaluation of the training data set in Sec. 3.3.2.

In Tab. 4.1 the numerical evaluation of the segmentation performance of the RF set $\mathcal{F}_{\text{EI}}$ is shown, while Fig. 4.1 shows the segmentation results for a defined set of four different scenes. The scenes are equal to those in Fig. 3.13 and show different hand-object interaction situations. The first scene shows an example of no hand-object interaction, while the scenes two to four show hand-object interaction. In particular, scene two shows the grasp of the skin colored box Fig. 3.3(f), scene three the interaction of the hand with a blue box and the fourth scene shows the clasp of the skin colored tin Fig. 3.3(e). The first two rows show the color image and the ground truth mask of the example scenes. The rows three to five show the set of segmentation results $\mathcal{R}_{\text{EI}}$ of the four scenes. The segmentation results of the different RFs are colored differently, where blue refers to depth features only, red to color features only and yellow to all features; furthermore, beneath each segmentation result the corresponding F_1-score is displayed. The inspection of all segmentation results show that all of them contain blobs misclassified as skin. The dedicated observations of the examples in Fig. 4.1 are described in the following.

Example 1: Although the skin colored tin is detected as skin if all features are used the printed hand in the background is not segmented at all. The print out of a hand does only mislead the RF using the color features only. The lower F_1-score of $f^{\text{both}}_{\text{EI}}$ compared to $f^{\text{depth}}_{\text{EI}}$ results from the larger areas misclassified as skin, in particular the skin colored tin.

Example 2: Using depth features only leads to misclassifications at the thumb or the upper part of the box. Although the hand interacts with a skin colored object the RF using color information only learned the differences between hand and object very well. The best result is achieved with the RF using both color and depth information.

Example 3: The segmentation result of the RF using color features only shows obvious issue if the classifier considers only color information. Interestingly, a lot of the segmented areas do not show skin colors. It seems that a specific shade of grey causes the misclassification. Furthermore, the pixels of the wrist and the back of the hand are not segmented as skin which may happen because this area is shadowed. In case of depth features only the RF does not achieve a good discrimination between object and hand. In case of objects which are not skin colored this issue is almost completely removed by applying both depth and color features. However, as in the case of color features only the area of the wrist or back of the hand shows misclassifications. Note, that the segmentation result of $f_{\mathrm{EI}}^{\mathrm{both}}$ in contrast to that of $f_{\mathrm{EI}}^{\mathrm{depth}}$ shows an increased misclassification of skin colored objects and thus the F_1-score is worse.

Example 4: This scene shows the problems of the trained RFs because each feature combination fails in this particular example. Although most or all skin pixels are classified as skin for depth features only and both feature types, the object is also segmented as skin. As a result, the differentiation of skin and object could not be achieved. This example indicates that the training data set does not contain enough images of such situations.

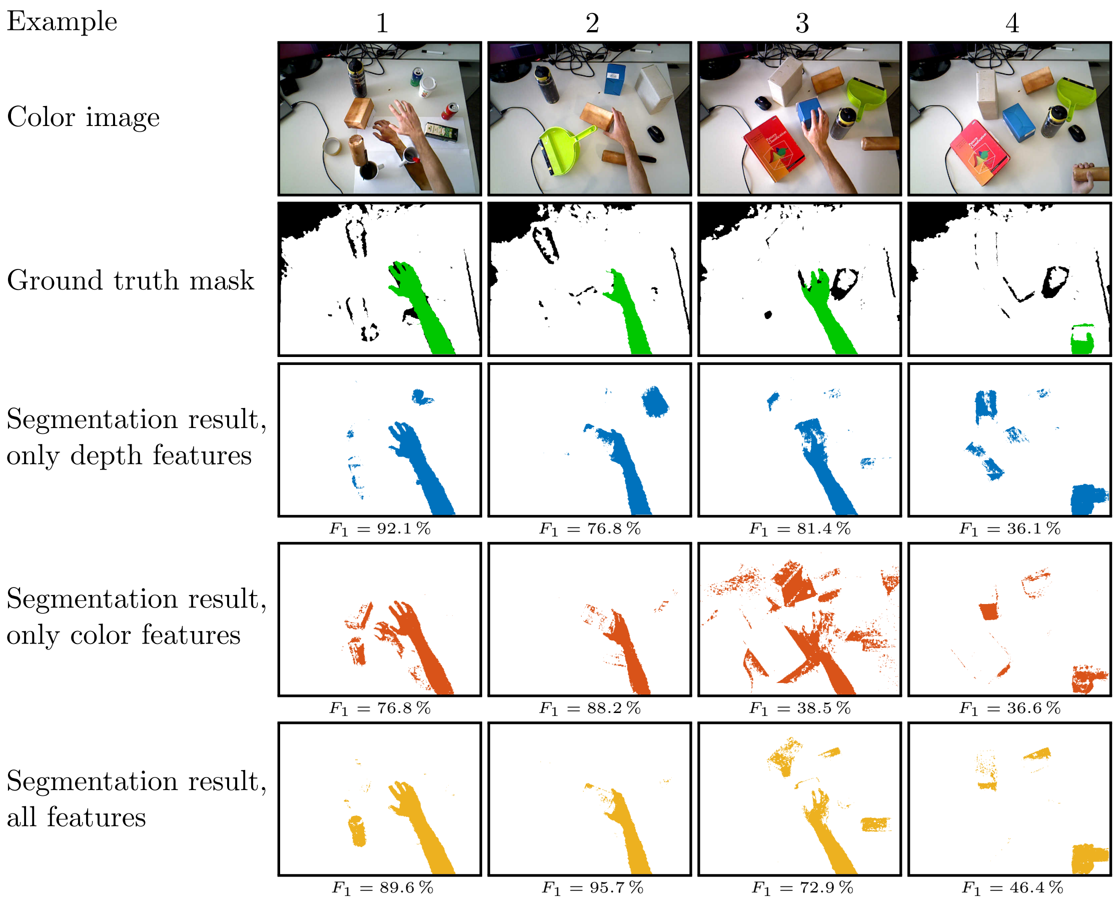


Figure 4.1: Segmentation results for different situations using the RFs trained on the entire image. Example 1 shows the hand without object interaction, example 2 shows a grasp of the skin colored box (Fig. 3.3(f)), in example 3 a blue box is grasped while in example 4 the skin colored tin (Fig. 3.3(e)) is grasped. The first row shows the color image and the second row the ground truth mask (green). The black pixels in the ground truth mask indicate invalid mask pixels due to invalid depth measurements. In rows three, four and five the segmentation results of the RFs respectively for depth only (blue), color only (red) and all features (yellow) are displayed. The colored pixels, green, blue, red and yellow, in the ground truth mask and segmentation results identify the skin pixels. The F_1-score of the segmentation results is written beneath them.

4.1.1 Discussion of Experiment 1

Most of the segmentation results in Fig. 4.1 show that there are many areas where objects are misclassified as skin. Furthermore, the examples show that the subtle differences between object and hand at the area of contact are not learned by the RFs. These problems arise from the fact that the RF is trained on the entire image without any spatial restriction of the training samples. The training scheme, compare Sec. 2.4.2, randomly draws 1000 non-skin pixels and 1000 skin pixels per image in the training set. We apply a uniform distribution because we do not put any restrictions on the spatial distribution of the training samples. However, the results indicate that the extraction of a RoI can improve the discrimination of the difficult areas of contact between hand and object because the region where the training samples are drawn from is spatially restricted. Thereby, more samples showing the difficult regions are used to train the RF. In our case the RoI contains the region which shows the hand because this is the interesting region.

Nevertheless, the visual inspection applying both depth and color features seems to achieve the best performance. This observation is confirmed by the numerical values of the threshold evaluation in Fig. 4.1. The examples generally show that if $\mathbf{R}_{\mathrm{EI}}^{\mathrm{depth}}$ and $\mathbf{R}_{\mathrm{EI}}^{\mathrm{color}}$ are merged, $\mathbf{R}_{\mathrm{EI}}^{\mathrm{both}}$ is obtained. This explains the increased number of misclassified blobs in example 3 of Fig. 4.1.

In summary, both the numerical evaluation of the F_1-score for the different feature combinations and the visual inspection of the segmentation results show that employing both depth and color information for classification achieves the best performance for the setup of this experiment. However, the results show that the differences between object and hand are not learned very well by the RF and thus, learning on the RoI and classifying the RoI might improve the results. Considering the RoI also reduces the computational expenses, because the classification is performed on a small subset of all pixels in the image.

4.2 Experiment 2: Region of Interest

In this experiment the RoI of each image in the training set is used to train another set of RF classifiers $\mathcal{F}_{\mathrm{RoI}}$. The configuration of the RF is as described in Sec. 3.3. The training of $\mathcal{F}_{\mathrm{RoI}}$ follows the same scheme as for the RF set $\mathcal{F}_{\mathrm{EI}}$. The training samples contain 1000 non-skin and 1000 skin pixels of each image in the training set. However, these pixels are drawn from the RoI. The set of segmentation results $\mathcal{R}_{\mathrm{RoI}}$ for each image in the verification set are generated and used to evaluate $\mathcal{F}_{\mathrm{RoI}}$. In this section, the numerical values of the segmentation performances of $\mathcal{F}_{\mathrm{RoI}}$ are compared with $\mathcal{F}_{\mathrm{EI}}$. In contrast to Sec. 4.1, we investigate another RF trained on the RoI which only applies a subset of all available features, called $f_{\mathrm{RoI}}^{\mathrm{subset}}$.

In order to be able to compare the RFs $\mathcal{F}_{\text{EI}}$ and $\mathcal{F}_{\text{RoI}}$, the probability maps $\mathcal{P}_{\text{EI}}$ of each image in the verification data set are evaluated inside the RoI. This means that the RoI is extracted from the probability maps $\mathcal{P}_{\text{EI}}$ of each image in the verification data set. Then the threshold evaluation is performed on the extracted RoIs. The threshold sets $\mathcal{T}_{\text{EI}'}$ and $\mathcal{T}_{\text{RoI}}$ are obtained by using the same scheme on the training data set and finding the threshold resulting in the best average F_1-score for the training data set. The identified threshold sets are given in Tab. 4.2.

The identification of the feature subset for the RF $f_{\text{RoI}}^{\text{subset}}$ is done by choosing the best 60 features in the proposed feature vector. The measure variable importance $\mathbf{w}^{\text{imp}}$, compare Sec. 2.4.2, is used to detect the best features. Figure 4.2 shows the scaled variable importance of the RF $f_{\text{RoI}}^{\text{both}}$ as a solid yellow line. The threshold which is used to obtain the 60 most important features, is displayed as dashed yellow line. Each feature with a higher $\mathbf{w}_{\text{scaled}}^{\text{imp}}$ is part of the feature subset. The variable importance is rescaled in order for the feature with the highest $\mathbf{w}^{\text{imp}}$ to match 100 %. As can be seen in Sec. 3.2, the features with numbers 1 to 129 comprise the color information and the ones with numbers from 130 to 229 comprise the depth information. The results in Sec. 4.1 show that the segmentation results if only depth features are used achieve a higher F_1-score than if only color features are used. The plot of the variable importance in Fig. 4.2 confirms this observation because in general the features from the color information are less important than features from depth information. In contrast to the conclusions in [LK13], our results show that all features from the Gabor filter bank are part of the least important features. The numbers of the Gabor features are within the range of 82 to 129. Furthermore, for our training data set it appears that the CIELAB color space is the most discriminant color space for detecting skin (features 55 to 81). In particular all 9 "a^*" components of the color space are contained in the feature subset. This component spans the range of red-green colors, as described in Sec. 2.1.3. The obtained feature subset contains 13 color features and 47 depth features.

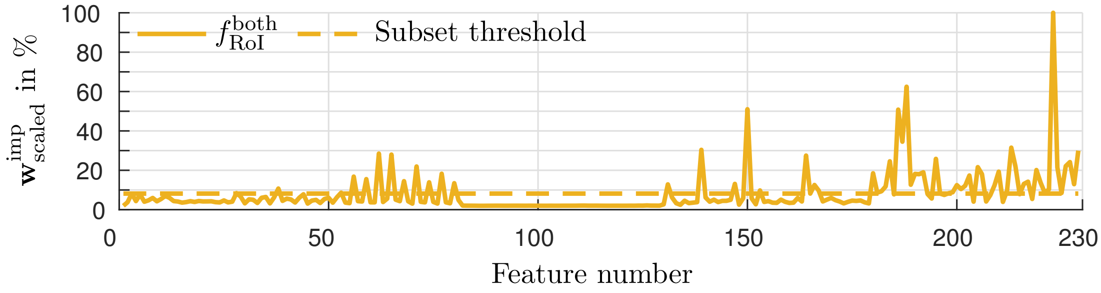


Figure 4.2: Variable importance of the RF trained on the RoI using both depth and color features. The scaled variable importance $\mathbf{w}_{\text{scaled}}^{\text{imp}}$ is obtained by rescaling all features such that for the most important feature $\mathbf{w}_{\text{scaled}}^{\text{imp}} = 100\,\%$ holds. The solid yellow line shows the scaled variable importance of the RF $f_{\text{RoI}}^{\text{both}}$ and the dashed yellow line indicates the threshold used to define the subset. The features 1 to 129 comprise the color features and the features 130 to 229 comprise the depth features.

Table 4.2 summarizes the optimal and chosen F_1-scores of the threshold evaluations. The top half of the table shows the results and $\mathcal{T}_{\mathrm{EI}'}$ for the case of evaluating the RoI in the segmentation results for the entire image. The rightmost column of the upper half of the table is empty because no RF using the feature subset is trained on the entire image. In the lower half of the table the results and $\mathcal{T}_{\mathrm{RoI}}$ of the RF set $\mathcal{F}_{\mathrm{RoI}}$ can be seen. The numbers in Tab. 4.2 emphasize that the training on the RoI improves the segmentation performance on average by more than 2 %. The lower half of the table shows that the maximum drop in F_1-score between optimal and chosen threshold is 1.3 % and occurs for the RF $f_{\mathrm{RoI}}^{\mathrm{depth}}$. Surprisingly, RF $f_{\mathrm{RoI}}^{\mathrm{depth}}$ achieves a better optimal performance than $f_{\mathrm{RoI}}^{\mathrm{both}}$. However, the performance of $f_{\mathrm{RoI}}^{\mathrm{both}}$ is better than $f_{\mathrm{RoI}}^{\mathrm{depth}}$ for the chosen thresholds $t_{\mathrm{RoI}}^{\mathrm{both}}$ and $t_{\mathrm{RoI}}^{\mathrm{depth}}$. Furthermore, comparing $\overline{F}_{1,\mathrm{chosen}}^{\mathrm{both,RoI}}$ and $\overline{F}_{1,\mathrm{chosen}}^{\mathrm{subset,RoI}}$ shows that the reduction in the feature set worsens the segmentation quality by 0.1 %. Moreover, $f_{\mathrm{RoI}}^{\mathrm{subset}}$ still performs better than $f_{\mathrm{RoI}}^{\mathrm{depth}}$ for the chosen thresholds.

Type	a = depth	a = color	a = both	a = subset
$\overline{F}_{1,\mathrm{opt}}^{a,\mathrm{EI}'}$	91.2 %	84.3 %	92.6 %	-
$\overline{F}_{1,\mathrm{chosen}}^{a,\mathrm{EI}'}$	**90.8 %**	**84.3 %**	**92.6 %**	-
$t_{\mathrm{EI}'}^{a}$	42 votes	43 votes	39 votes	-
$\overline{F}_{1,\mathrm{opt}}^{a,\mathrm{RoI}}$	95.4 %	85.6 %	95.1 %	94.9 %
$\overline{F}_{1,\mathrm{chosen}}^{a,\mathrm{RoI}}$	**94.1 %**	**85.6 %**	**94.7 %**	**94.6 %**
t_{RoI}^{a}	47 votes	39 votes	41 votes	43 votes

Table 4.2: The table summarizes the results from the threshold evaluation. The top half shows the results for $\mathcal{F}_{\mathrm{EI}}$ evaluated on the RoI and the bottom half shows the results for $\mathcal{F}_{\mathrm{RoI}}$. Besides, the F_1-scores for optimal and chosen threshold of both RF sets, the corresponding skin detection thresholds $\mathcal{T}_{\mathrm{EI}'}$ and $\mathcal{T}_{\mathrm{RoI}}$ are given. The rightmost column of the upper half of the table is empty because no RF using the feature subset is trained on the entire image.

In Fig. 4.3 four different example scenes are shown, the scenes are the same as in Fig. 4.1. However, Fig. 4.3 only shows the RoI of the color images, ground truth mask and segmentation results. The first example shows a hand without object interaction above the desktop scene; whereas, example scenes two to four show object interaction scenarios. In example two the skin colored box from Fig. 3.3(f), in example three a blue box and in example four the skin colored tin from Fig. 3.3(e) is grasped. The figures show a comparison of the segmentation results obtained by the RFs $\mathcal{F}_{\mathrm{RoI}}$. The first row shows the RoI of the scene's color image and the second row displays the RoI of the ground truth mask. The black pixels in the ground truth mask identify pixels with invalid depth measurements, white pixels mark non-skin pixels and green pixels indicate pixels showing skin. The rows three to six show the segmentation results of the RFs $\mathcal{F}_{\text{RoI}}$. The blue, red, yellow and purple colored pixels indicate pixels classified as skin. Beneath each of the segmentation results the achieved F_1-score is written. The blue mask in row three is obtained with RF $f_{\text{RoI}}^{\text{depth}}$ employing depth information only, the red mask in the next row results if only color features are applied. The yellow in row five and purple in row five show the result if both depth and color information are used. The yellow mask is obtained if all available features are used and the purple results from employing the feature subset. The displayed segmentation results are generated by applying the thresholds $\mathcal{T}_{\text{RoI}}$ to the corresponding classification output $\mathcal{P}_{\text{RoI}}$.

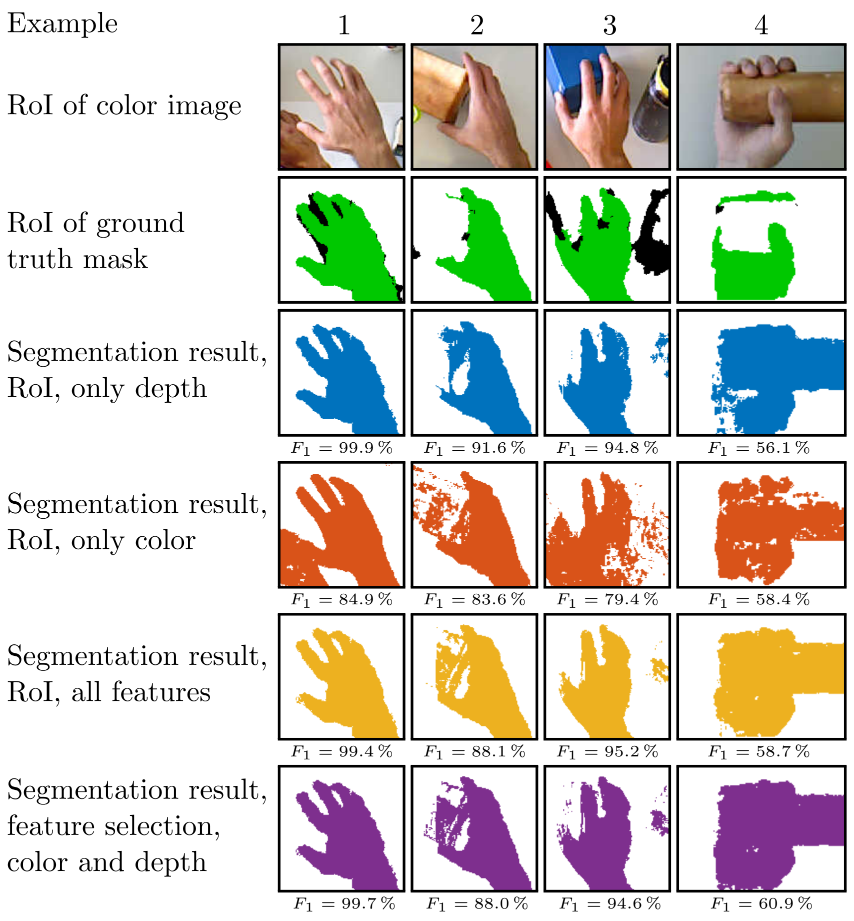


Figure 4.3: The figure shows segmentation results of the RFs $\mathcal{F}_{\text{RoI}}$. Example 1 shows the segmentation result without object interaction, example 2 shows a grasp of the skin colored box (Fig. 3.3(f)), in example 3 a blue box is grasped while in example 4 the skin colored tin (Fig. 3.3(e)) is grasped. The first row shows the color image and the second row the ground truth mask (green). The black pixels in the ground truth mask indicate invalid mask pixels due to invalid depth measurements. The white and green pixels indicate non-skin and skin pixels. The segmentation masks resulting from using only depth features (blue), only color features (red), all features (yellow) and feature subset (purple) are shown respectively in rows three, four, five and six. The corresponding F_1-score is written beneath each segmentation result.

The visual inspection of Fig. 4.3 shows that the RF employing the color information only does not show good results. Therefore, classifying based on the color information alone is not robust enough to be applied in cluttered or complex desktop scenes. The classifier is wrong if the color is similar to the skin color and even in case of certain shades of grey, as example 3 shows. All four classifiers are not able to distinguish between skin and object in example 4. The reason for this is the lack of training data showing this type of grasp. In the following example 1, 2 and 3 are discussed in more detail.

Example 1: The RFs $f_{\text{RoI}}^{\text{depth}}$, $f_{\text{RoI}}^{\text{both}}$ and $f_{\text{RoI}}^{\text{subset}}$ perform very well, while $f_{\text{RoI}}^{\text{color}}$ is mislead by the print-out of the hand on the desk. It seems that the RFs using depth features perform well because the scene shows the hand above the desk and there is no hand-object interaction. Thus, the classification task is reduced to a simple foreground background segmentation.

Example 2: All four RFs do have the problem of distinguishing skin and object. At least for the classifiers using both color and depth information this might result from the color of the box. Although $f_{\text{RoI}}^{\text{depth}}$ shows the same problems the extent of misclassification is less compared to the other classifiers. Consequently, even if only depth features are used the difference between object and hand is hard to distinguish in certain situations. Nevertheless, the achieved F_1-score is around 90 %.

Example 3: The quality of the segmentation of the RFs $f_{\text{RoI}}^{\text{depth}}$, $f_{\text{RoI}}^{\text{both}}$ and $f_{\text{RoI}}^{\text{subset}}$ appear to be equivalent. The result of the RF using only color information does contain a lot of areas falsely detected as skin.

The Fig. 4.4, 4.5 and 4.6 show more examples of the segmentation performance of the RFs. The structure of the figures is as follows. The first row shows the RoI of the RGB image. The even rows show the color image which is masked with the segmentation result from the next row. The rows two and three, four and five, six and seven, eight and nine show the results obtained by the RF $f_{\text{RoI}}^{\text{depth}}$ (blue), $f_{\text{RoI}}^{\text{color}}$ (red), $f_{\text{RoI}}^{\text{both}}$ (yellow) and $f_{\text{RoI}}^{\text{subset}}$ (purple).

In the following we discuss some of the examples of Fig. 4.4 in more detail. The example in the third column shows that the RF $f_{\text{RoI}}^{\text{depth}}$ struggles to classify parts of the arm, while $f_{\text{RoI}}^{\text{color}}$ does not misclassify parts of the arm but misclassifies the skin colored object as skin. On the other hand, the RFs $f_{\text{RoI}}^{\text{both}}$ and $f_{\text{RoI}}^{\text{subset}}$ overcome the problems of the other two RFs. The example in the eighth column shows that incorporating both depth and color information improves the quality even if the RF $f_{\text{RoI}}^{\text{color}}$ classifies a lot of non-skin pixels as skin and the RF $f_{\text{RoI}}^{\text{depth}}$ achieves a good result. Especially, the misclassification of the upper right corner of the box of $f_{\text{RoI}}^{\text{depth}}$ is reduced by applying all features. On the other hand, RF $f_{\text{RoI}}^{\text{subset}}$ shows a larger patch of misclassification in the upper right corner of the box because only 13 color features are included in the feature subset.

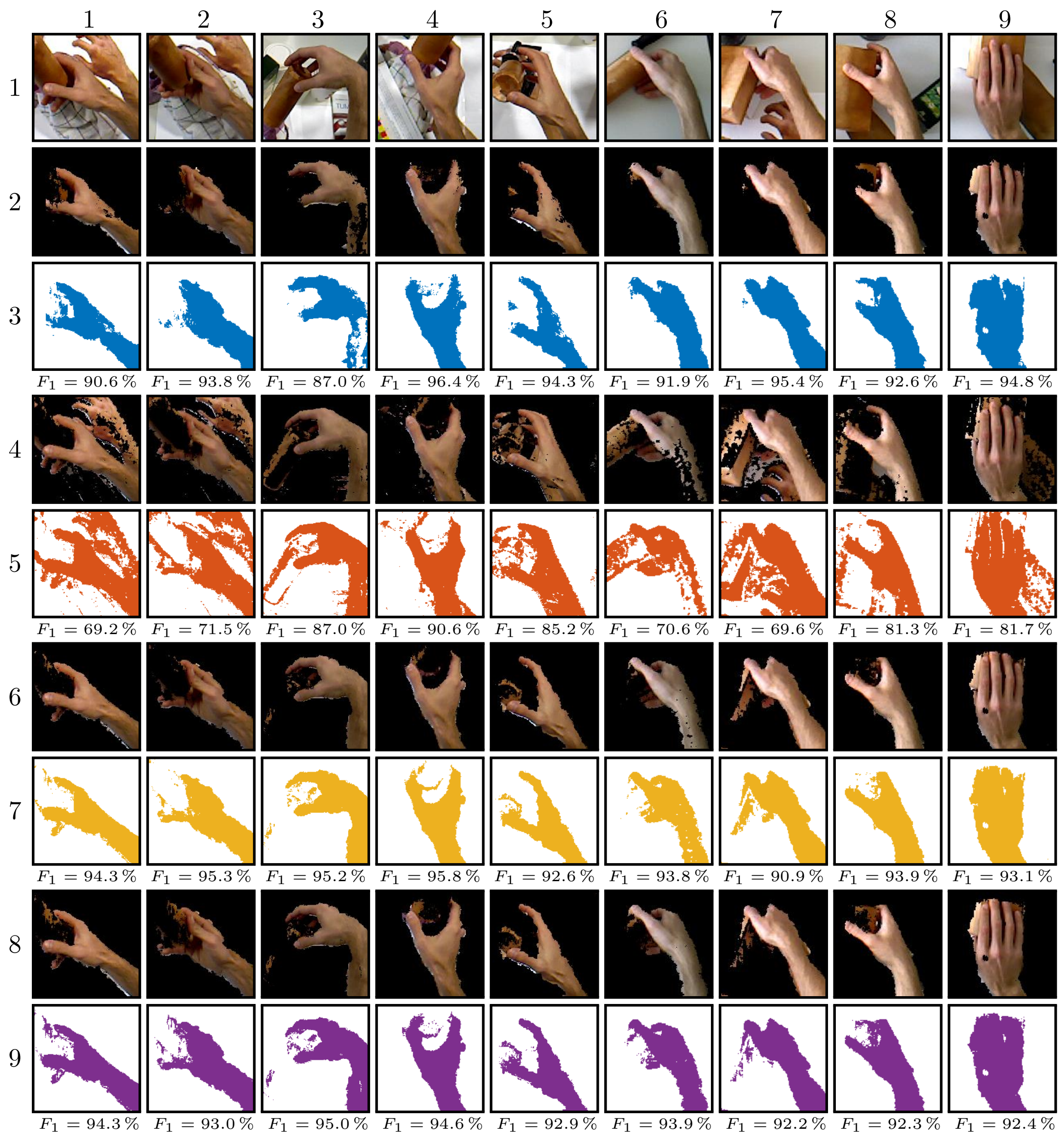

1 2 3 4 5 6 7 8 9
1
2
3
$F_1 = 90.6\,\%$ $F_1 = 93.8\,\%$ $F_1 = 87.0\,\%$ $F_1 = 96.4\,\%$ $F_1 = 94.3\,\%$ $F_1 = 91.9\,\%$ $F_1 = 95.4\,\%$ $F_1 = 92.6\,\%$ $F_1 = 94.8\,\%$
4
5
$F_1 = 69.2\,\%$ $F_1 = 71.5\,\%$ $F_1 = 87.0\,\%$ $F_1 = 90.6\,\%$ $F_1 = 85.2\,\%$ $F_1 = 70.6\,\%$ $F_1 = 69.6\,\%$ $F_1 = 81.3\,\%$ $F_1 = 81.7\,\%$
6
7
$F_1 = 94.3\,\%$ $F_1 = 95.3\,\%$ $F_1 = 95.2\,\%$ $F_1 = 95.8\,\%$ $F_1 = 92.6\,\%$ $F_1 = 93.8\,\%$ $F_1 = 90.9\,\%$ $F_1 = 93.9\,\%$ $F_1 = 93.1\,\%$
8
9
$F_1 = 94.3\,\%$ $F_1 = 93.0\,\%$ $F_1 = 95.0\,\%$ $F_1 = 94.6\,\%$ $F_1 = 92.9\,\%$ $F_1 = 93.9\,\%$ $F_1 = 92.2\,\%$ $F_1 = 92.3\,\%$ $F_1 = 92.4\,\%$

Figure 4.4: The figure shows the segmentation results of different grasping positions for two different skin colored objects. The first row shows the RoI of the color image. In rows two and three the results of the RF $f_{\text{RoI}}^{\text{depth}}$, in rows four and five the results of $f_{\text{RoI}}^{\text{color}}$, in rows six and seven results of $f_{\text{RoI}}^{\text{both}}$ and in rows eight and nine results of $f_{\text{RoI}}^{\text{subset}}$ are shown. The segmentation results $\mathbf{R}_{\text{RoI}}^{\text{depth}}$, $\mathbf{R}_{\text{RoI}}^{\text{color}}$, $\mathbf{R}_{\text{RoI}}^{\text{both}}$ and $\mathbf{R}_{\text{RoI}}^{\text{subset}}$ are respectively shown in row three, five, seven and nine. The F_1-score corresponding to the segmentation result is written beneath it. The color image from row one is masked with the segmentation results $\mathcal{R}_{\text{RoI}}$ and displayed in rows two, four, six and eight.

Figure 4.5 shows hand-object interaction with objects which are not skin colored. The example in column three shows that $f_{\text{RoI}}^{\text{depth}}$ does not detect the thumb at all; whereas, $f_{\text{RoI}}^{\text{color}}$ detects the entire hand very well. The RFs $f_{\text{RoI}}^{\text{both}}$ and $f_{\text{RoI}}^{\text{subset}}$ do not achieve the quality of $f_{\text{RoI}}^{\text{color}}$ but the performance of the RF using depth features clearly is improved. A similar situation is shown in the seventh column. The

RF $f_{\mathrm{RoI}}^{\mathrm{depth}}$ does not detect the thumb or the fingers, while the other three RFs detect those parts more clearly. However, the other three RFs misclassify a small part of the box as skin.

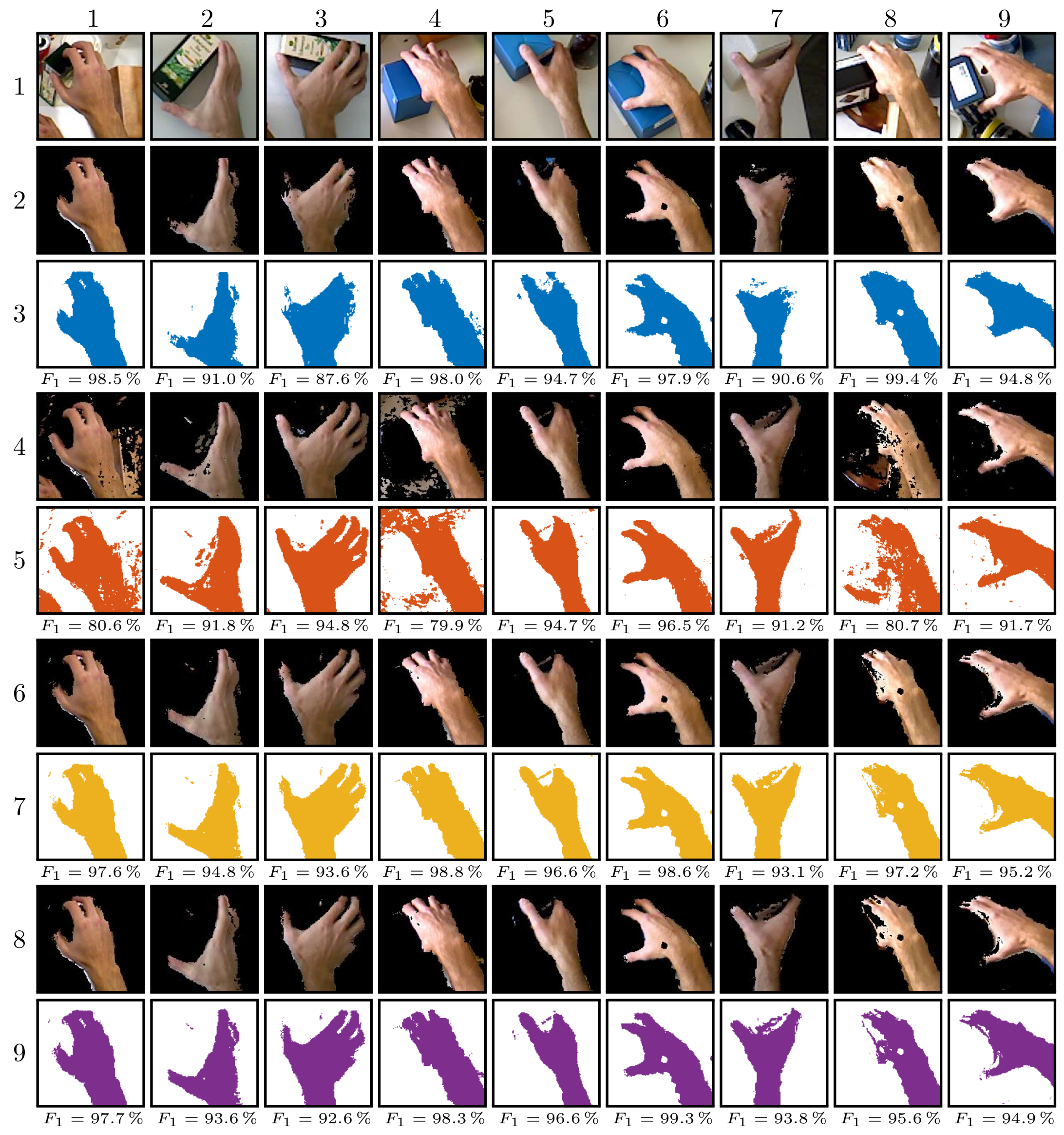

1 2 3 4 5 6 7 8 9
1
2
3
$F_1 = 98.5\,\%$ $F_1 = 91.0\,\%$ $F_1 = 87.6\,\%$ $F_1 = 98.0\,\%$ $F_1 = 94.7\,\%$ $F_1 = 97.9\,\%$ $F_1 = 90.6\,\%$ $F_1 = 99.4\,\%$ $F_1 = 94.8\,\%$
4
5
$F_1 = 80.6\,\%$ $F_1 = 91.8\,\%$ $F_1 = 94.8\,\%$ $F_1 = 79.9\,\%$ $F_1 = 94.7\,\%$ $F_1 = 96.5\,\%$ $F_1 = 91.2\,\%$ $F_1 = 80.7\,\%$ $F_1 = 91.7\,\%$
6
7
$F_1 = 97.6\,\%$ $F_1 = 94.8\,\%$ $F_1 = 93.6\,\%$ $F_1 = 98.8\,\%$ $F_1 = 96.6\,\%$ $F_1 = 98.6\,\%$ $F_1 = 93.1\,\%$ $F_1 = 97.2\,\%$ $F_1 = 95.2\,\%$
8
9
$F_1 = 97.7\,\%$ $F_1 = 93.6\,\%$ $F_1 = 92.6\,\%$ $F_1 = 98.3\,\%$ $F_1 = 96.6\,\%$ $F_1 = 99.3\,\%$ $F_1 = 93.8\,\%$ $F_1 = 95.6\,\%$ $F_1 = 94.9\,\%$

Figure 4.5: The figure shows the segmentation results of different grasping positions for five different boxes, the boxes differ in size and color. The structure of the figure is equal to Fig. 4.4.

The eighth column in Fig. 4.6 show as similar issue as the third and seventh column in Fig. 4.5. The RF $f_{\mathrm{RoI}}^{\mathrm{depth}}$ struggles to segment the fingers as skin. Employing the color information as well improves this problem as the RFs $f_{\mathrm{RoI}}^{\mathrm{both}}$ and $f_{\mathrm{RoI}}^{\mathrm{subset}}$ show. On the other hand, the example in column four shows that if only color features are used the RF is not robust enough to reliably detect the skin pixels. In this case the RFs $f_{\mathrm{RoI}}^{\mathrm{both}}$ and $f_{\mathrm{RoI}}^{\mathrm{subset}}$ also show that merging color and depth information helps to obtain a more reliable result. The third column shows that overexposed RGB images cause problems for all RFs using color features.

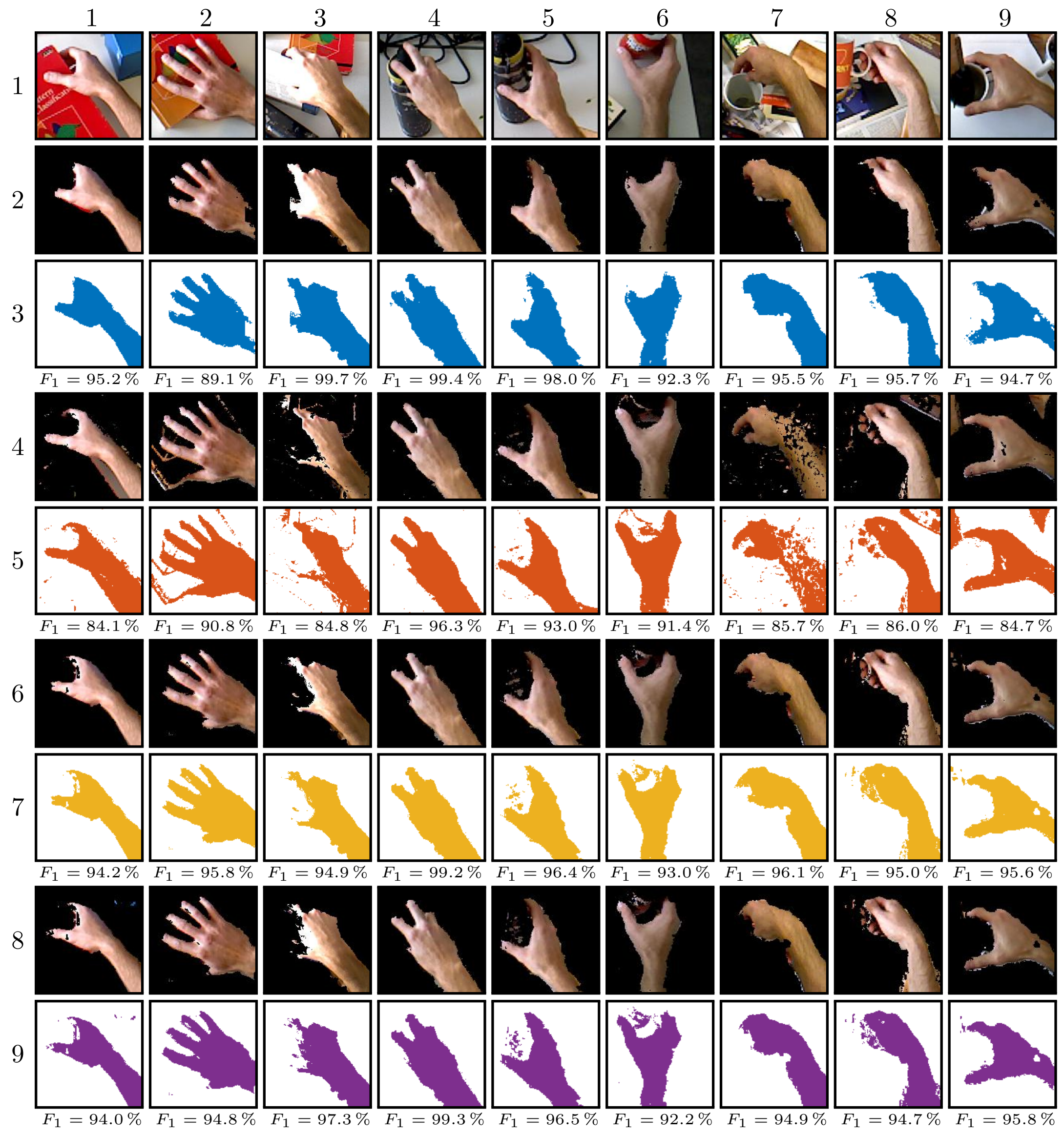

1 2 3 4 5 6 7 8 9
1
2
3
$F_1 = 95.2\,\%$ $F_1 = 89.1\,\%$ $F_1 = 99.7\,\%$ $F_1 = 99.4\,\%$ $F_1 = 98.0\,\%$ $F_1 = 92.3\,\%$ $F_1 = 95.5\,\%$ $F_1 = 95.7\,\%$ $F_1 = 94.7\,\%$
4
5
$F_1 = 84.1\,\%$ $F_1 = 90.8\,\%$ $F_1 = 84.8\,\%$ $F_1 = 96.3\,\%$ $F_1 = 93.0\,\%$ $F_1 = 91.4\,\%$ $F_1 = 85.7\,\%$ $F_1 = 86.0\,\%$ $F_1 = 84.7\,\%$
6
7
$F_1 = 94.2\,\%$ $F_1 = 95.8\,\%$ $F_1 = 94.9\,\%$ $F_1 = 99.2\,\%$ $F_1 = 96.4\,\%$ $F_1 = 93.0\,\%$ $F_1 = 96.1\,\%$ $F_1 = 95.0\,\%$ $F_1 = 95.6\,\%$
8
9
$F_1 = 94.0\,\%$ $F_1 = 94.8\,\%$ $F_1 = 97.3\,\%$ $F_1 = 99.3\,\%$ $F_1 = 96.5\,\%$ $F_1 = 92.2\,\%$ $F_1 = 94.9\,\%$ $F_1 = 94.7\,\%$ $F_1 = 95.8\,\%$

Figure 4.6: The figure shows the segmentation results of different grasping positions for various objects. The scenes include three grasps of books, three grasps of cylindrical shaped objects and three grasps of mugs. The structure of the figure is equal to Fig. 4.4 and 4.5.

The Fig. 4.4, 4.5 and 4.6 show that the segmentation quality of each RF depends on the scene. In some cases the color information provides a good cue for skin pixels, while others show that the depth features are more useful to correctly segment skin pixels. In general, the examples show that the robustness of the segmentation quality is improved by employing both depth and color information. Furthermore, reducing the number of features and applying the best 60 features does only slightly decrease the quality of the segmentation result. The figures show 27 different situations, where 16 times the RF $f_{\text{RoI}}^{\text{both}}$ performs better than the RFs $f_{\text{RoI}}^{\text{depth}}$ and $f_{\text{RoI}}^{\text{color}}$.

In Tab. 4.3 the time needed to calculate the features for the classification and the time needed for classification are summarized. The upper half of the table shows the results if the entire image is classified while the lower half shows the time consumption if only the RoI is considered. The RF $f_{\text{EI}}^{\text{subset}}$ is not trained in experiment one; thus, the corresponding cells of the table are empty. The table shows that if the classification process is restricted to the RoI then the time consumption of feature calculation and classification is reduced by a factor of approximately 10. According to the numbers in Tab. 4.3 it seems that the classification time depends on the type of feature, too. Apparently, including color features makes the classification process slower, inferring that on average more nodes have to be processed. The time consumed to classify the entire image is too much and shows that restricting the classification to the RoI is clearly more efficient. Furthermore, the RF $f_{\text{RoI}}^{\text{subset}}$ consumes on average less than 600 ms which makes it the fastest RF in the comparison. Therefore, $f_{\text{RoI}}^{\text{subset}}$ does achieve comparable segmentation quality to $f_{\text{RoI}}^{\text{both}}$, while being the most efficient RF.

Time	$a =$ depth	$a =$ color	$a =$ both	$a =$ subset	b
Feature calculation	2.08 s	1.52 s	3.60 s	-	$b =$ EI
Classification	6.73 s	8.86 s	6.84 s	-	
Feature calculation	0.155 s	0.125 s	0.278 s	0.078 s	$b =$ RoI
Classification	0.455 s	0.687 s	0.662 s	0.493 s	

Table 4.3: The table shows the average time needed to compute the employed features and the probability maps for the different RFs. The upper half of the table shows the time consumption if the entire image is classified and the lower half shows the time needed if only the RoI is classified. The RF $f_{\text{EI}}^{\text{subset}}$ is not trained, so the entries in the corresponding cells are empty. The times are acquired by averaging over the entire verification data set.

4.2.1 Discussion of Experiment 2

This experiment shows that the quality of the segmentation results is improved if the training samples are drawn from the RoI. Thereby, the importance of the training samples is increased because they reflect the subtle differences between hand and object better. Furthermore, it helps to exclude possible distractions on the desktop scene. This result is supported by the numerical comparison in Tab. 4.2. The evaluation of the examples in Fig. 4.4, 4.5 and 4.6 show that the RF generally performs better than the other RFs. However, this result is not reflected by the numbers in Tab. 4.2, especially the result of the RF only using the depth features is very close to the performance of RF $f_{\text{RoI}}^{\text{both}}$. This might result from the fact that the verification data set contains around 50 % of RGB-D images showing the hand above the desktop without object interaction. In case of considering only the RoI this is reduced to a simple background foreground detection for a RF using depth features. Thus, the numbers for RF $f_{\text{RoI}}^{\text{depth}}$ are similarly good as for $f_{\text{RoI}}^{\text{both}}$ and the numerical values do not reflect the true improvement of employing both depth and color information.

4.3 Experiment 3: Extreme Cases

The experiment in Sec. 4.2 shows that the verification data set does not reflect the advantage of employing both depth and color information compared to using depth features only. This issue arises because only 50 % of the images in the verification data set show hand-object interaction. However, we are particularly interested in hand-object interaction situations but do not want to exclude situations without object interaction. Therefore, we collected two smaller data sets which show extreme cases, where the RF using color features excels the RF using depth information and vice versa. The data set EC1 (Extreme Case Data Set 1) consists of cases where color gives a better hint than depth and EC2 (Extreme Case Data Set 2) contains cases where depth information is more useful than color. The data set EC12 (Extreme Case Data Set 1 & 2) contains both EC1 and EC2. The data sets show that the RFs $f_{\text{RoI}}^{\text{both}}$ and $f_{\text{RoI}}^{\text{subset}}$ perform similarly good as $f_{\text{RoI}}^{\text{color}}$ on the data set EC1 and $f_{\text{RoI}}^{\text{depth}}$ on data set EC2. However, the combined data set EC12 show that the RFs using both depth and color information outperform the two other RFs.

This experiment is executed with the RFs $\mathcal{F}_{\text{RoI}}$ from the experiment in Sec. 4.2. Table 4.4 shows the results obtained by the three data sets. The last column indicates which data set is used to compute the numbers. The table is divided into three parts the top part shows the results for EC1, the middle part for EC2 and the bottom part for EC12. The thresholds $\mathcal{T}_{\text{RoI}}$ are given in Tab. 4.2. In the following the quality of the segmentation results $\mathcal{R}_{\text{RoI}}$ is discussed.

EC1: The RFs $\mathcal{F}_{\text{RoI}}$ show the expected characteristics. The RF $f_{\text{RoI}}^{\text{color}}$ performs best, while $f_{\text{RoI}}^{\text{depth}}$ achieves the worst result. In fact, the RF $f_{\text{RoI}}^{\text{both}}$ does not achieve the best F_1-score but it is just 1.3 % lower while excelling $f_{\text{RoI}}^{\text{depth}}$ by more than 12 %. The classifier using both depth and color does not achieve the same performance as $f_{\text{RoI}}^{\text{color}}$ because the depth features are more important for $f_{\text{RoI}}^{\text{both}}$ in the training data set, as can be seen in Fig. 4.2. The images in EC1 contain situations where the hand is not the closest object to the camera. Such situations do not often appear in the training data set. Therefore, the RF using depth features only does not perform very well. However, this issue is considerably improved by using both color and depth information.

EC2: As expected, the RF $f_{\text{RoI}}^{\text{depth}}$ outperforms $f_{\text{RoI}}^{\text{color}}$ and the RF $f_{\text{RoI}}^{\text{both}}$ achieves the same average F_1-score as $f_{\text{RoI}}^{\text{depth}}$ in data set EC2. The F_1-score of RF $f_{\text{RoI}}^{\text{color}}$ is 16 % lower than the maximum F_1-score. Some of the images in data set EC2 for example show a print-out of the hand on the desk which misleads the RF $f_{\text{RoI}}^{\text{color}}$ but not the RFs using both depth and color features.

EC12: The combined data set EC12 containing both extreme case data sets shows that using both feature types achieve the best performance. The F_1-score of $f_{\text{RoI}}^{\text{both}}$ is 4.5 % higher than for $f_{\text{RoI}}^{\text{color}}$ and 8.1 % higher than for $f_{\text{RoI}}^{\text{depth}}$.

Table 4.4 shows that situations which are problematic to classifiers using either depth or color do not affect the RFs using both depth and color information as negatively. This result indicates that using both depth and color features clearly improves the segmentation quality of the classifier. Moreover, the table shows that the RF $f_{\text{RoI}}^{\text{subset}}$ is more affected by the extreme cases than $f_{\text{RoI}}^{\text{both}}$. This might indicate, that problematic scenes cannot be described as good with the feature subset as with the complete feature set.

Type	a = depth	a = color	a = both	a = subset	
$\overline{F}_{1,\text{opt}}^{a,\text{RoI}}$	87.3 %	95.0 %	94.7 %	92.1 %	EC1
$\overline{F}_{1,\text{chosen}}^{a,\text{RoI}}$	81.4 %	**95.0 %**	**93.7 %**	90.8 %	EC1
$\overline{F}_{1,\text{opt}}^{a,\text{RoI}}$	97.1 %	81.0 %	96.7 %	96.0 %	EC2
$\overline{F}_{1,\text{chosen}}^{a,\text{RoI}}$	**96.7 %**	80.7 %	**96.7 %**	96.0 %	EC2
$\overline{F}_{1,\text{opt}}^{a,\text{RoI}}$	90.3 %	90.4 %	95.4 %	93.5 %	EC12
$\overline{F}_{1,\text{chosen}}^{a,\text{RoI}}$	86.8 %	90.4 %	**94.9 %**	92.7 %	EC12

Table 4.4: The table gives the average F_1-score of the RF set $\mathcal{F}_{\text{RoI}}$ for the data sets EC1, EC2 and EC12 which contain extreme cases. The extreme cases are defined as cases, where $f_{\text{RoI}}^{\text{depth}}$ excels $f_{\text{RoI}}^{\text{color}}$ or vice versa. The table is divided into three parts. The top part shows the results for data set EC1, the middle part for EC2 and the bottom part for EC12. This is indicated in the last column, as well.

Figure 4.7 shows four examples of EC1 in columns one to four and four examples of EC2 in columns five to eight. The figure's structure is equal to Fig. 4.4. The examples show that the segmentation quality of $f_{\text{RoI}}^{\text{both}}$ and $f_{\text{RoI}}^{\text{subset}}$ shows similar quality as the better performing RF of $f_{\text{RoI}}^{\text{depth}}$ and $f_{\text{RoI}}^{\text{color}}$. The example in the third column clearly shows the problem of $f_{\text{RoI}}^{\text{depth}}$ if there is another object in front of the hand, even if there is no hand-object interaction. In such a case the color information provides important additional information for skin segmentation. Furthermore, the example shows that $f_{\text{RoI}}^{\text{both}}$ achieves a better performance because it uses both depth and color information. The segmentation quality for the example in the third column of RF $f_{\text{RoI}}^{\text{subset}}$ is slightly worse than compared to $f_{\text{RoI}}^{\text{color}}$ and $f_{\text{RoI}}^{\text{both}}$ but is still around 39 % better than the result of $f_{\text{RoI}}^{\text{depth}}$. The difference between $f_{\text{RoI}}^{\text{depth}}$ and $f_{\text{RoI}}^{\text{color}}$ is not as big for the example in column six but it shows the problems if the classifier only relies on color information. The result shows that $f_{\text{RoI}}^{\text{color}}$ misclassifies almost the entire box as skin which is intuitive because the color of the box is similar to the color of the skin. Nevertheless, the segmentation results of $f_{\text{RoI}}^{\text{both}}$ and $f_{\text{RoI}}^{\text{subset}}$ do not suffer from the same problem and achieve a similar quality as $f_{\text{RoI}}^{\text{depth}}$.

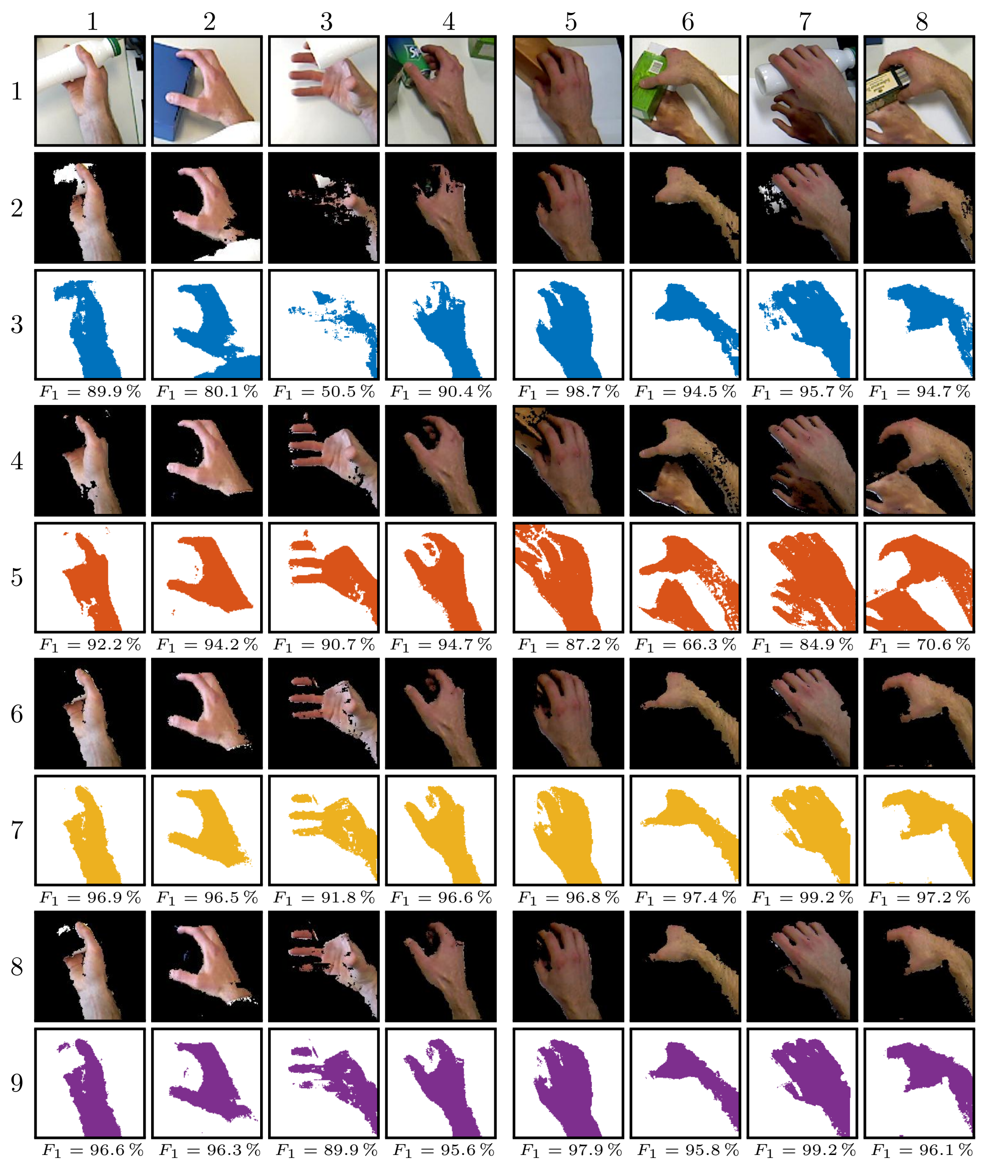

1 2 3 4 5 6 7 8
1
2
3
$F_1 = 89.9\,\%$ $F_1 = 80.1\,\%$ $F_1 = 50.5\,\%$ $F_1 = 90.4\,\%$ $F_1 = 98.7\,\%$ $F_1 = 94.5\,\%$ $F_1 = 95.7\,\%$ $F_1 = 94.7\,\%$
4
5
$F_1 = 92.2\,\%$ $F_1 = 94.2\,\%$ $F_1 = 90.7\,\%$ $F_1 = 94.7\,\%$ $F_1 = 87.2\,\%$ $F_1 = 66.3\,\%$ $F_1 = 84.9\,\%$ $F_1 = 70.6\,\%$
6
7
$F_1 = 96.9\,\%$ $F_1 = 96.5\,\%$ $F_1 = 91.8\,\%$ $F_1 = 96.6\,\%$ $F_1 = 96.8\,\%$ $F_1 = 97.4\,\%$ $F_1 = 99.2\,\%$ $F_1 = 97.2\,\%$
8
9
$F_1 = 96.6\,\%$ $F_1 = 96.3\,\%$ $F_1 = 89.9\,\%$ $F_1 = 95.6\,\%$ $F_1 = 97.9\,\%$ $F_1 = 95.8\,\%$ $F_1 = 99.2\,\%$ $F_1 = 96.1\,\%$

Figure 4.7: The figure shows segmentation results of the data sets EC1 (rows one to four) and EC2 (rows five to eight) containing extreme cases. An extreme case is defined as a situation where either $f_{\mathrm{RoI}}^{\mathrm{depth}}$ excels $f_{\mathrm{RoI}}^{\mathrm{color}}$ or vice versa. In general the structure of the figure is equal to Fig. 4.4, 4.5 and 4.6.

4.3.1 Discussion of Experiment 3

The numerical comparison in Tab. 4.4 shows that if only extreme cases are considered the RFs using both depth and color information, such as $f_{\mathrm{RoI}}^{\mathrm{both}}$ and $f_{\mathrm{RoI}}^{\mathrm{subset}}$, outperform the classifiers relying either on depth or color features only. The reason for the similar results in experiment 2 in Sec. 4.2 for RF $f_{\mathrm{RoI}}^{\mathrm{both}}$ and $f_{\mathrm{RoI}}^{\mathrm{depth}}$ could originate from the 50 % of images in the verification data set showing the hand above the scene which is easy to detect for a classifier using depth features. Another reason might be, that the verification data set does not conclude a considerable amount of extreme cases where $f_{\mathrm{RoI}}^{\mathrm{depth}}$ performs poorly. The lack of such examples results from the challenging circumstances collecting data showing these examples.

4.4 Experiment 4: Post-Processing

In this section we apply a post-processing step and investigate the resulting effects. The results in Sec. 4.2 show that using a subset of both color and depth information is sufficient to achieve a performance similar to using all 229 features. Therefore, the post-processing step is investigated on the results of $f_{\mathrm{RoI}}^{\mathrm{subset}}$. The probability maps $\mathbf{P}_{\mathrm{RoI}}^{\mathrm{subset}}$ of Sec. 4.2 are post-processed and called $\mathbf{P}_{\mathrm{PP}}^{\mathrm{subset}}$. The chosen threshold $t_{\mathrm{PP}}^{\mathrm{subset}}$ is applied to $\mathbf{P}_{\mathrm{PP}}^{\mathrm{subset}}$ to obtain the segmentation results $\mathbf{R}_{\mathrm{PP}}^{\mathrm{subset}}$.

The post-processing of the segmentation results is inspired by the scheme of Kang et al. [KTT+16]. They apply a modified bilateral filter to smooth the probability map based on the depth map. In particular their approach smoothens the probability map but preserves edges of the depth map in the probability map. Their approach is solely based on the depth map because the color image is not available. This limitation is not present in our approach because we exploit both the depth map and the color image. We use the color image instead of the depth map because it helps to preserve the discontinuities at skin-object borders, especially for objects not colored unlike skin. The modified bilateral filter is defined as

$$\tilde{\mathbf{P}}(\mathbf{x}) = \frac{1}{w} \sum_{\mathbf{x}_i \in \Omega} g_r\left(c_{\mathrm{dist}}^{\mathrm{CIELAB}}(\mathbf{x}_i - \mathbf{x})\right) g_s\left(\|\mathbf{x}_i - \mathbf{x}\|\right) \mathbf{P}(\mathbf{x_i}) \, , \tag{4.8}$$

where $\mathbf{P}(\mathbf{x})$ denotes the probability in $\mathbf{P}_{\mathrm{RoI}}^{\mathrm{subset}}$ at pixel position $\mathbf{x}$. The filtered probability value at $\mathbf{x}$ is denoted as $\tilde{\mathbf{P}}(\mathbf{x})$. We apply the proposal of Tomasi et al. [TM98] and use the Euclidean distance of the CIELAB color space for bilateral filtering of color images. Consequently, variable $c_{\mathrm{dist}}^{\mathrm{CIELAB}}$ describes the Euclidean distance of the CIELAB color space and is defined as

$$c_{\mathrm{dist}}^{\mathrm{CIELAB}}(\mathbf{x}_i - \mathbf{x}) = \left\| \begin{bmatrix} L_{\mathbf{x}_i}^* & a_{\mathbf{x}_i}^* & b_{\mathbf{x}_i}^* \end{bmatrix}^{\mathrm{T}} - \begin{bmatrix} L_{\mathbf{x}}^* & a_{\mathbf{x}}^* & b_{\mathbf{x}}^* \end{bmatrix}^{\mathrm{T}} \right\| \, , \tag{4.9}$$

where the vector elements $L_{\mathbf{x}}^*$, $a_{\mathbf{x}}^*$ and $b_{\mathbf{x}}^*$ denote the colors of the CIELAB color space at the pixel position $\mathbf{x}$. The factor w in (4.8) is defined as

$$w = \sum_{\mathbf{x}_i \in \Omega} g_r\left(c_{\mathrm{dist}}^{\mathrm{CIELAB}}(\mathbf{x}_i - \mathbf{x})\right) g_s\left(\|\mathbf{x}_i - \mathbf{x}\|\right) \, . \tag{4.10}$$

The functions $g_r(c)$ and $g_s(c)$ in (4.8) and (4.10) are the exponentials

$$g_r(c) = \exp\left(\frac{c^2}{2\sigma_r^2}\right) \qquad \text{and} \qquad g_s(c) = \exp\left(\frac{c^2}{2\sigma_s^2}\right) \, , \tag{4.11}$$

where c is a scalar value and σ_r and σ_s denote the smoothing constants. We decide to use a kernel size of 15×15 pixels. The standard deviations are chosen as $\sigma_r = 10$

and $\sigma_r = 5$. Our definition of the modified bilateral filter in (4.8) smoothens the probability map $\mathbf{P}_{\text{RoI}}^{\text{subset}}$ while preserving the edges of the color image.

Table 4.5 shows the F_1-scores at the optimal and chosen thresholds resulting from the probability maps $\mathbf{P}_{\text{RoI}}^{\text{subset}}$ and $\mathbf{P}_{\text{PP}}^{\text{subset}}$. The table shows that the performance after post-processing for chosen threshold and optimal threshold is equal; whereas, without post-processing the average F_1-score drops by 0.3 %. The segmentation performance after post-processing is 1.4 % worse than without.

Type	$b = \text{RoI}$	$b = \text{PP}$
$\overline{F}_{1,\text{opt}}^{\text{subset},b}$	94.9 %	93.2 %
$\overline{F}_{1,\text{chosen}}^{\text{subset},b}$	94.6 %	93.2 %
t_b^{subset}	42 votes	35 votes

Table 4.5: The table shows the optimal and chosen average F_1-score of the RF $f_{\text{RoI}}^{\text{subset}}$ before and after post-processing. The column $b = \text{RoI}$ contains the results before post-processing and the column $b = \text{PP}$ contains the results after filtering. Additionally, the skin detection thresholds $t_{\text{RoI}}^{\text{subset}}$ and $t_{\text{PP}}^{\text{subset}}$ are given.

In Fig. 4.8 the segmentation results before and after post-processing of various grasp situations are compared. The set of examples is the same as in Sec. 4.1 and 4.2. The chosen thresholds $t_{\text{RoI}}^{\text{subset}}$ and $t_{\text{PP}}^{\text{subset}}$ are given in Tab. 4.5. The first row shows the RoI of the color image and the second row displays the ground truth mask (green) of the examples. The black pixels in the ground truth mask correspond to pixels with invalid depth measurement. The white and green pixels identify non-skin and skin pixels. The third and fourth row show the segmentation results before (purple) and after (olive) filtering with the modified bilateral filter. The corresponding F_1-score is shown beneath each of the segmentation results. At first sight the filtered segmentation masks have smoother contours and at least from visual inspection do reflect the skin pixels better, although in example 1, 2 and 4 the F_1-score drops. In the following the observations of the examples are described:

Example 1: The application of post-processing smooths the contours of the segmentation results. Additionally, the dent between thumb and index finger in $\mathbf{R}_{\text{RoI}}^{\text{subset}}$ is “filled” in $\mathbf{R}_{\text{PP}}^{\text{subset}}$. The same holds for the left side of the wrist.

Example 2: The example shows that post-processing can worsen the result. The post-processing makes the misclassified areas of the skin colored box more “solid”. This effect is intuitive because post-processing smoothens the probability values in the area of the box. In this example the result is worsened by the post-processing step but the example shows a difficult situation because the box and skin do have a similar color.

Example 3: This example shows, that the smoothing characteristic of the filtering step “fills” small holes in the segmentation result, compare the left border of the index finger. Moreover, the thumb is more “solid” in $\mathbf{R}_{\mathrm{PP}}^{\mathrm{subset}}$ than in $\mathbf{R}_{\mathrm{RoI}}^{\mathrm{subset}}$.

Example 4: The segmentation result of the shown grasping position cannot be improved by post-processing. The result is similar as in the other two experiments. As mentioned in Sec. 4.2 the reason might be that the training set does not contain enough examples showing a similar grasping situation.

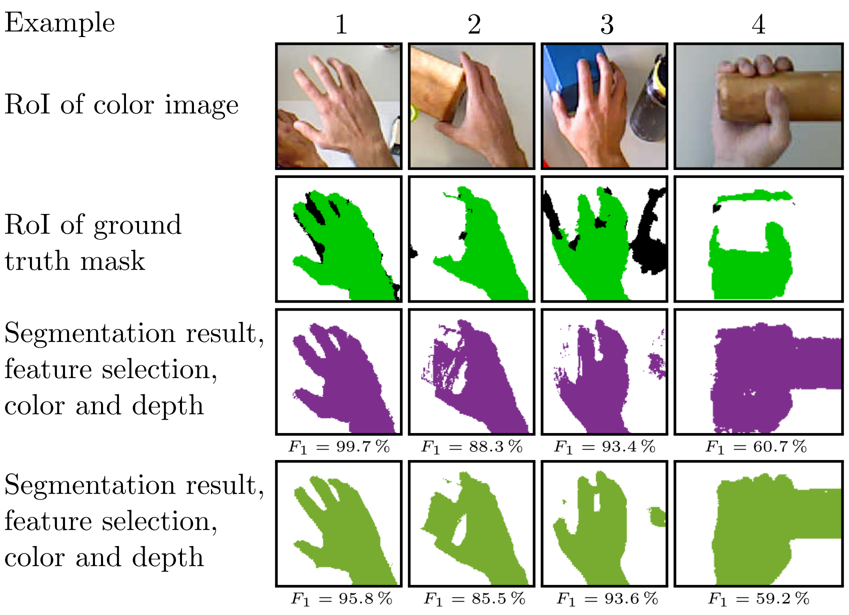


Figure 4.8: The figure shows the segmentation results $\mathbf{R}_{\mathrm{RoI}}^{\mathrm{subset}}$ and $\mathbf{R}_{\mathrm{PP}}^{\mathrm{subset}}$ of the RF $f_{\mathrm{RoI}}^{\mathrm{subset}}$ of different situations. Example 1 shows the segmentation result for no-object interaction, example 2 shows a grasp of the skin colored box (Fig. 3.3(f)), in example 3 a blue box is grasped while in example 4 the skin colored tin (Fig. 3.3(e)) is grasped. The first row shows the color image and the second row the ground truth mask (green). The black pixels in the ground truth mask indicate invalid mask pixels due to invalid depth measurements. The white and green pixels indicate non-skin and skin pixels. The purple pixels in $\mathbf{R}_{\mathrm{RoI}}^{\mathrm{subset}}$ and the olive pixels in $\mathbf{R}_{\mathrm{PP}}^{\mathrm{subset}}$ identify skin pixels. The corresponding F_1-score is shown beneath each segmentation result.

The Fig. 4.9, 4.10 and 4.11 show the same set of examples as in Sec. 4.2. In contrast to previous section, the figures show the probability maps before and after post-processing. The RoI of the color image is shown in the first row. In rows two, three and four the scaled color image, the probability map and the segmentation mask of the RF $f_{\mathrm{RoI}}^{\mathrm{subset}}$ are shown. The rows five, six and seven show the scaled color image, the probability map and the segmentation mask of the RF $f_{\mathrm{PP}}^{\mathrm{subset}}$. The scaled color images are obtained by multiplying the color image from row one with the probability map in row three or six. The probability maps in rows three and six are scaled to the value range $[0;1]$. The F_1-score corresponding to the segmentation result is written beneath it.

In the following we discuss some of the examples in Fig. 4.9 in more detail. The figure only shows examples of hand-object interaction with skin colored objects. These examples are difficult for the RF to segment because the subtle differences between object and skin at the areas of contact are difficult to identify. The example in the third column shows that post-processing helps to correctly classify misclassified areas as can be seen at the left side of the wrist. The benefits of post-processing with the modified bilateral filter can be seen in column nine. The "gaps" in $\mathbf{R}_{\text{RoI}}^{\text{subset}}$ very likely occur due to invalid depth measurements for these pixels. The segmentation result $\mathbf{R}_{\text{PP}}^{\text{subset}}$ does not contain the "gaps" anymore. The filtered probability maps shows how the "gaps" have been smoothed out.

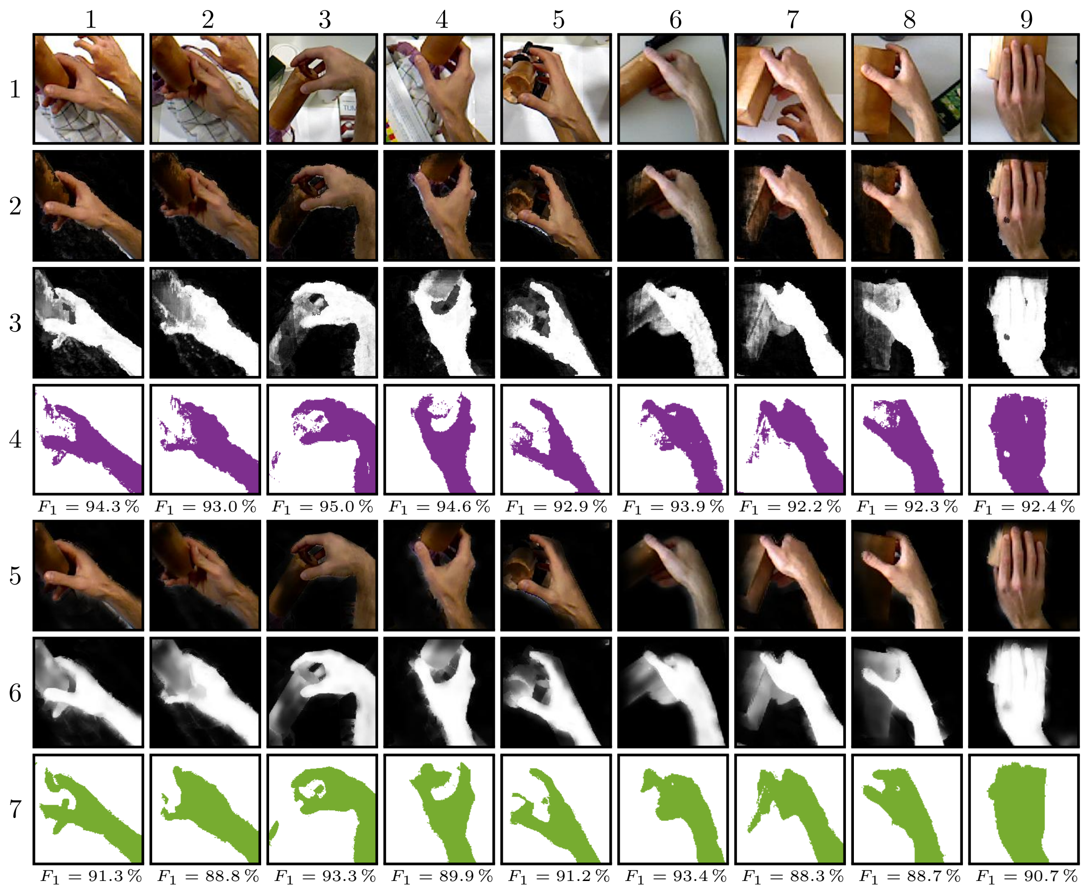

1 2 3 4 5 6 7 8 9
1
2
3
4
$F_1 = 94.3\,\%$ $F_1 = 93.0\,\%$ $F_1 = 95.0\,\%$ $F_1 = 94.6\,\%$ $F_1 = 92.9\,\%$ $F_1 = 93.9\,\%$ $F_1 = 92.2\,\%$ $F_1 = 92.3\,\%$ $F_1 = 92.4\,\%$
5
6
7
$F_1 = 91.3\,\%$ $F_1 = 88.8\,\%$ $F_1 = 93.3\,\%$ $F_1 = 89.9\,\%$ $F_1 = 91.2\,\%$ $F_1 = 93.4\,\%$ $F_1 = 88.3\,\%$ $F_1 = 88.7\,\%$ $F_1 = 90.7\,\%$

Figure 4.9: The figure shows the segmentation results of different grasping positions for two different skin colored objects. The first row shows the RoI of the color image. In rows two, three and four the scaled color image, the probability map and the segmentation mask of the RF $f_{\text{RoI}}^{\text{subset}}$ is shown. Rows five, six and seven show the scaled color image, the probability map and the segmentation mask after post-processing. The scaled color images are obtained by scaling the color image from row one with the probability map in row three or six. The probability maps in rows three and six are scaled to the value range $[0; 1]$. The F_1-score corresponding to the segmentation result is written beneath it.

Figure 4.10 shows examples for interaction with objects which do not have a color similar to skin; moreover, all of the examples are cuboids. The segmentation results before and after post-processing clearly show better results than those in Fig. 4.9. The example in column two shows that filtering the probability map helps to extract the tips of fingers better, this can be seen by comparing the segmentation result of the thumb before and after post-processing. The examples in column six and eight confirm that the filtering process with the modified bilateral filter helps to smooth out “gaps” due to invalid depth measurements.

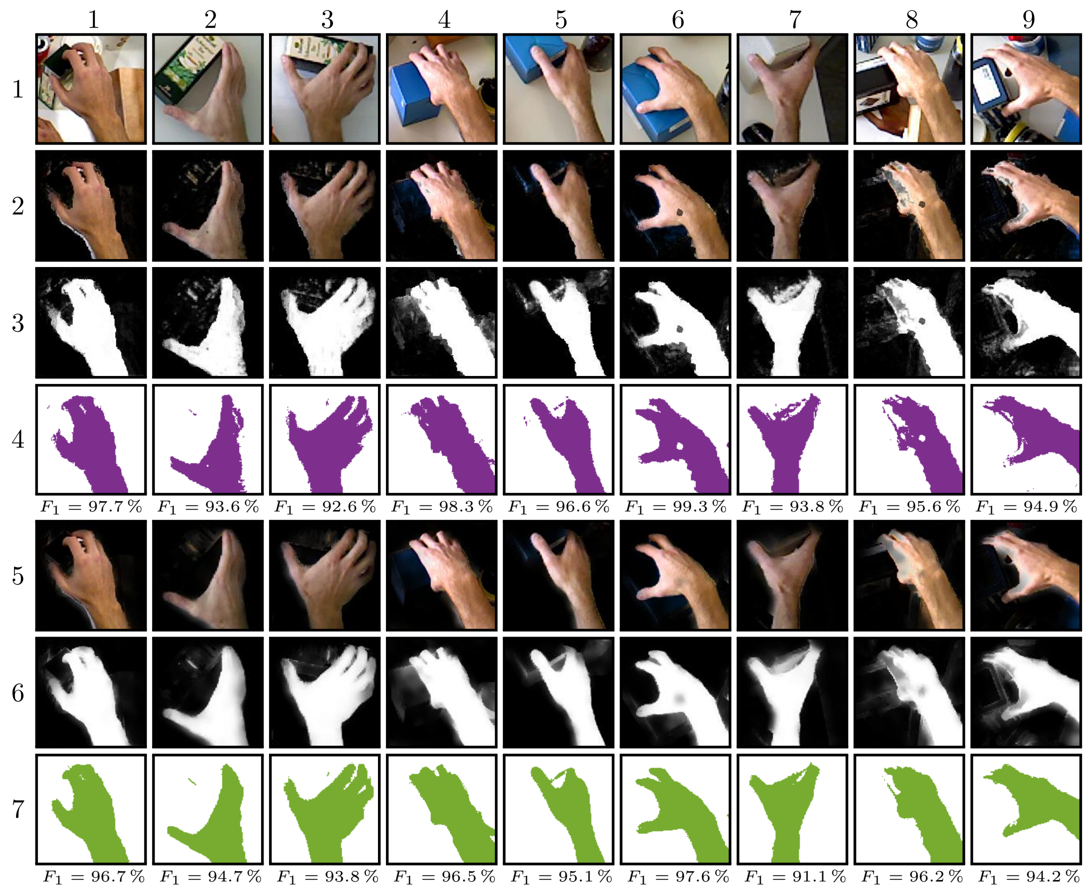

1 2 3 4 5 6 7 8 9
1
2
3
4
$F_1 = 97.7\,\%$ $F_1 = 93.6\,\%$ $F_1 = 92.6\,\%$ $F_1 = 98.3\,\%$ $F_1 = 96.6\,\%$ $F_1 = 99.3\,\%$ $F_1 = 93.8\,\%$ $F_1 = 95.6\,\%$ $F_1 = 94.9\,\%$
5
6
7
$F_1 = 96.7\,\%$ $F_1 = 94.7\,\%$ $F_1 = 93.8\,\%$ $F_1 = 96.5\,\%$ $F_1 = 95.1\,\%$ $F_1 = 97.6\,\%$ $F_1 = 91.1\,\%$ $F_1 = 96.2\,\%$ $F_1 = 94.2\,\%$

Figure 4.10: The figure shows the segmentation results of different grasping positions for five different boxes, the boxes differ in size and color. The structure of the figure is equal to Fig. 4.9.

The example in Fig. 4.11 shows both, box-shaped and cylindrical objects. The scene in the first column shows another example, where the post-processing “fills” gaps in the segmentation result and extracts the tip of fingers and thumb more precisely than without post-processing. The example in the last column shows that misclassifications due to invalid depth measurements are improved as well as the dent at the upper edge of the index finger is smoothed. Furthermore, the post-processing step extracts the thumb more clearly than without post-processing.

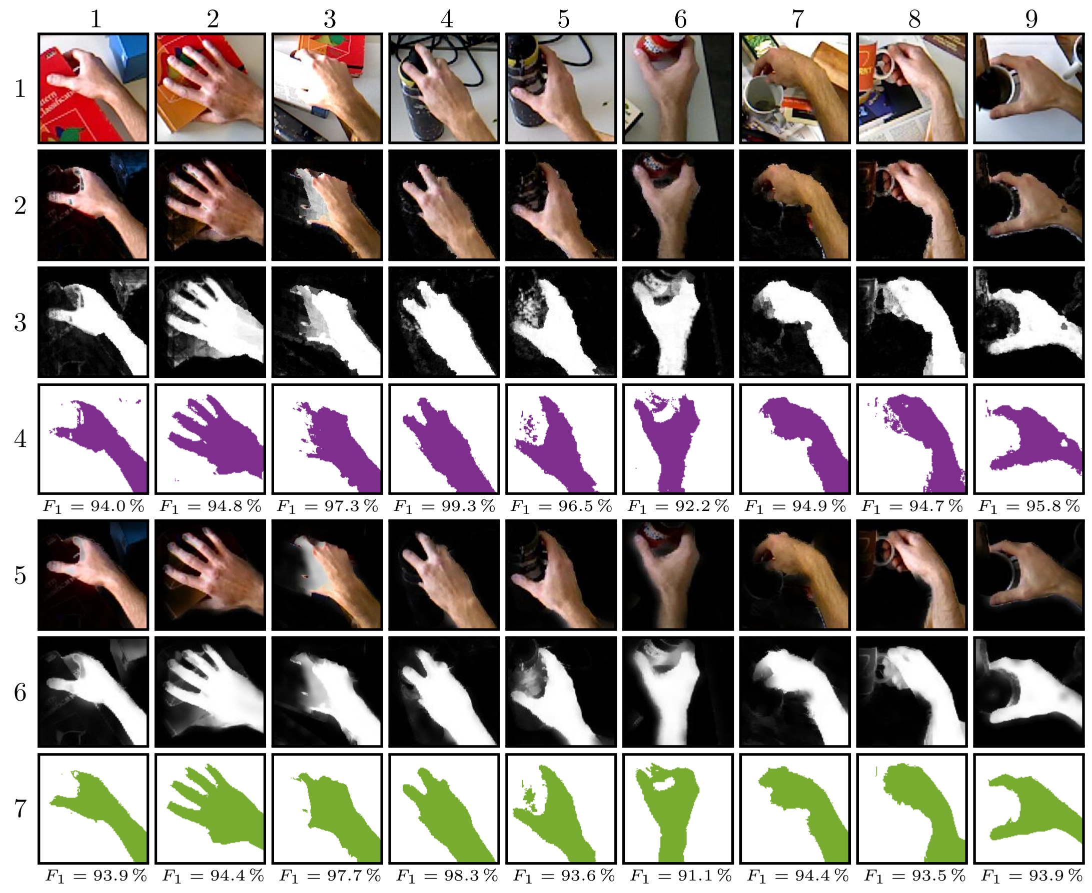

1 2 3 4 5 6 7 8 9
1
2
3
4
$F_1 = 94.0\,\%$ $F_1 = 94.8\,\%$ $F_1 = 97.3\,\%$ $F_1 = 99.3\,\%$ $F_1 = 96.5\,\%$ $F_1 = 92.2\,\%$ $F_1 = 94.9\,\%$ $F_1 = 94.7\,\%$ $F_1 = 95.8\,\%$
5
6
7
$F_1 = 93.9\,\%$ $F_1 = 94.4\,\%$ $F_1 = 97.7\,\%$ $F_1 = 98.3\,\%$ $F_1 = 93.6\,\%$ $F_1 = 91.1\,\%$ $F_1 = 94.4\,\%$ $F_1 = 93.5\,\%$ $F_1 = 93.9\,\%$

Figure 4.11: The figure shows the segmentation results of different grasping positions for various objects. The scenes include three grasps of books, three grasps of cylindrical shaped objects and three grasps of mugs. The structure of the figure is equal to Fig. 4.9 and 4.10.

The overall impression of the Fig. 4.9, 4.10 and 4.11 is that the post-processing step makes the hand more perceptible for humans in the probability maps. Additionally, the contours of the hand are smoother and look more like the expectation of humans. However, the result also show that depending on the situations, areas which are misclassified as skin are reduced or increased. Nevertheless, this result is not unexpected because areas with a high probability of being skin are not removed by smoothing the probability map with the modified bilateral filter. Furthermore, comparing all three figures shows that especially cases of hand-object interaction with skin colored objects limit the performance of the post-processing step.

4.4.1 Discussion of Experiment 4

The numerical comparison of the segmentation quality before and after applying the post-processing step in Tab. 4.5 and the F_1-scores given in Fig. 4.9, 4.10 and 4.11

indicate that post-processing worsens the result. However, the visual impression of the result is improved after post-processing because the probability map and the resulting segmentation results are smoother and look more like humans would expect them. Especially areas where the probability is very noisy are smoothed out. A possible reason for a reduced average F_1-score after post-processing is, that the ground truth masks themselves do not have a smooth border. Thereby, the raw classification result without post-processing might be closer to the ground truth mask. Furthermore, the ground truth masks are created by means of the depth map and noise around the discontinuities in the depth map might cause that the segmentation mask is not perfectly aligned with the RGB image. Thereby, causing the drop in F_1-score.

Chapter 5

Discussion

This chapter discusses the results of the experiments as well as the quality of the captured data sets. The data sets and the estimation of the ground truth is discussed in Sec. 5.1, and the results of the experiments are analyzed in Sec. 5.2.

5.1 Data Sets and Ground Truth

First, let us focus on the ground truth estimation. The experiments show that the ground truth contains too many invalid pixels which is due to the characteristics of the estimation scheme. A ground truth pixel being invalid is caused by invalid depth measurements either in the depth map of the reference frame or the training frame. Invalid depth measurements can occur in regions of discontinuities in the depth map, for instance at the line of intersection between foreground and background. Thereby, pixels at the area of contact between hand and object are marked as invalid because our ground truth estimation scheme cannot detect if such pixels show skin or not. In particular, invalid mask pixels cannot be assigned to either non-skin or skin class which implies that we cannot use these pixels for our supervised training scheme because the class information is missing. As a consequence, the RF never learned to find the correct class for pixels with missing depth measurement, some of the examples in the experiment section show small "gaps" which are caused by missing depth information.

Second, the ground truth estimation scheme detects skin if there is a change in the depth map between reference frame and training frame of more than 1 cm. The application of the threshold on the difference is necessary due to noise in the depth measurement. As a result, the fingertips may not be marked as skin when they touch an object. Furthermore, the thresholding makes the edges of the hand or skin rather jagged than smooth.

Third, 50 % of the images in the training data set and verification data set show no hand-object interaction. These examples are included in the data sets because we do not want to exclude situations without object interaction. However, detecting the skin of the hand alone is simple if depth features are available, because it boils down to a simple background foreground detection. On the other hand, situations with hand-object interaction are more challenging to segment because the change in depth between skin and object is small. We think that increasing the number of examples showing more challenging situations can improve the segmentation quality.

Fourth, the ground truth estimation scheme puts another restriction on the desktop scene, i.e. the objects may not move during the collection process because otherwise the objects are also marked as skin pixels. This restriction makes the set up of the desktop scene extremely complicated for realistic grasping situations. Basically, the object has to be fixed in the air so that it can be grasped in each possible and realistic way. Although the data sets contain such examples, we think the number should be increased to improve the quality of the segmentation results. The data sets contain a lot of situations which are difficult for color features but not many situations which are difficult for depth features. This could be a reason why the RF using only depth features performs similarly good as the RF using both depth and color features.

Finally, some of the examples in the data sets are overexposed. Overexposure of the RGB image can cause that parts of the skin are captured as pure white. Especially, the color features may mislead the RF to detect these areas as skin. This issue can be observed in the example in the third column of Fig. 4.6. Section 4.2 shows that depth features are more important for the segmentation; nevertheless, the RF relying on depth and color features does not robustly detect the skin pixels.

5.2 Discussion of Experiments

The experiment in Sec. 4.1 shows that training of the RFs on the entire image does not achieve a good performance. Considering we are mainly interested in segmenting the hand from the object, the process of segmenting the entire image is overly time consuming. Furthermore, the classification result shows too many regions which are misclassified as skin and thus, it is difficult to find the areas in the image we are interested in. Regardless of the aforementioned issues the RFs do not learn to distinguish hand and object because the training samples are drawn from the RGB-D images according to a uniform distribution. Thereby, the training samples are distributed over the entire image although it is favorable to restrict the training samples to be in the RoI.

The results from the experiment in Sec. 4.1 indicate that training on the RoI helps to overcome the identified problems. For experiments in Sec. 4.2, 4.3 and 4.4 we assume that we know where the RoI is in each image. This can be achieved in a real set-up for instance with the hand position obtained by the Kinect skeletal tracker or with a bracelet in an intense color. Extracting a 20 cm $\times$ 20 cm image patch reduces the classification time per image by a factor of around 10; additionally, the feature calculation time also is reduced by the same factor.

Especially the experiment in Sec. 4.3 shows that applying both feature types, depth and color, performs better in difficult situations. With difficult situations we refer to circumstances where either the RF using depth features only or the RF using color features only perform poorly. It is a reasonable assumption that such difficult situations occur in realistic scenarios. As mentioned in Sec. 5.1, the reason for the similar performance of the RF using depth features only and the RF using depth and color features could originate from the fact that the data sets do not contain a lot of situations which are difficult for the RF using only depth features. Independently of the absence of a large number of such difficult situations in the training data set, the RF $f_{\mathrm{RoI}}^{\mathrm{both}}$ performs very well and better than the RFs $f_{\mathrm{RoI}}^{\mathrm{depth}}$ and $f_{\mathrm{RoI}}^{\mathrm{color}}$ in case of difficult situations.

In the experiments we are also interested in the performance of the RF if only a subset of features is applied. The subset is found by taking the 60 most important features of the feature set and train another RF with those features. Considering the verification data set using only the subset causes almost no loss of segmentation quality. However, investigating the performance of the feature subset on the data sets EC1, EC2 and EC12 shows that it stays behind the feature vector containing all features. Nevertheless, the RF using the feature subset performs better than the RFs using either depth or color features.

The applied post-processing on the segmentation deteriorates the quality with respect to the ground truth. However, as pointed out in Sec. 5.1 the ground truth is not optimal. This could explain why post-processing decreases the average F_1-score. The appearance of the filtered probability maps is closer to the expectation of humans. Additionally, the noisy nature of the probability maps is reduced. The most obvious improvement of the probability maps is the smoothness of the hand shape which makes the result look more realistic and exact. According to the perceptual impression of the results, finger tips are more visible than without post-processing. Moreover, post-processing fills "gaps" due to invalid depth measurements inside the hand. The aforementioned observations are most visible if the skin's color clearly differs from the rest of the objects in the scene which is plausible because the modified bilateral filter smoothes the probability maps while preserving the discontinuities of the color image.

Chapter 6

Conclusion

This chapter concludes the Master's Thesis by briefly summarizing the work performed and providing ideas for future work.

This thesis deals with skin or hand segmentation classification on pixel-level. The approach chosen in this thesis uses a RF classifier and employs both depth and color information in the features. At the beginning a literature study about hand segmentation methods has been conducted. Next, a software framework for ground truth estimation was implemented. This framework was used to collect two data sets, a training data set containing 3420 RGB-D images and a verification data set with 922 RGB-D images. The data collection was done because to the best of our knowledge there is no public labeled data set available containing RGB-D images from an ego-centric perspective. Additionally, a scheme to extract the RoI in the labeled data set was created because the hand position is not available in the data sets. Next, a software framework to train RF classifiers and to classify RGB-D images with the RF was implemented. Finally, the framework has been used to train several RFs with different feature vectors. The training was done on the entire image as well as on the RoI. The different feature vectors contained either depth features only, color features only, the entire feature set or the 60 best features of the entire feature set. The trained RFs are compared and evaluated. Furthermore, the influence of post-processing the classification outputs with a modified bilateral filter has been evaluated.

The results of the experiments in Chap. 4 and the discussion in Chap. 5 indicate that the collected data sets suffer from certain problems, such as too many examples showing only the hand, not enough realistic grasps and the ground truth contains too many invalid pixels. Especially, invalid pixels put a limitation on the training process because the classifier does not learn how to decide for skin or non-skin if there is an invalid depth measurement. The problems of the data set have various reasons. First, we are interested in segmenting the hand in hand-object scenarios but we do not want to exclude situations without hand-object interaction. Second, the scheme for ground truth estimation is kept extremely simple and thus, may cause more invalid pixels in the masks. Third, collecting data containing realistic grasps is challenging and time consuming because the object may not move during ground truth estimation. This implies that the object would have to be fixed in the air.

The experiments, especially in Sec. 4.3, show that the feature vector containing both depth and color features performs better than the vectors containing either depth or color features. Furthermore, the experiments show that the RF using our proposed feature vector achieves good results for difficult situations where the object's shape and color are similar to the skin. Although the training data set lacks examples which are difficult for the RF using only depth features, the RF using depth and color features performs very well on such difficult situations. The results are promising because the segmentation is done pixel-wise, meaning the classification result of each pixel is independent from the classification result of surrounding pixels and no interpretation of the hand shape is done.

The post-processing step filters the probability map with a modified bilateral filter. This filter smoothens the probability map but preserves the discontinuities of the color image in the probability map. The application of such a filter especially is useful if hand and object are of distinct color. The filtered output looks more realistic and less noisy compared to the raw output. Additionally, the tips of the finger are carved out better and the contours of the hand are smoother. These observation make the filtered probability map look more natural and realistic.

6.1 Future Work

In future work the data sets should be improved. In the improved data sets more examples of realistic grasps are needed and less examples of the hand held above the desk are required. Furthermore, our proposed feature vector on average gives better results than feature vectors containing either color or depth features alone, especially if difficult situations are considered. As a result the improvement of our approach over preceding ones using either depth or color is dependent on the situation. In order to validate in which situations our approach works better a categorized data set containing different situations is needed. Furthermore, the ground truth estimation method should be improved because the resulting masks contain too many invalid pixels. An improvement could be to combine several RGB-D images per frame and average the depth while excluding invalid depth measurements.

Modifying the feature vector could improve the classification as well. First, the experiment in Sec. 4.2 shows that the features from the Gabor filter bank are the least important features in the vector with respect to our training data set. Thereby, the features from the Gabor filter bank could either be removed from the vector or should be improved. The features could be improved by taking the absolute value of real part and imaginary part of the filter output. All absolute real parts of each scale are summed up, then the features are extracted from result. The same approach is applied to the imaginary parts. Second, the color image can be divided into superpixels and for each superpixel the mean and standard deviation of the color can be added to the feature vector. Another similar idea is to extract a patch of the color image around the pixel and compute the mean and the standard deviation of that exact patch. The size of the patch can either be dependent or independent on the depth of the pixel. Third, by increasing the number of depth features a more distinctive set of depth features can be found since the distribution of the depth features might not be optimal.

In order to make the segmentation algorithm applicable in real time the feature calculation has to be parallelized as well as the classification process. The parallelization can be done by executing the feature calculation as well as the classification process on the GPU.

Appendix A

Tables for Color Spaces

Point	Color	R	G	B	H	S	V
S	Black	0.00	0.00	0.00	–	0.00	0.00
R	Red	1.00	0.00	0.00	0/6	1.00	1.00
Y	Yellow	1.00	1.00	0.00	1/6	1.00	1.00
G	Green	0.00	1.00	0.00	2/6	1.00	1.00
C	Cyan	0.00	1.00	1.00	3/6	1.00	1.00
B	Blue	0.00	0.00	1.00	4/6	1.00	1.00
M	Magenta	1.00	0.00	1.00	5/6	1.00	1.00
W	White	1.00	1.00	1.00	–	0.00	1.00
K	50 % Gray	0.50	0.50	0.50			
$\mathbf{R}_{25}$	25 % Red	0.25	0.00	0.00	0.00	1.00	0.25
$\mathbf{R}_{50}$	50 % Red	0.50	0.00	0.00	0.00	1.00	0.50
$\mathbf{R}_{75}$	75 % Red	0.75	0.00	0.00	0.00	1.00	0.75
P	Pink	1.00	0.50	0.50	0.00	0.50	1.00

Table A.1: Values of the color points in Fig. 2.1 and Fig. 2.2 [BB09, Chap. 12, p. 310 & 326].

Point	Color	R	G	B	X	Y	Z	x	y
S	Black	0.00	0.00	0.00	0.0000	0.0000	0.0000	0.3127	0.3290
R	Red	1.00	0.00	0.00	0.4125	0.2127	0.0193	0.6400	0.3300
Y	Yellow	1.00	1.00	0.00	0.7700	0.9278	0.1385	0.4193	0.5052
G	Green	0.00	1.00	0.00	0.3576	0.7152	0.1192	0.3000	0.6000
C	Cyan	0.00	1.00	1.00	0.5380	0.7873	1.0694	0.2247	0.3288
B	Blue	0.00	0.00	1.00	0.1804	0.0722	0.9502	0.1500	0.0600
M	Magenta	1.00	0.00	1.00	0.5929	0.2848	0.9696	0.3209	0.1542
W	White	1.00	1.00	1.00	0.9505	1.0000	1.0888	0.3127	0.3290

Table A.2: Color values in the RGB color space, the CIEXYZ color space and the chromaticity diagram. The table is taken from [BB09, Chap. 14, p. 365].

Appendix B

Gabor Filter Complex Conjugate Property

The Gabor filter is 2π-periodic in the orientation parameter θ. Thus, for a general input signal to the Gabor filter bank it is sufficient to choose the interval of θ as $[0, 2\pi[$. Considering a real valued input signal the interval of θ can be set to $[0, \pi[$. This restriction is valid because for real valued input signals the responses for $\theta \in [\pi, 2\pi[$ are the complex conjugates of the responses in the interval $\theta = [0, \pi[$. Thus, designing Gabor filters with orientations from the set $[0, \pi[$ is sufficient for real signals [KKK06]. The Gabor filter function with orientation angle θ is defined as

$$\begin{aligned} h_\theta[x,y] = {} & \exp\left(-\frac{1}{2\sigma^2}\left((x\cos\theta + y\sin\theta)^2 + \gamma^2\left(-x\sin\theta + y\cos\theta\right)^2\right)\right)\cdot\ldots \\ & \exp\left(\mathrm{j}\frac{2\pi}{\lambda}\left(x\cos\theta + y\sin\theta\right)\right) . \end{aligned}$$

Let us define $\theta \in [0, \pi[$. Remember the periodicity property of the trigonometric functions $\cos(\theta + \pi) = -\cos\theta$ and $\sin(\theta + \pi) = -\sin\theta$. The Gabor filter function for orientations $\theta' = \theta + \pi$ can be written as

$$\begin{aligned} h_{\theta+\pi}[x,y] = {} & \exp\left(-\frac{1}{2\sigma^2}\left((x\cos\theta + y\sin\theta)^2 + \gamma^2\left(-x\sin\theta + y\cos\theta\right)^2\right)\right)\cdot\ldots \\ & \exp\left(-\mathrm{j}\frac{2\pi}{\lambda}\left(x\cos\theta + y\sin\theta\right)\right) \\ = {} & h^*_\theta[x,y] , \end{aligned}$$

where * denotes the complex conjugate. The filter output $y_\theta[x,y]$ can be described as

$$y_\theta[x,y] = \sum_{k_1\in\mathcal{K}}\sum_{k_2\in\mathcal{K}} I[x-k_1, y-k_2]h_\theta[k_1,k_2] ,$$

where $I[x,y]$ is the input signal at position $[x,y]$ and $\mathcal{K} = [-L, -L+1, \ldots, L+1, L]$ is the set of indices of the filter kernel along the rows and columns. The parameter L describes the last index of the filter kernel. The filter output $y_{\theta+\pi}[x,y]$ is given as

$$y_{\theta+\pi}[x,y] = \sum_{k_1 \in \mathcal{K}} \sum_{k_2 \in \mathcal{K}} I[x-k_1, y-k_2] h_{\theta+\pi}[k_1, k_2] \, ,$$

where $h_{\theta+\pi}[k_1,k_2]$ can be replaced by $h^*_\theta[x,y]$. The replacement leads to

$$y_{\theta+\pi}[x,y] = \sum_{k_1 \in \mathcal{K}} \sum_{k_2 \in \mathcal{K}} I[x-k_1, y-k_2] h^*_\theta[x,y] = y^*_\theta[x,y] \, ,$$

which shows that the filter response of a Gabor filter rotated by θ is the complex conjugate of the not rotated Gabor filter.

List of Figures

List of Tables

List of Abbreviations

BRIEF	Binary Robust Independent Elementary Features 35
CIE	International Commission on Illumination 12
EC1	Extreme Case Data Set 1 67
EC12	Extreme Case Data Set 1 & 2 67
EC2	Extreme Case Data Set 2 67
GIMP	GNU Image Manipulation Program 33
GPU	Graphics Processing Unit 7
HOG	Histogram of Oriented Gradients 35
HSV	Hue, Saturation and Value 10
NN	Neural Network 7
OOB	Out-of-Bag 26
ORB	Oriented FAST and Rotated BRIEF 35
RADAR	Radio Detection And Ranging 18
RF	Random Forest 6
RGB	Red, Green and Blue 7
RGB-D	Red, Green, Blue and Depth 6
RNO	Random Node Optimization 26
RoI	Region of Interest 6

ROS	Robot Operating System 34
SIFT	Scale-Invariant Feature Transform 8
SL	Structured Light 9
ToF	Time-of-Flight 18
VR	Virtual Reality 5

List of Symbols

$\mathcal{F}_b$	Set of f_b^a for experiment b. 53
$\mathcal{P}_b$	Set of $\mathbf{P}_b^a$ for experiment b. 54
$\mathcal{R}_b$	Set of $\mathbf{R}$ for experiment b. 54
$\mathcal{T}_b$	Set of t_b^a for experiment b. 54
$\mathbf{P}_b^a$	Matrix of the classification output, called probability map, for feature vector type a and experiment b. 54
$\mathbf{R}_b^a$	Matrix of the segmentation result obtained with type a feature vector of experiment b. 54
$\mathbf{v}$	Feature vector. 22
$\mathbf{w}^{\text{imp}}$	Vector containing the variable importance. 27
$\overline{F}_{1,\text{chosen}}^{a,b}$, $\overline{F}_{1,\text{opt}}^{a,b}$	Mean F_1-score for chosen and optimal threshold with feature type a and experiment b. 55
a	Type of the feature vector $\big(a \in [\text{depth}, \text{color}, \text{both}, \text{subset}]\big)$. 53
b	Type of the experiment $\big(b \in [\text{EI}, \text{RoI}, \text{PP}]\big)$. 53
f_b^a	RF with feature vector type a of experiment b. 53
t_b^a	Threshold t_{skin} for feature vector type a of experiment b. 54
t_{skin}	General skin detection threshold. 45

Bibliography

[AG94] Yali Amit and Donald Geman. Randomized inquiries about shape: An application to handwritten digit recognition. Technical report, DTIC Document, 1994.

[AG97] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588, 1997.

[BB09] Wilhelm Burger and Mark James Burge. *Digitale Bildverarbeitung: Eine algorithmische Einführung mit Java*. Springer-Verlag, 2009.

[BF07] Francesco Bianconi and Antonio Fernández. Evaluation of the effects of gabor filter parameters on texture classification. *Pattern Recognition*, 40(12):3325–3335, 2007.

[BFSO84] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, Apr 2002. doi:10.1109/34.993558.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[CS13] Antonio Criminisi and Jamie Shotton. *Decision forests for computer vision and medical image analysis*. Springer Science & Business Media, 2013.

[Dau85] John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169, 1985.

[Gab46] Dennis Gabor. Theory of communication. part 1: The analysis of information. *Electrical Engineers-Part III: Radio and Communication Engineering, Journal of the Institution of*, 93(26):429–441, 1946.

[Gra78] Goesta H Granlund. In search of a general picture processing operator. *Computer Graphics and Image Processing*, 8(2):155–173, 1978.

[HI16] Eberhard Hasche and Patrick Ingwer. *Game of Colors: Moderne Bewegtbildproduktion: Theorie und Praxis für Film, Video und Fernsehen*. Springer-Verlag, 2016.

[KKK04] Ville Kyrki, Joni-Kristian Kamarainen, and Heikki Kälviäinen. Simple gabor feature space for invariant object recognition. *Pattern recognition letters*, 25(3):311–318, 2004.

[KKK06] J-K Kamarainen, Ville Kyrki, and Heikki Kalviainen. Invariance properties of gabor filter-based features-overview and applications. *IEEE transactions on Image Processing*, 15(5):1088–1099, 2006.

[KMB07] Praveen Kakumanu, Sokratis Makrogiannis, and Nikolaos Bourbakis. A survey of skin-color modeling and detection methods. *Pattern recognition*, 40(3):1106–1122, 2007.

[KTT+16] Byeongkeun Kang, Kar-Han Tan, Hung-Shuo Tai, Daniel Tretter, and Truong Q Nguyen. Hand segmentation for hand-object interaction from depth map. *arXiv preprint arXiv:1603.02345*, 2016.

[LK13] Cheng Li and Kris M Kitani. Pixel-level hand detection in ego-centric videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3570–3577, 2013.

[Mar80] S Marĉelja. Mathematical description of the responses of simple cortical cells. *JOSA*, 70(11):1297–1300, 1980.

[OWL15] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Training a feedback loop for hand pose estimation. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[Poy97] Charles Poynton. Frequently asked questions about color. *Retrieved June*, 19:2004, 1997.

[Sha08] Toby Sharp. Implementing decision trees and forests on a gpu. In *European conference on computer vision*, pages 595–608. Springer, 2008.

[SKR+15] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, Daniel Freedman, Pushmeet Kohli, Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3633–3642, New York, NY, USA, 2015. ACM. URL: `http://doi.acm.org/10.1145/2702123.2702179`, `doi:10.1145/2702123.2702179`.

[SSK+13] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[TM98] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.

[VSA03] Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva. A survey on pixel-based skin color detection techniques. In *Proc. Graphicon*, volume 3, pages 85–92. Moscow, Russia, 2003.

[WA15] Shaohua Wan and JK Aggarwal. Robust object recognition in rgb-d egocentric videos based on sparse affine hull kernel. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 97–104, 2015.

[ZMDM$^+$16] Pietro Zanuttigh, Giulio Marin, Carlo Dal Mutto, Fabio Dominio, Ludovico Minto, and Guido Maria Cortelazzo. *Time-of-Flight and Structured Light Depth Cameras.* Springer, 2016.