FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Autonomous in hand object learning from rgb-d input with a mobile robot

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Computer Science**

by

**Dominik Streicher, BSc**
Registration Number 1026446

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
Assistance: Dipl.-Ing. Dr.techn. Michael Zillich

Vienna, 13th September, 2017

Dominik Streicher | Markus Vincze

# Erklärung zur Verfassung der Arbeit

Dominik Streicher, BSc
Heide 2.a Strasse 3, 3331 Kematen an der Ybbs

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 13. September 2017

Dominik Streicher

# Danksagung

Besonders danken möchte ich dem Betreuer dieser Arbeit, Herrn Projektass. Dipl.-Ing. Dr.techn. Michael Zillich. Er unterstützte meine Arbeit durch wertvolle Anregungen und interessante Diskussionen und trug damit wesentlich zur Qualität dieser Arbeit bei. Ebenfalls möchte ich dem ACIN Institut danken, die mich herzlichst in Ihrer Gruppe aufgenommen haben und mir stets bei allen Anliegen zur Seite standen.
Weiters möchte ich noch meinem verstorbenen Vater danken, der mir stets ein Vorbild war und mich in meinem Leben stetig unterstützte und mir das Studium so erst ermöglichte.

# Kurzfassung

Diese Arbeit präsentiert einen methodischen Ansatz, um ein Objekt in einer menschlichen Hand zu modellieren. Ein großer Vorteil unserer Methode ist, dass die Modellierung in Echtzeit und ohne aufwändige manuelle Kalibrierung funktioniert. Ebenfalls kann das Objekt direkt in der Hand modelliert werden, was einen großen Vorteil in der Human Robot Interaction (HRI) bietet. Hier können Objekte direkt in der Hand des Benutzers erkannt werden, ohne dass dieser die Objekte irgendwo speziell platziert. Um das zu erreichen präsentieren wir eine Lösung zur effizienten Segmentierung der Hand und des Hintergrundes mittels einer RGB-D Kamera. Hierbei werden wir anhand des Gesichts die Hautfarbe extrahieren und anschließend damit die Hand im Bild entfernen. Danach wird das freistehende Objekt segmentiert. Anschließend verwenden wir die Vision for Robotics (V4R)-Bibliothek um die einzelnen Bilder zu einem vollständigen 3D-Modell hinzuzufügen. Am Ende unserer Arbeit werden wir anhand einiger Objekte die Effizienz demonstrieren und die Grenzen dieser Methode aufzeigen.

# Abstract

This thesis presents a full methodological approach to model objects, which are held in human hands. A major advantage of our method is, that it works in real time and does not need a complex manuel calibration. Another advantage is, that the object can be modeled inside of the users hand, which is a big advantage in HRI. That means objects can be learned without puting them in some designated place. To acheive this, we present a solution for efficient segmention of the user's hand and the background, using an RGB-D camera. For this we will extract the skin color of the user's face to detect the user's hand and remove it in the image. Afterwards the object will be segmented. To combine the different views, we use V4R-Librabry to map the single images to a 3D-model. To show the efficiency of our method, we present at the end some models, which we created with our method and discuss the limits.

# Contents

CHAPTER 1

# Introduction

Modeling of objects has a significant importance in the area of computer vision. Having 3D models of objects can be useful for higher-level tasks like recognition, tracking and manipulation. For robots, which are operating in unstructed real-world enviroments it is important that they autonomously learn and adapt their environment. One important element of such a mobile robot's environment are the objects present there. Fully three dimensional scanners, which are now widely available allows us easily to create 3D point clouds from the environment. A major challenge is to apply segmentation, recognition and pose estimation in occluded environments, where scenes are captured by low-resolution RGBD-sensors like the Microsoft Kinect. [FAB$^+$17] [NL13] [LN10] [AAD$^+$11] [ML].
One project challenging all these tasks is the SQUIRREL project. SQUIRREL addresses these issues by actively controlling clutter and incrementally learning to extend the robot's capabilities while doing so. The robot tackles clutter one bit at a time and also extends its knowledge continuously as new bits of information become available. SQUIRREL is inspired by a user driven scenario, that exhibits all the rich complexity required to convincingly drive research, but allows tractable solutions with high potential for exploitation. We propose a toy cleaning scenario, where a robot learns to collect toys scattered in loose clumps or tangled heaps on the floor in a child's room, and to stow them in designated target locations. [Squ].
This work is part of the SQUIRREL project and was initially motivated by the purpose of object regocnition in a children's room. Before we can efficiently recognize objects, we need the corresponding models for the objects, which should be recognized. To get a full 3D model of an object we need different views of the same object, which can be combined to one model. One approach is to drive around the object and get so different views. But often its not possible to drive around, for example not enough space or the robot is not mobile. The goal of this project is to create a 3D model of an object by rotating it with a human hand in front of a RGB-D camera. There should be also no manual calibration necessary to obtain the model. Another requirement is, that it should

work in real-time, which means, that the object can be observed in less than one second. We do not found any ready to use solutions for our project. So we decided to divide the problem in three parts. The first part is responsible for hand detection. The second part contains the segmentation of the object in the human hand. The third part contains the modelling of the object with V4R [V4R]. In figure 1.1 a simplified visual representation of our method is shown.

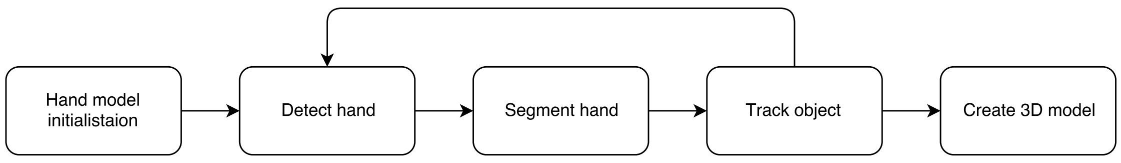


Figure 1.1: Visual representation of our workflow.

At the end of this thesis, we will present our experimental results and will discuss the limitations of our method.

CHAPTER 2

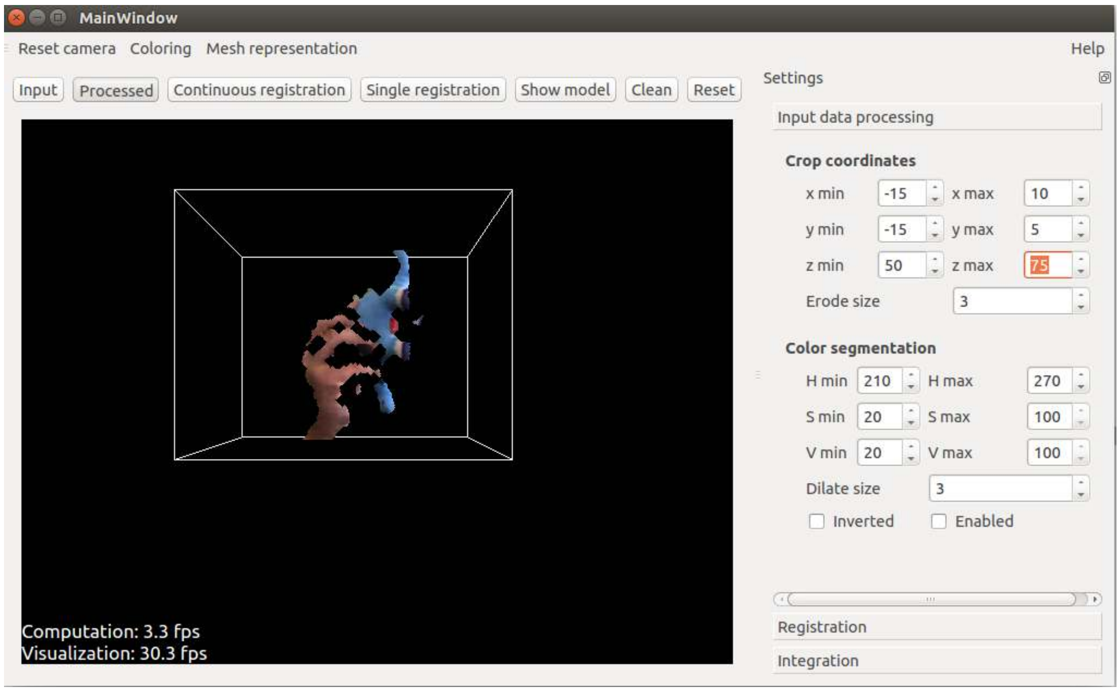
# Related work

## 2.1 Object detection

Especially for our specific problem, detecting objects in human hands, there is not much related work available.

There exists already an implementation of an in hand scanner from Point Cloud Library (PCL) [PCL]. The purpose of the in-hand scanner application is to obtain a 3D model from a small object using a RGB-D camera. The user is turning the object around in front of the sensor, while the geometry is reconstructed gradually. We also tried this implementation, which is shown in figure 2.1. The big disadvantage of this application is, that the user has to adjust manually a bounding box around the object. Also the color for segmenting the hand, has to be manually adjusted. Afterwards the user has to hold the object very carefully inside of the bounding box, which is not very easy, if the user has to rotate the object too. So this program is not usable for our project, because we need a user-friendly system usable by naive users, without the need top keep looking at the screen.

Another approach [KHRF11] is building 3D models of unknown objects based on a depth camera observing the robot's hand while moving an object. The approach integrates both shape and appearance information into an articulated Iterative Closest Point (ICP) approach to track the robot's manipulator and the object. Objects are modeled by sets of surfels, which are small patches providing occlusion and appearance information. Experiments show that their approach provides very good 3D models even when the object is highly symmetric and lacks visual features and the manipulator motion is noisy. Autonomous object modeling represents a step toward improved semantic understanding, which will eventually enable robots to reason about their environments in terms of objects and their relations rather than through raw sensor data. In their approach they use a kalman filter to estimate the pose of the object by the use of joint encoders at the



Crop coordinates

x min -15 x max 10

y min -15 y max 5

z min 50 z max 75

Erode size 3

Color segmentation

H min 210 H max 270

S min 20 S max 100

V min 20 V max 100

Dilate size 3

☐ Inverted ☐ Enabled

Computation: 3.3 fps
Visualization: 30.3 fps

Figure 2.1: PCL In-hand-scanner.

gripper. In our project the object is in a human hand, therefore we have no further information beside the camera image. That means that this approach is not suitable in our environment.

One approach [KLK14] uses an incremental learning method, which allows tracking objects without prior knowledge. In this paper, they propose a novel model-free approach for tracking multiple objects from RGB-D point set data. This study aims to achieve the robust tracking of arbitrary objects against dynamic interaction cases in real-time. In order to represent an object without prior knowledge, the probability density of each object is represented by Gaussian mixture models with a tempo-spatial topological graph. A flexible object model is incrementally updated in the pro-posed tracking framework, where each RGB-D point is identified to be involved in each object at each time step. Furthermore, the proposed method allows the creation of robust temporal associations among multiple updated objects during split, complete occlusion, partial occlusion, and multiple contacts dynamic interaction cases. The performance of the method was examined in terms of the tracking accuracy and computational efficiency by various experiments, achieving over 97% accuracy with five frames per second computation time. The problem with this solution is, that it only works if the hand and the object are not in touch all the time. Therefore this approach would be not able to segment the hand from the object.

Another approach [SMZ$^+$16] uses a 3D articulated Gaussian mixture alignment strategy tailored to hand object tracking that allows fast pose optimization. The alignment energy uses novel regularizers to address occlusions and hand-object contacts. For added robustness, they guide the optimization with discriminative part classification of the hand and segmentation of the object. The problem on this approach is, that it requires a calibration of the users' hand shape, which is currently a manual process and can not be done in our environment.

### 2.1.1 Conclusion

As we can see there is no ready to use solution for our project. So we have to divide the problem. A very important part of our project is detecting the hand in the image. If we can find the hand, we can detect the object and segment out the hand from the object. There is also a lot of research alreday done on hand-detection on which we want to take some closer look.

## 2.2 Hand detection

A solution for hand detection [Han] is based on color recognition. The program is initialized by sampling color from the hand, which is shown in figure 2.2. The hand is then extracted from the background by using a threshold using the sampled color profile. Each color in the profile produces a binary image which in turn are all summed together. A nonlinear median filter is then applied to get a smooth and noise free binary representation of the hand. In the final step the convex hull is calculated. We evaluated the program, as seen in figure 2.3. The program works really well, but the big problem is the initialization step, which is shown in figure 2.2. The program provides only a timer, which triggers the snapshot of the hand. So it is not so easy to take a good sample of the hand. In our environment the user does not have any screen, so its not possible to take a predefined snapshot of the hand.

Another approach would be to use the NiTE [NiT] framework from Primesense, which takes also 3D information into account. It gives us the ability to track the hand and also the skeleton of a user. We also tried this with our Kinect, which is shown in figure 2.4. In an initialization step the user in front of the camera has to wave with his hand. After this step if the NiTE framework detects successfully the hand, a coloured point is drawn on the depth image. Also a line is drawn to the coloured point, which shows the tracked path of the detected hand, which is shown in figure 2.5. As seen in figure 2.4 it works really fine. Also if the user has an object in the hand like in figure 2.5, not the hand is tracked, only the object, which would be also fine for our project. The problem is the robustness of this application. As seen in figure 2.6 and in figure 2.7 that no coloured point is drawn. This means that the tracking dit not work in these cases. Another problem is, that there is no distinction between an object in the hand and the hand itself. So its not suitable for our project.

As already mentioned, there is also the possibility to use a skeleton tracker with NiTE,

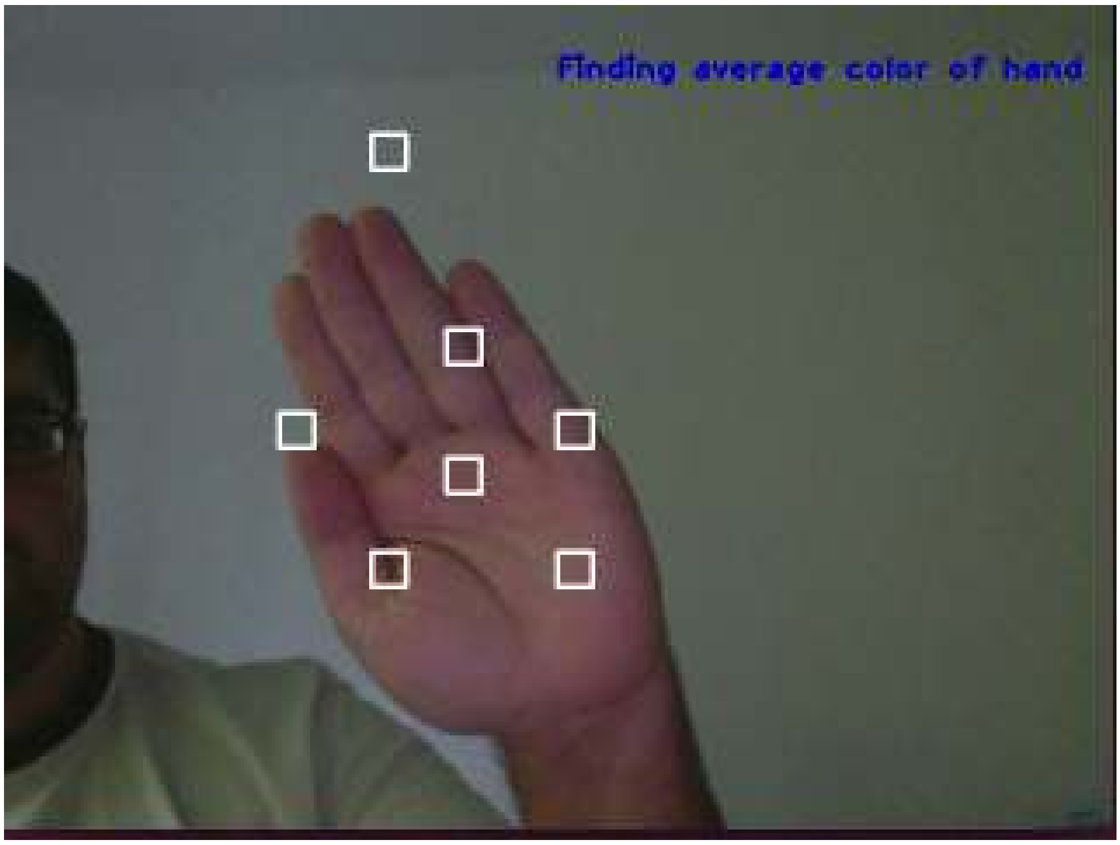
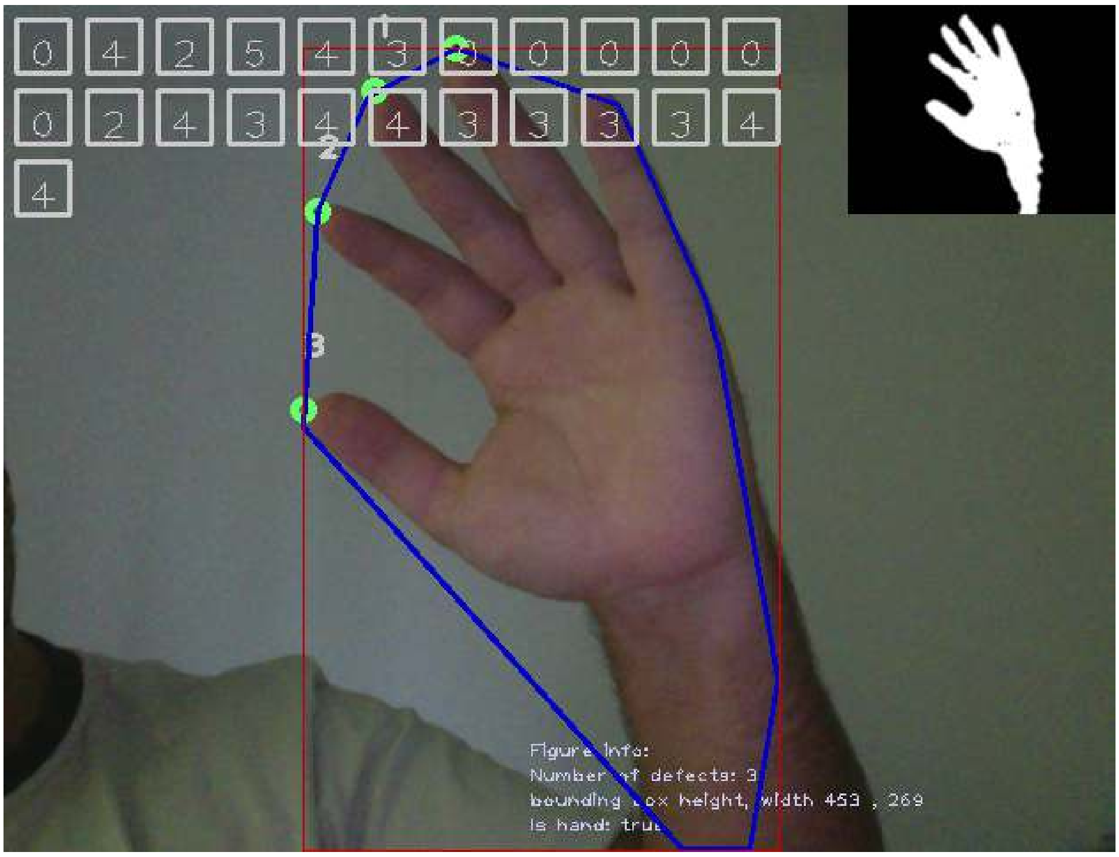


Figure 2.2: Initialization step, taking color samples of the hand.

which is shown in figure 2.8. If the skeleton is found by the NiTE framework a coloured line is shown in the image. In figure 2.8 the purple line shows the estimated skeleton. This method works really fine, if the whole body is captured by the camera. In figure 2.9, figure 2.10 and in figure 2.11 are shown, if only parts of the body seen by the camera it does not work very well. Especially in our case it is really difficult to set up a condition for the user to show the whole body on the camera. This means that the user has to ensure that she/he has a minimal distance to the camera, which is about 2-3 meters. Therefore a skeleton tracker is not usable for our project.

Another approach [TST$^{+}$15], presents a robust method for capturing articulated hand motions in realtime using a single depth camera. The system is based on a realtime registration process that accurately reconstructs hand poses by fitting a 3D articulated hand model to depth images. They register the hand model using depth, silhouette, and temporal information. To effectively map low-quality depth maps to realistic hand poses, they regularize the registration with kinematic and temporal priors, as well as a data-driven prior built from a database of realistic hand poses. We also tried this approach, which is shown in figure 2.12. This method is has the best result compared with the previous hand tracking methods. But it has two big disadvantages. The first



```
Figure Info:
Number of defects: 3
bounding box height, width 453 , 269
Is hand: tru
```

Figure 2.3: Result of the OpenCV hand detection.

is, the user needs a wristband, that this method works. Also the color of the wristband has to be adjusted manually. And the second big problem is, that it does not work with objects in hand, which is shown in figure 2.13. With these two limitations it is not suitable for our project.

Another paper [YGHS$^+$17] discusses the performance of 11 state-of-the-art methods on hand pose estimation. They provide an evaluation on three different tasks, single frame 3D pose estimation, 3D hand tracking, and hand pose estimation during object interaction. They analyze the performance of different convolutional neural networks (CNN) structures with regard to hand shape, joint visibility, view point and articulation distributions. As testbed for their analysis they use The Hands In the Million challenge [YYGHK17], which provides a big dataset of hand images. One problem is, that they do not talk about execution time. So we do not know how long this methods run on one scenario. Another problem is that the images shown in the paper are very simple compared to our scenario. This means in their images the hand is always visible and a big part of the image. In our scenario only parts of the hand or the fingers are shown.

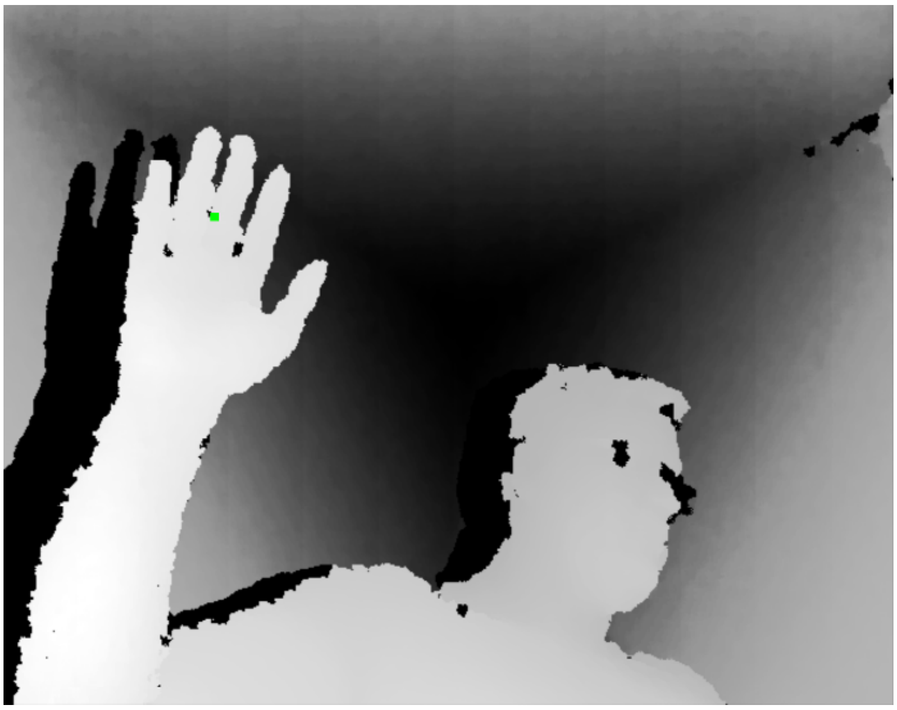


Figure 2.4: Using the NiTE framework for hand detection.

### 2.2.1 Conclusion

There is no unique recipe for hand detection. Some of them use color information and some are using 3D information to get the hand by its shape. To decide which concept is better suitable for our project, we decided to do a study, how people are holding objects in front of a camera, which is shown in the next chapter.

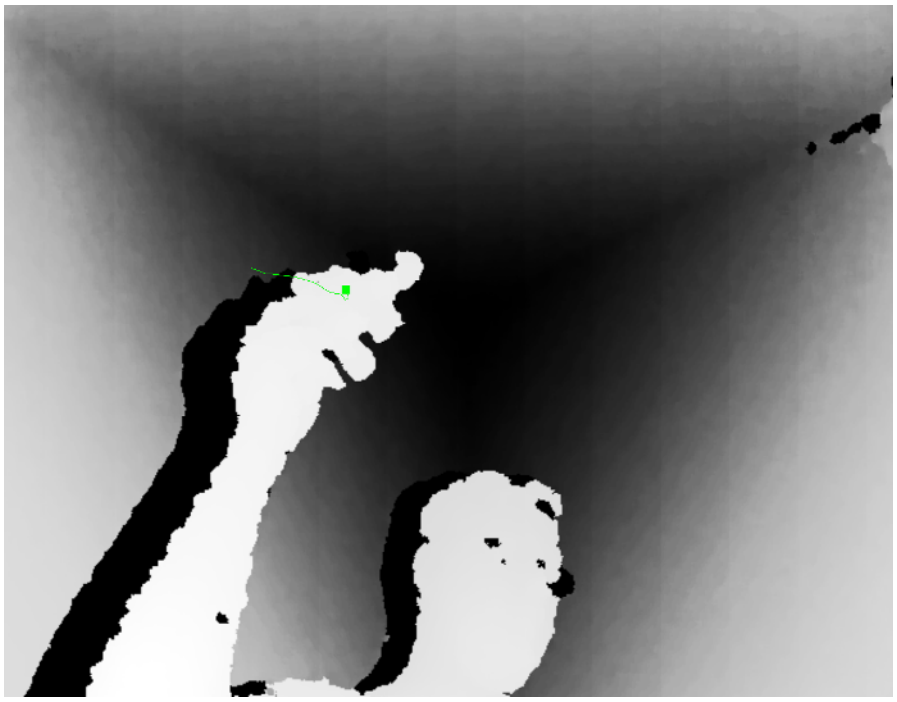


Figure 2.5: Using the NiTE framework for hand detection with an object in the hand.

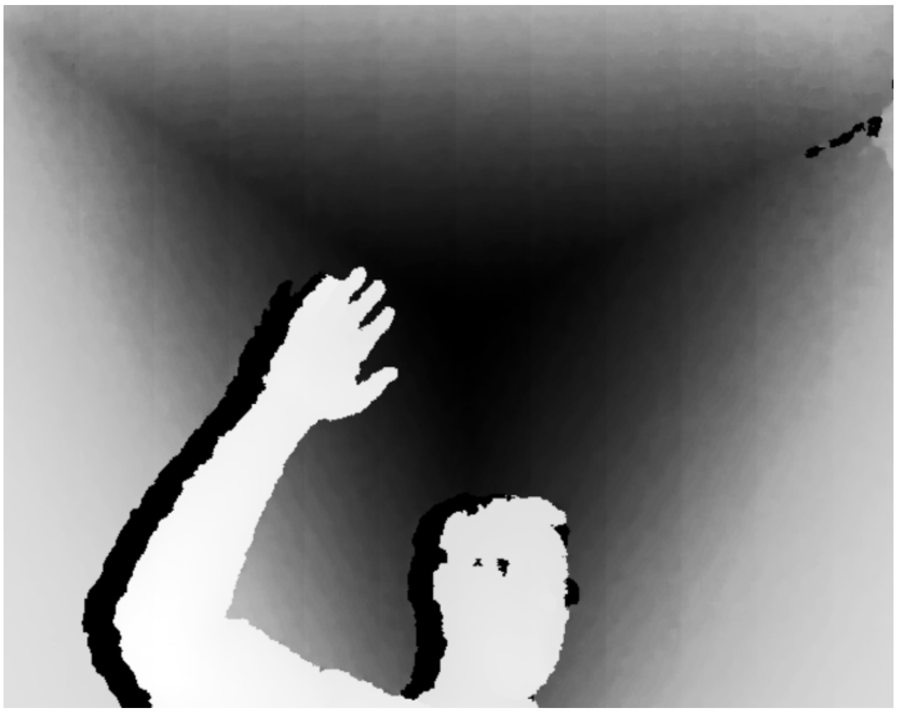


Figure 2.6: Hand is not detected with NiTE

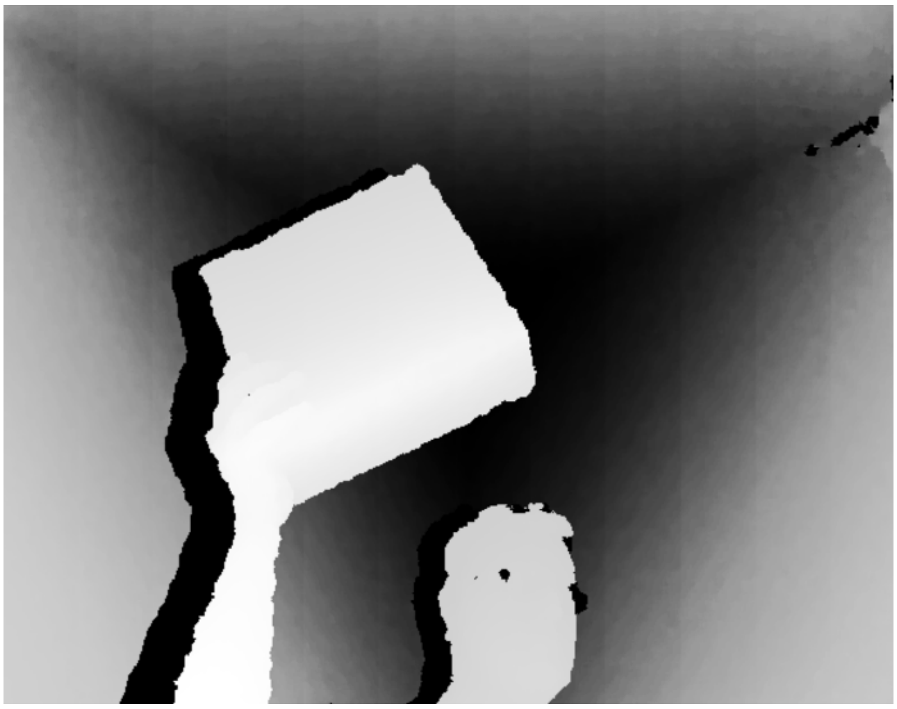


Figure 2.7: Hand is not detected with NiTE.

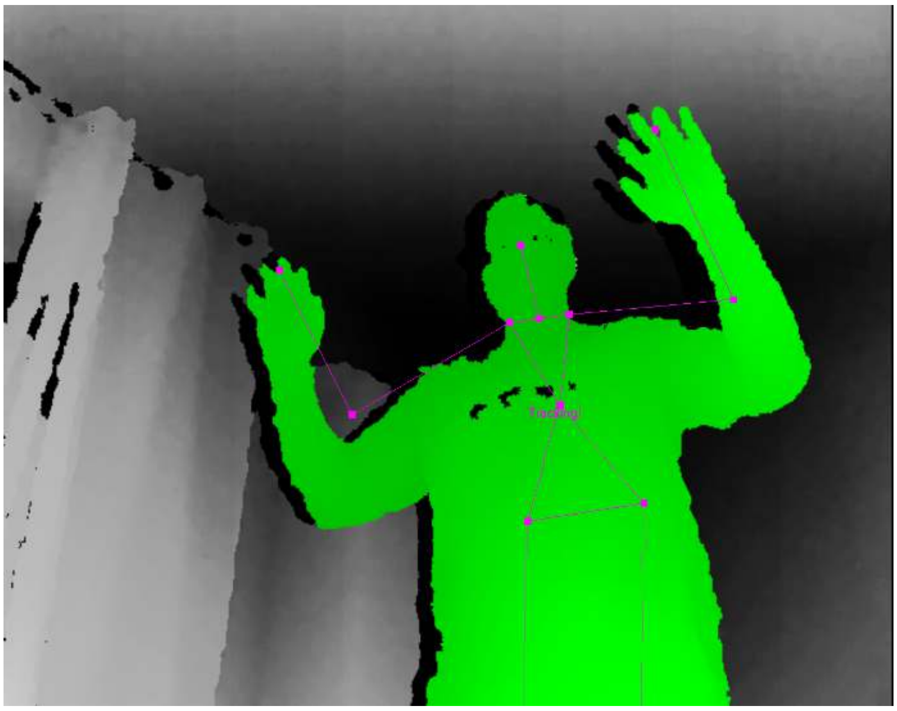


Figure 2.8: Using the NiTE framework for skeleton tracking.

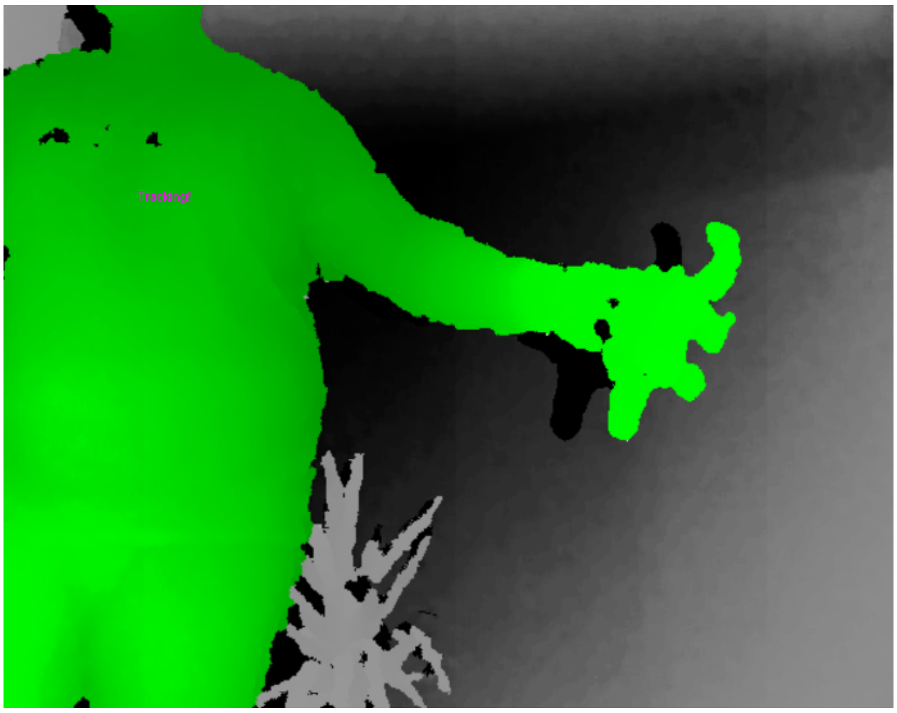


Figure 2.9: Skeleton tracking failed, case 1.

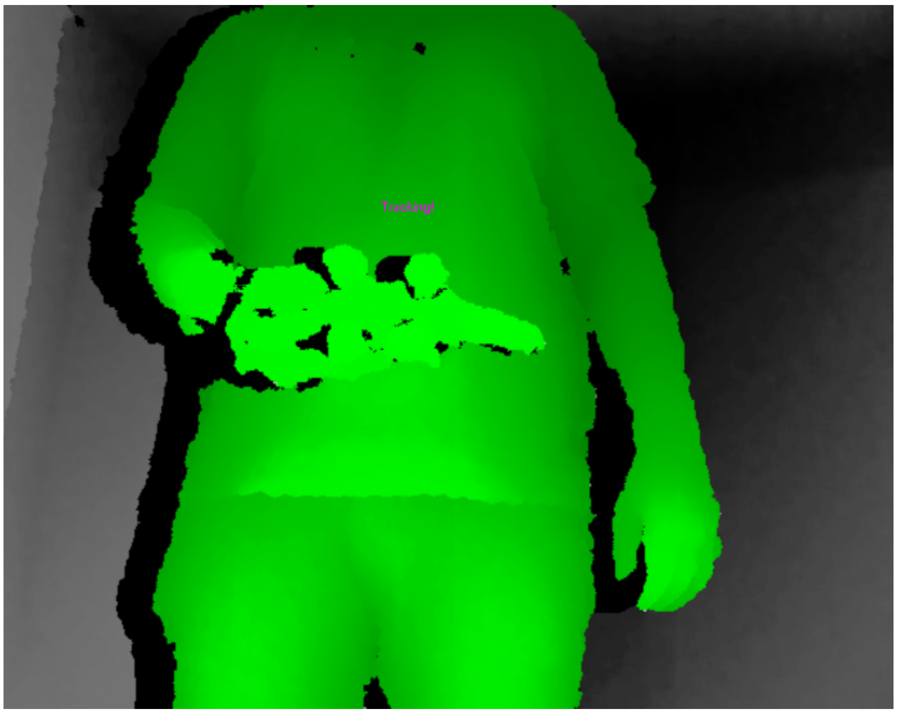


Figure 2.10: Skeleton tracking failed, case 2.

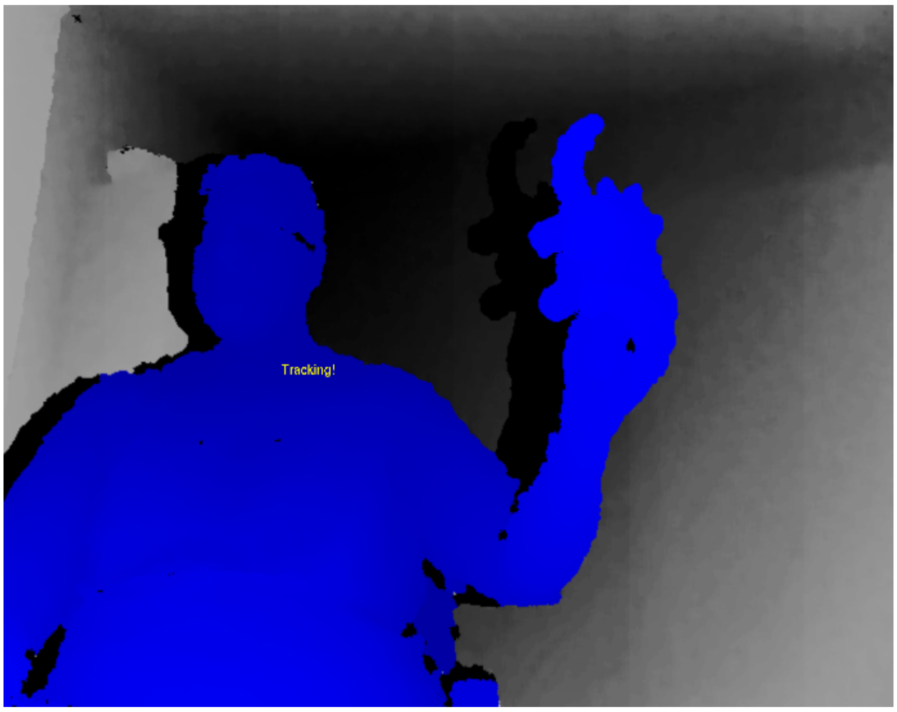


Figure 2.11: Skeleton tracking failed, case 3.

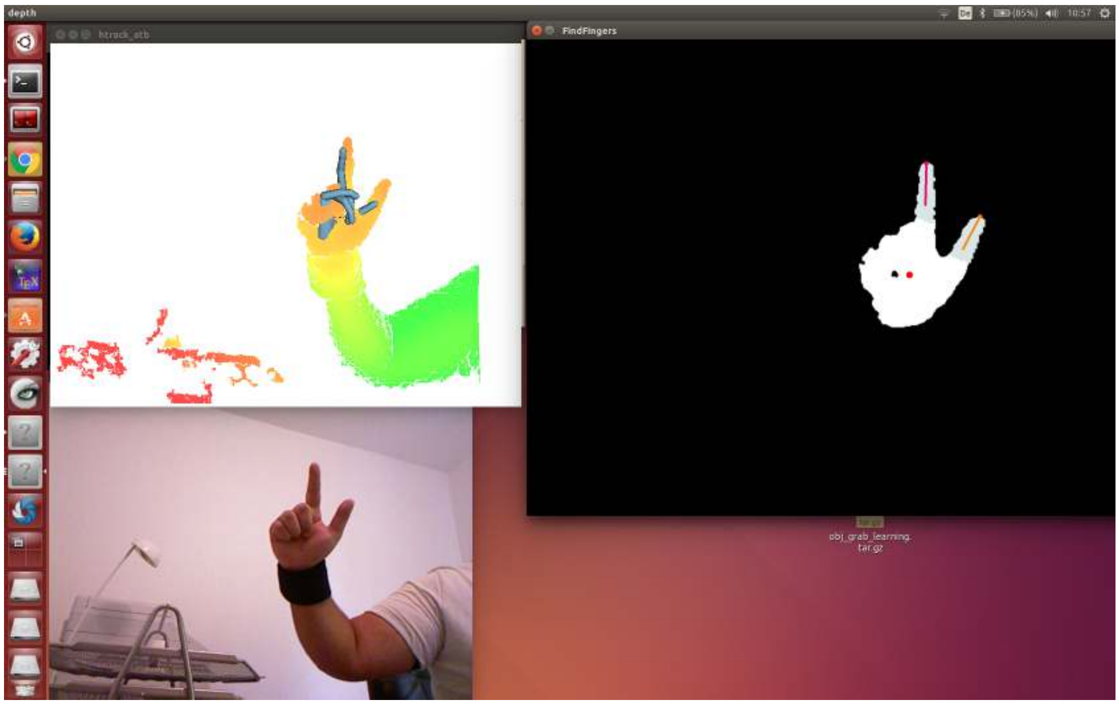


Figure 2.12: Hand tracking with htrack

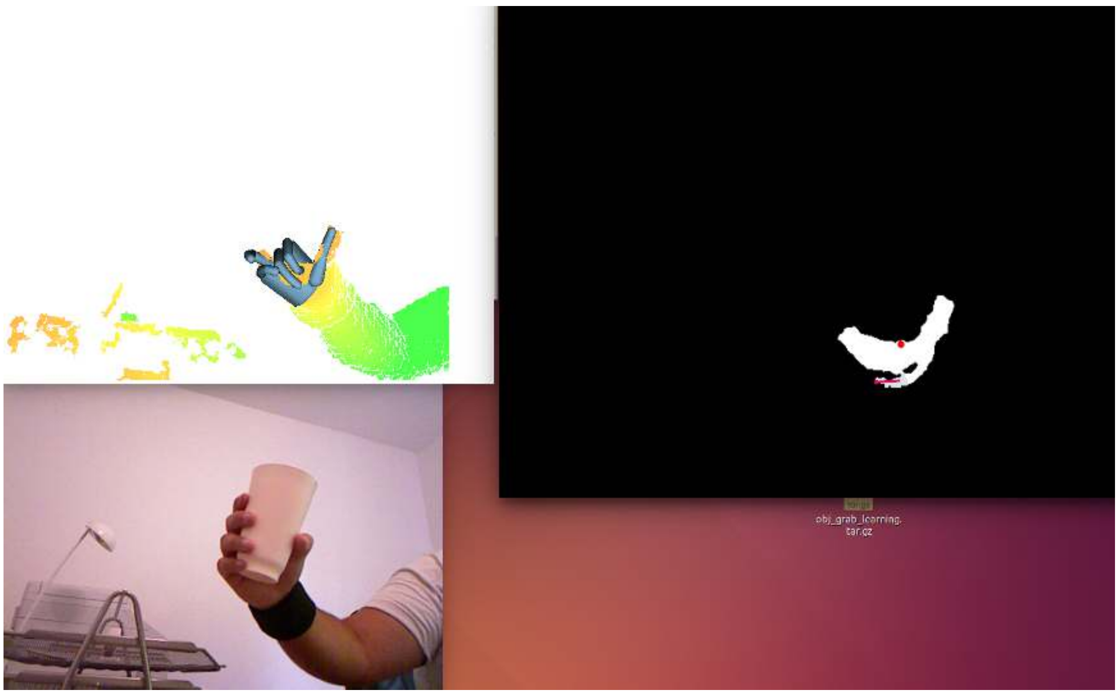


Figure 2.13: Hand tracking with htrack, with an object in the hand

CHAPTER 3

# Study on object grasping

To find a robust solution for our problem, we studied the cases how people are interacting with objects in front of the camera. The study will show us, how people are grasping different types of objects. It also shows us, how people position themselves in front of the camera and which body parts are seen by the camera. This gives us an initial set of conditions, which we have to cover and shows us which preconditions are useful to get a good image from the RGB-D camera. For our study we used four different objects. The test person is also doing the test one time sitting and one time standing in front of the camera.

## 3.1 Environment of the study

### 3.1.1 Camera

For the study we used the Kinect, which is an RGB-D camera from Microsoft. We installed it on a height of 85cm and ensured, that there are no obstacles between the camera and the test person. We also ensured that there is enough space in front of the camera for the test person.

### 3.1.2 Conditions

For the study we have to ensure that the object is visible for the camera. Therfore we needed some conditions for the test persons, which are described below.

- Hold only one object at the same time in front of the camera.
- Try to hold the object in the center of the camera view.
- The feedback shows you, if the camera can see the object.

#### Feedback

As already mentioned in the conditions section above, we needed some kind of feedback, that the test person knows, if the camera can see the object. Especially the Kinect has a gap of about 60cm in front of the camera, where no depth data is available. In the big picture, the application should run on a robot, where a feedback is limited. Therefore we decided to perform a case study with two cases. In case study A the test person only gets a binary feedback, if the camera can see the object. In case study B the test person gets a full feedback of the current view of the robot.

## 3.2 Case study A

### 3.2.1 Study description

Please hold each object one after another in front of the camera. Try to hold the object in the center of the camera view. Also try to hold the object not too near and not too far away from the camera. The image on the screen will show you, if the camera can see the object.

If the image is **red**, the camera can not see the object. Otherwise if it is **green** the camera can see the object.

Please do this process one time sitting on the chair and one time standing in front of the camera.

### 3.2.2 Images

In the figures 3.1 - 3.6 can be seen the results from case study A. In the first column the test person is sitting in front of the camera. In the second column the test person is standing in front of the camera.

## 3.3 Case study B

### 3.3.1 Study description

Please hold each object one after another in front of the camera. Try to hold the object in the center of the camera view. Also try to hold the object not too near and not too far away from the camera. The image on the screen will show you, if the camera can see the object.

If the object disappears in the image the camera is not able to see it.

Please do this process one time sitting on the chair and one time standing in front of the camera.

### 3.3.2 Images

In the figures 3.7 - 3.11 can be seen the results from case study b. In the first column the test person is sitting in front of the camera. In the second column the test person is standing in front of the camera.

## 3.4 Conclusion

We can see that people are holding different objects in many different ways. Some persons are holding the object with both hands and others with one hand. In conclustion there is a wide diversity, how objects could be held. Therefore we are not able to cover up all cases, how objects can be held. Especially if the hand masks the object or the object is held too close to another object or body part. In this case the clustering will fail. We also see that the type of feedback has no major effect, how objects are held in front of the camera. It has also no effect, if the person sits or stands. An important aspect we can see in the result of the study is, that in some cases the hand is only partially seen by the camera. In some cases only one or two fingers are seen by the camera. Therefore hand tracking approaches with kinematic hand models are not suitable and we decided to only rely on skin color for hand detection. In figure 1.1 the complete process of our method is shown.

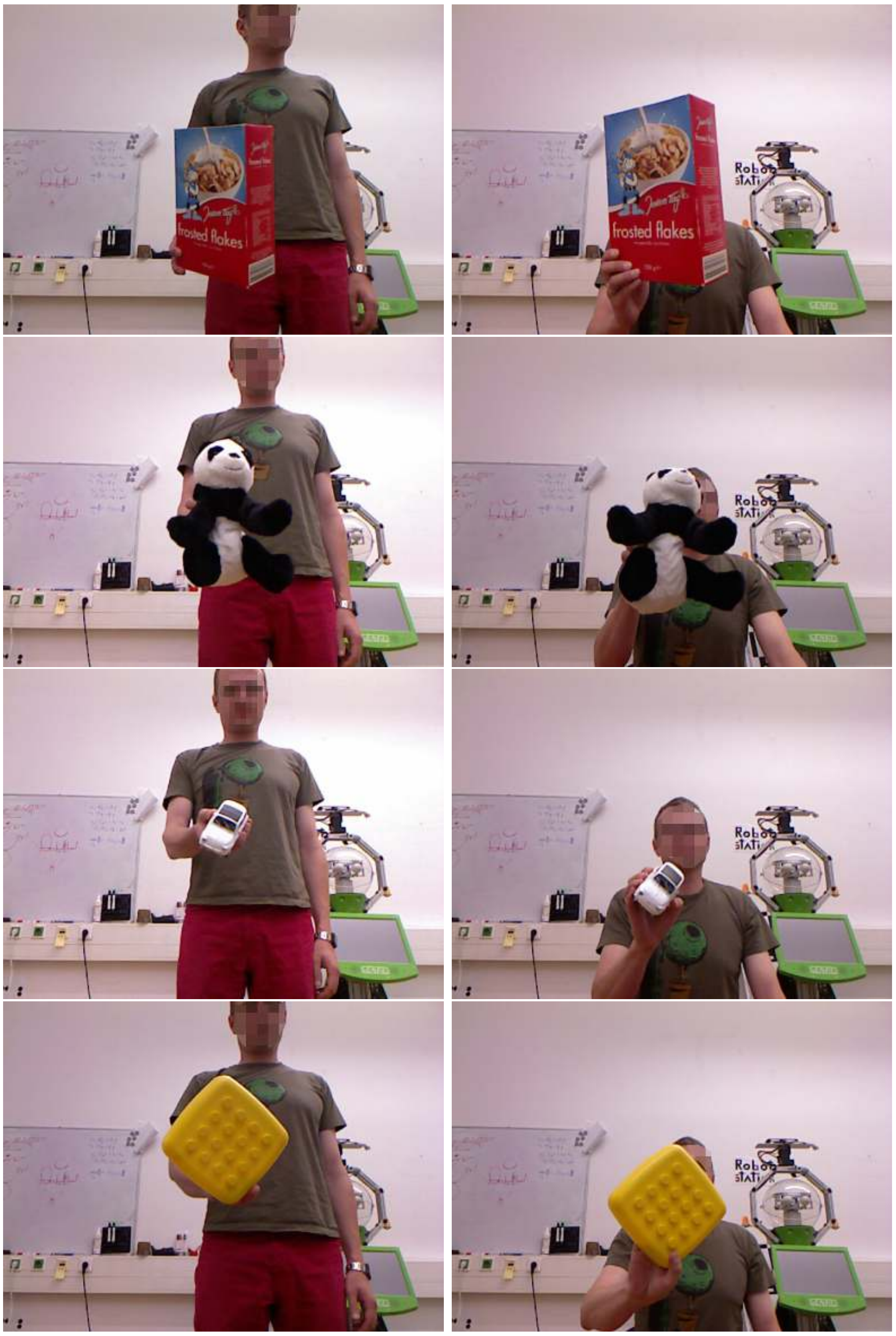


Figure 3.1: Study on grasping with binary feedback (I).

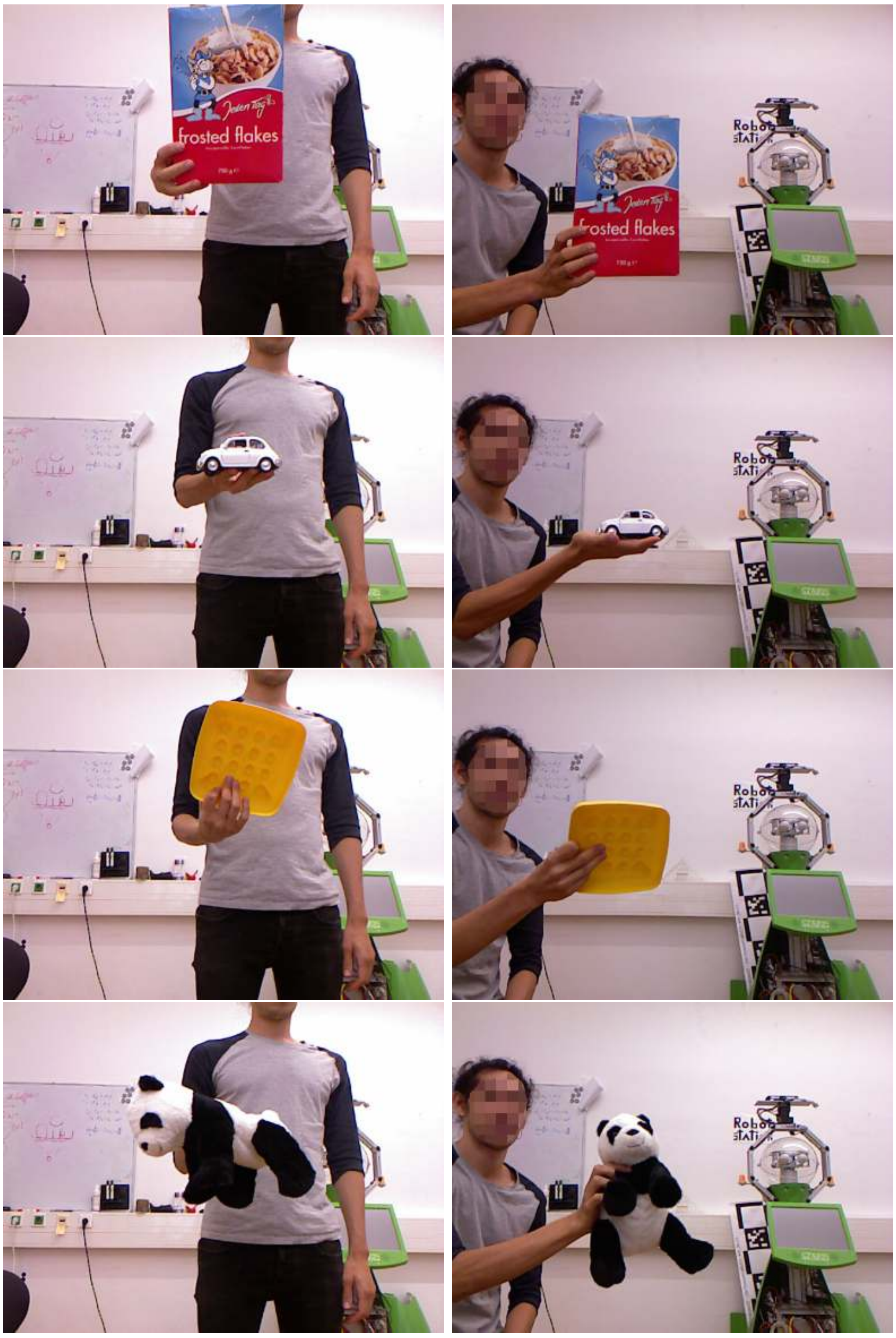


Figure 3.2: Study on grasping with binary feedback (II).

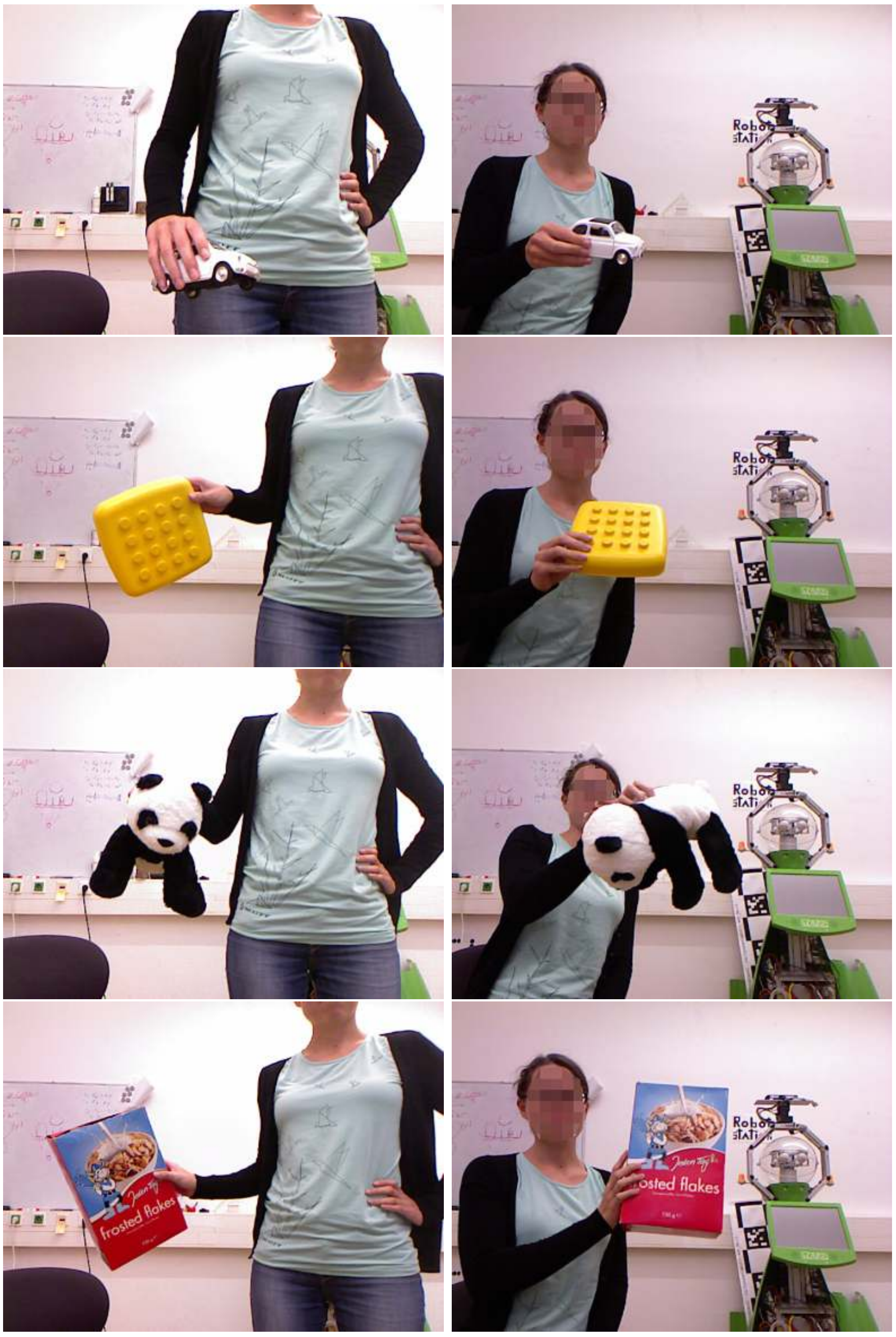


Figure 3.3: Study on grasping with binary feedback (III).

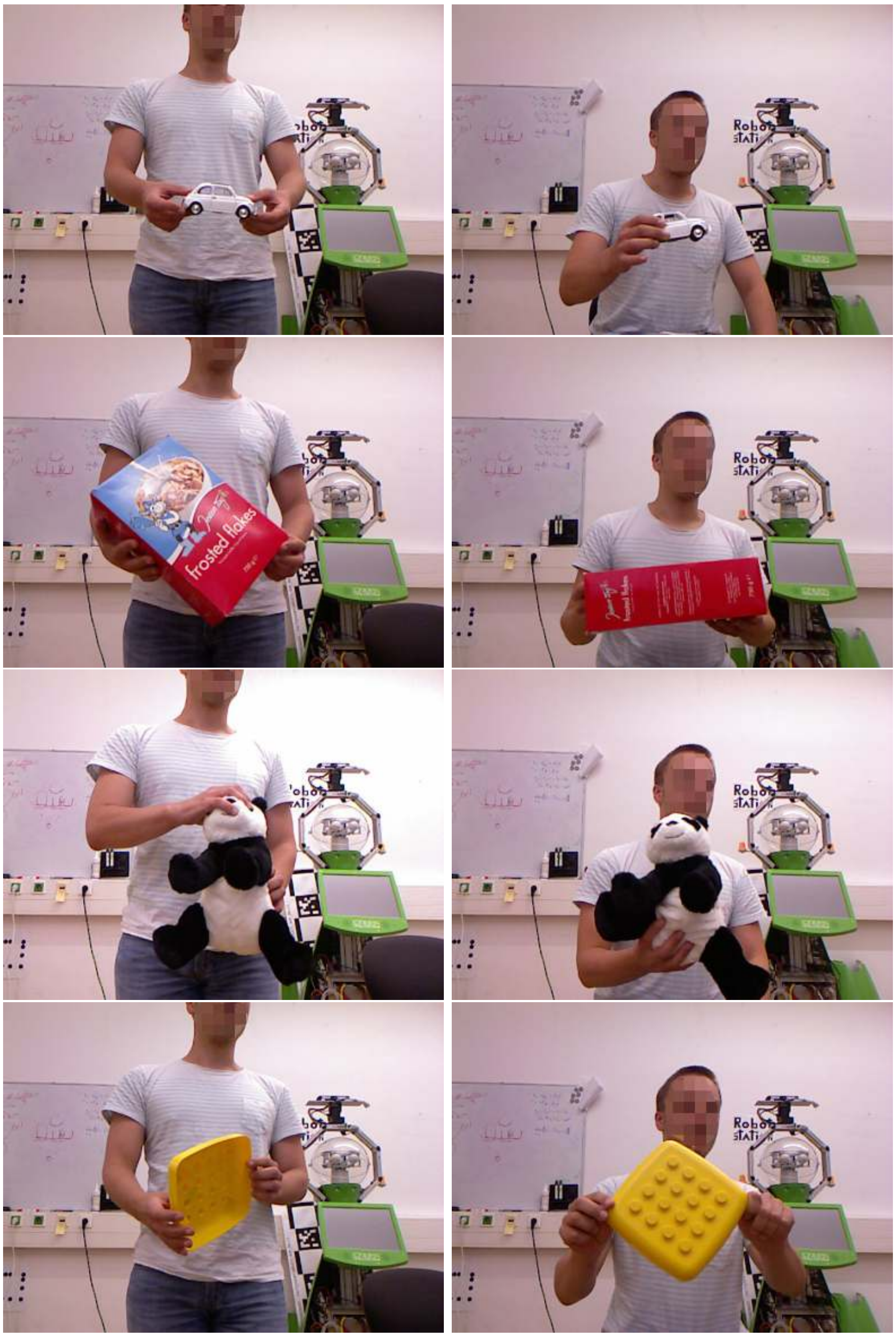


Figure 3.4: Study on grasping with binary feedback (IV).

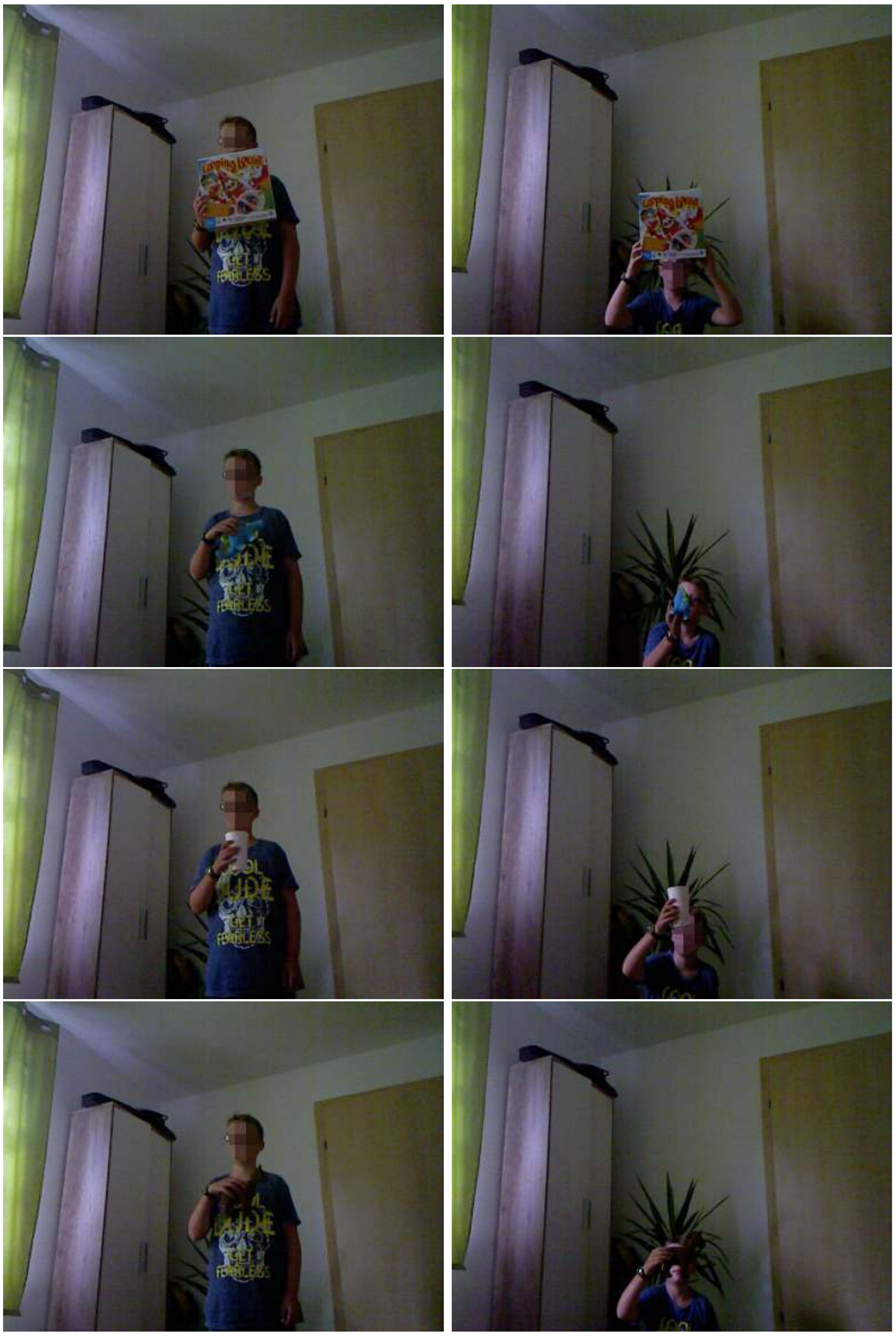


Figure 3.5: Study on grasping with binary feedback (V).

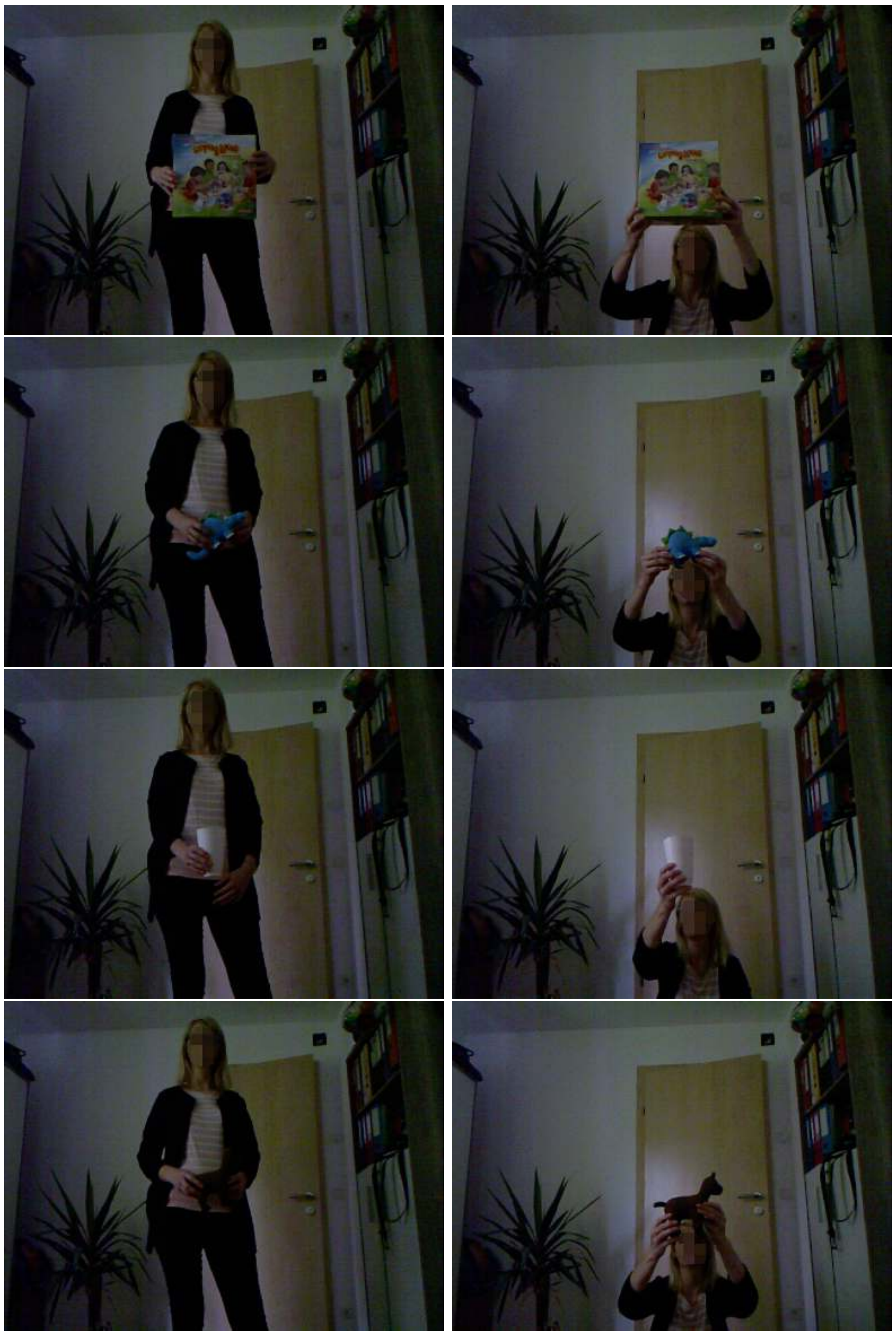


Figure 3.6: Study on grasping with binary feedback (VI).

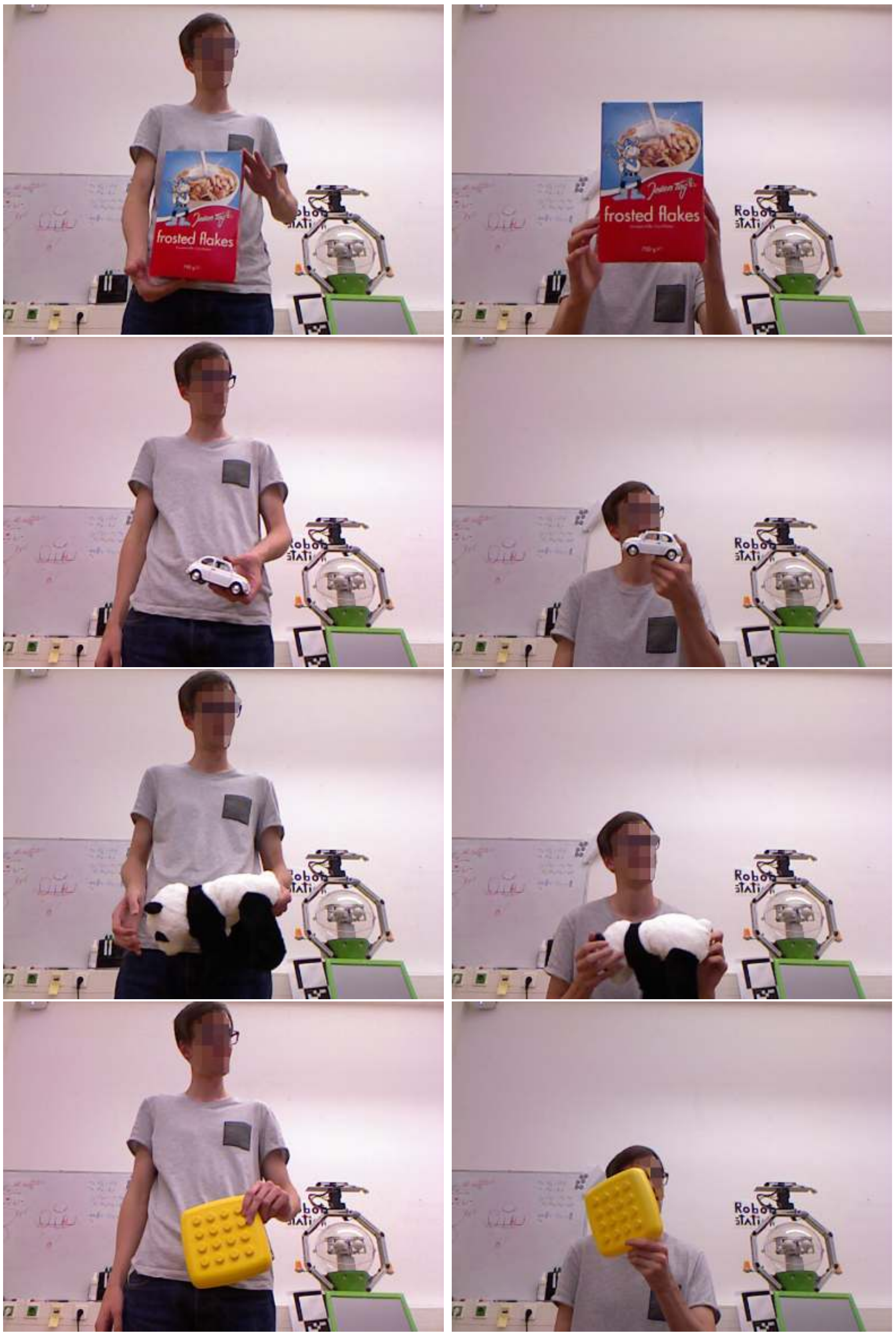


Figure 3.7: Study on grasping with full feedback (I).

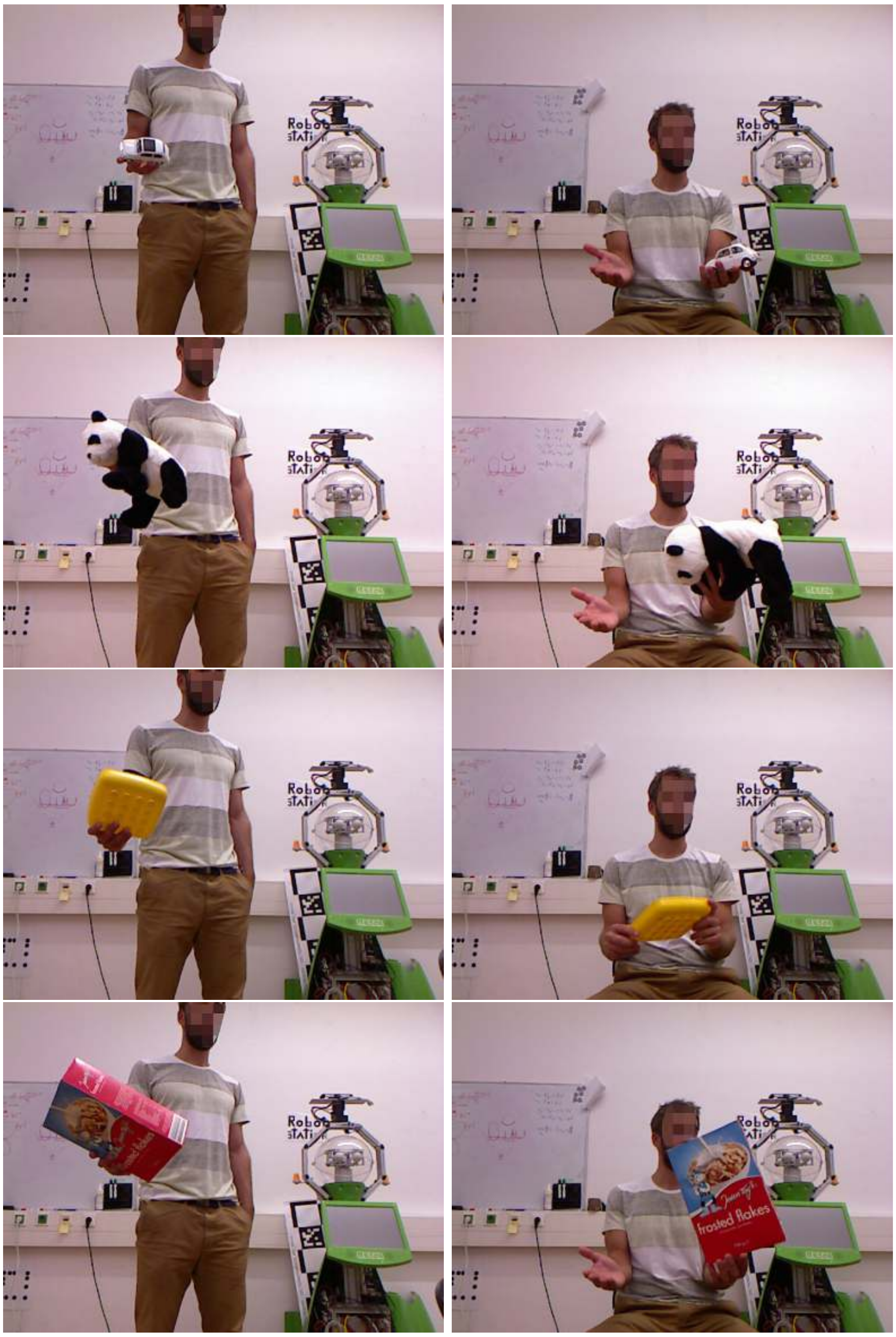


Figure 3.8: Study on grasping with full feedback (II).

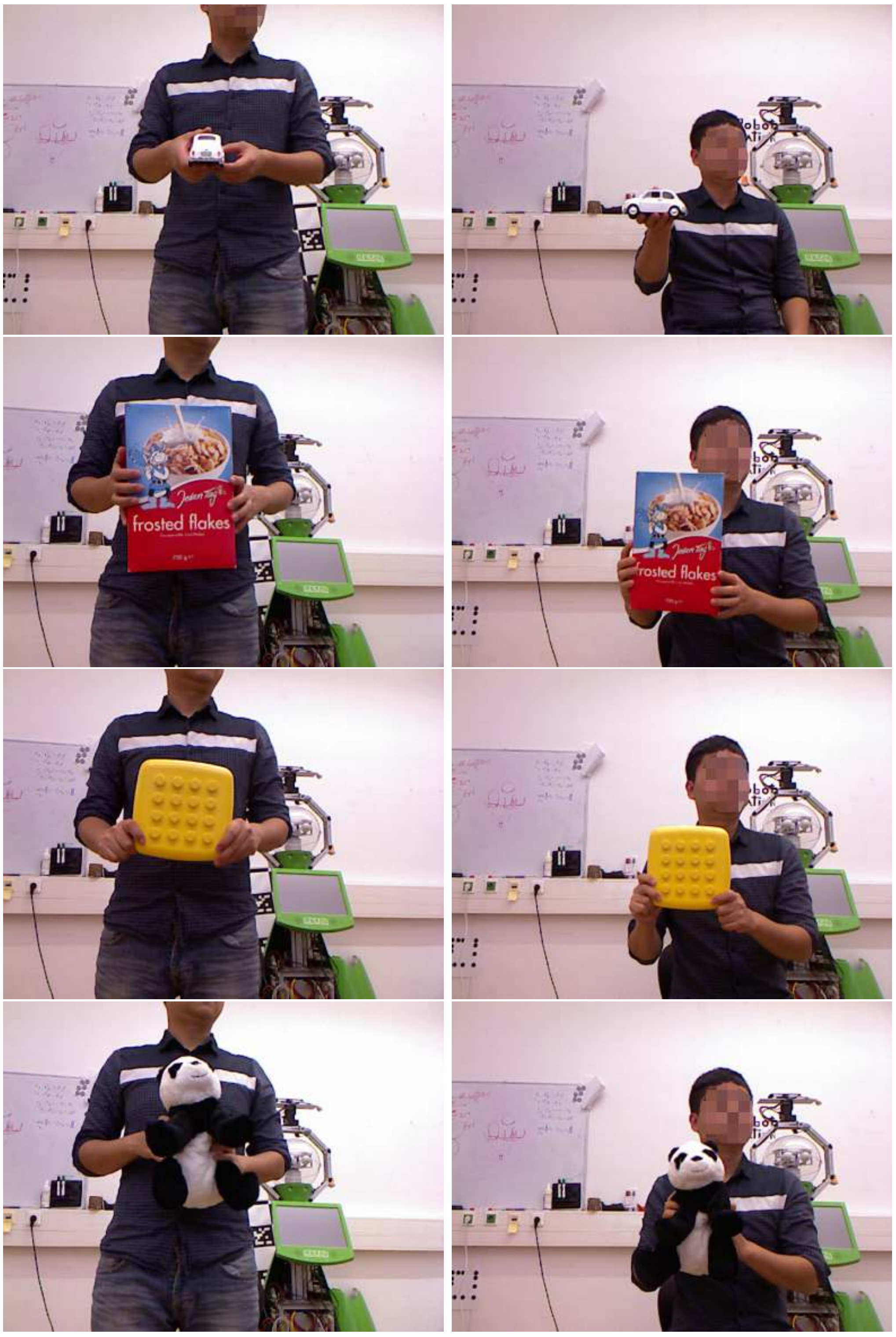


Figure 3.9: Study on grasping with full feedback (III).

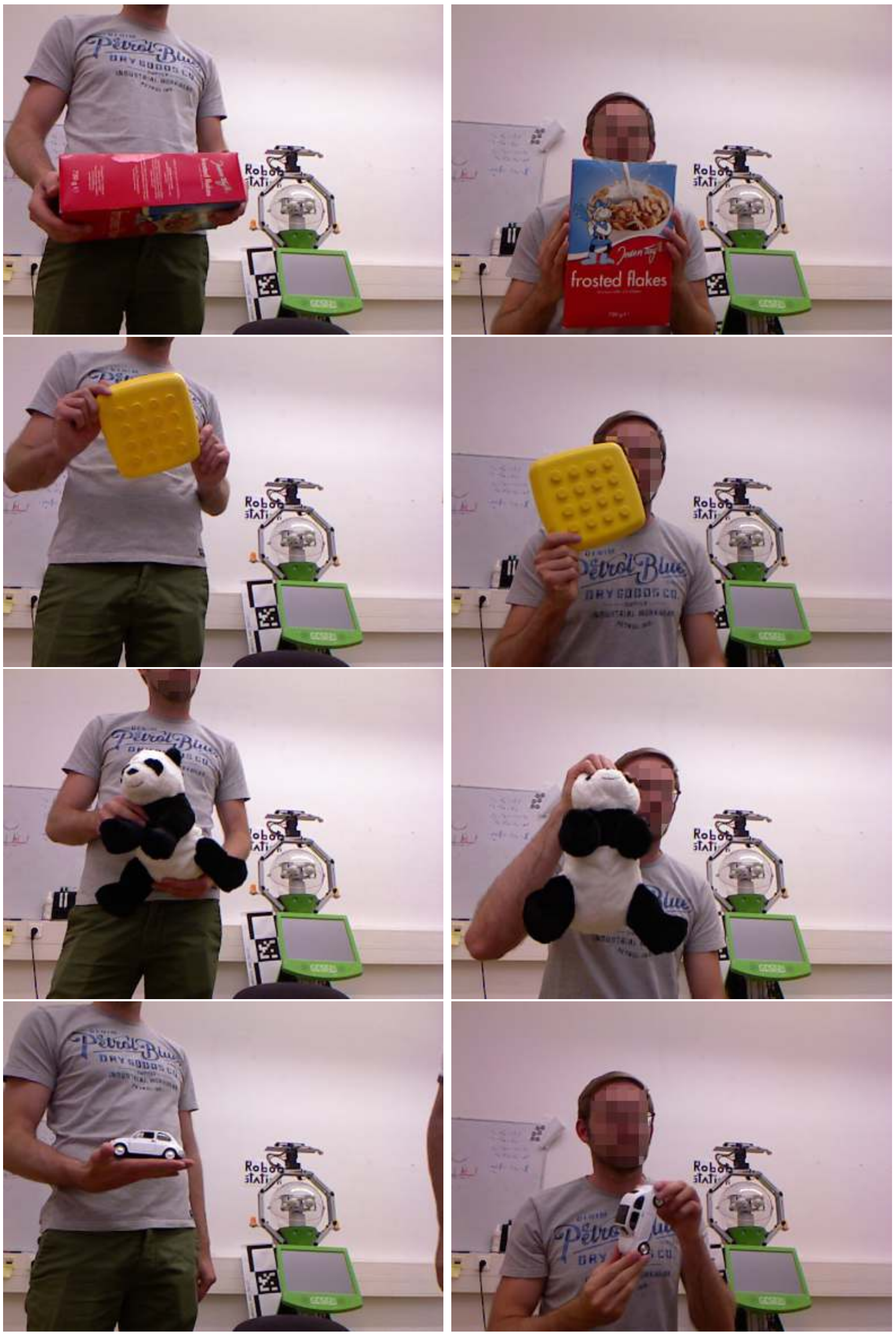


Figure 3.10: Study on grasping with full feedback (IV).

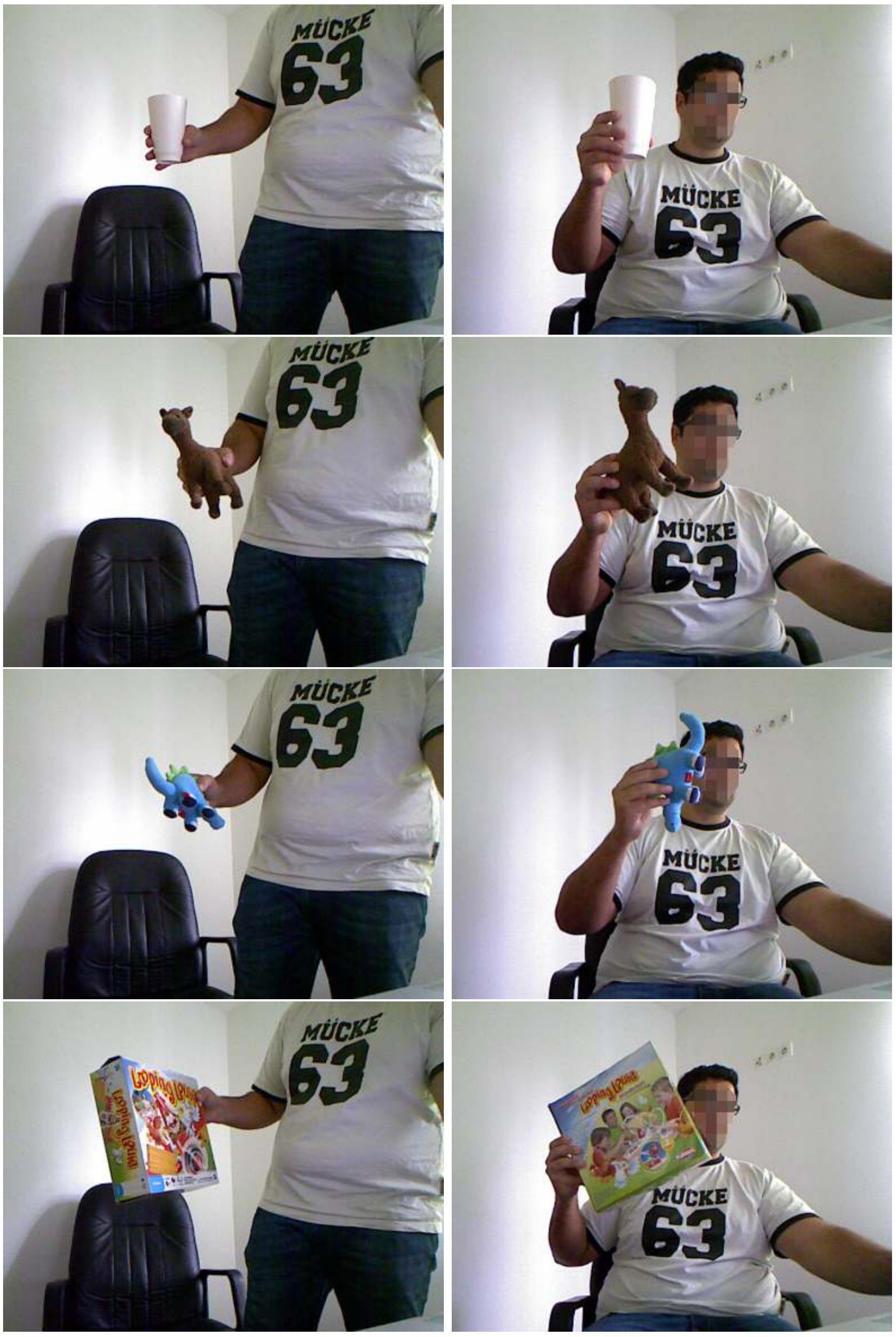


Figure 3.11: Study on grasping with full feedback (V).

CHAPTER 4

# Hand detection

To find the hand of the user we make use of the skin color. There are several skin color detection methods based on a scope of possible skin colors available [RMG97] [VSA03]. They provide equations for the skin detection in different color spaces. Another method [LzP10] compared the algorithms based on the three colorspaces RGB, $\mathrm{YC_hC_r}$ and HSV and combined them into a new algorithm to increase the accuracy. A big problem with these skin detection methods is the dependency on camera settings, illumination, people's tans and ethnic groups. [KMB07]. Therefore many false positives and false negatives in the skin detection can occur [VSA03]. This would lead to a bad object segmentation. To get a more robust skin detection, we decided to use an adaptive skin-colour-model based on the user's face [CCH10] [MZT11]. Especially face detection using Haar Cascades [VJ01] [LR03] works very well and can be easily applied. This method uses a grayscale image as input, therefore it has not so much effect on illumination and skin color.

## 4.1 Face based Adaptive Skin Color Model

In our project we are using the face detection algorithm provided by OpenCV, which uses the Haar Cascade method to get an individual skin color model for the person. In figure 4.1 and figure 4.2 the face detection applied in our test scenario are shown. The pink circle around the face represents the detected face. If we have found a face, we continue with the algorithm described in [CCH10] to get a individual skin model of the person in front of the camera. To get rid of the eyes, hair and other dark regions in the detected face region, we make use of the luminance distribution of the pixels in the face region. Observing the histogram of the face, darker pixels like eyes and hair are located at the extreme left [CCH10]. At the other side we get the mean value located at the peak of the luminance histogram, which is shown in figure 4.3 based on the image in figure 4.1. To remove the darker pixels of the face region, we trim the left hand side by mirroring the right hand side. In figure 4.3 this is shown by the green lines. All pixels outside of these two lines are ignored. In figure 4.4 can be seen the filtered face, where the black regions are representing the pixels which are filtered out.

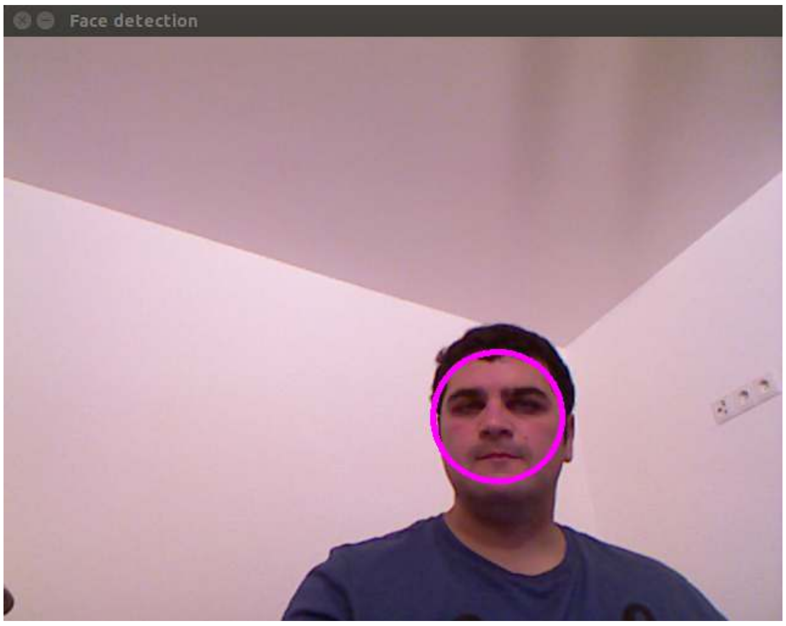


Figure 4.1: Face detecion using Haar Cascade method.

For our skin color model we use the R component of the RGB color space and the normalized RG colors (r,g). With these three components we set up our adaptive skin color model (r,g,R), which is less sensitive to changes in light source [CCH10]. In equation (4.1) and (4.2) it is shown how the normalized color values are calculated. In figure 4.5 the color histogram of the three channels r,g and R are shown, which can be modeled by three narrow Gaussian distributions. To get a symmetric distribution, we also mirror the right hand side to the left hand side, which is shown by the green lines. All pixels outside these lines are ignored. Afterwards we can model the skin colors by Gaussian distributions $GM_i(\mu_i, \sigma_i)$, where $\sigma_i$ is the standard deviation and $\mu_i$ is the mean, $i = r, g$, and $R$. $\mu_i$ and $\sigma_i$ are calculated as in 4.3 and 4.4, where $n$ is the number of pixels in the trimmed face region [CCH10].

$$r = \frac{R}{R+G+B} \tag{4.1}$$

$$g = \frac{G}{R+G+B} \tag{4.2}$$

$$\mu_i = \frac{1}{n} \sum_{(x,y)\in face\ region} I_i(x,y), \quad i = r, g\ and\ R \tag{4.3}$$

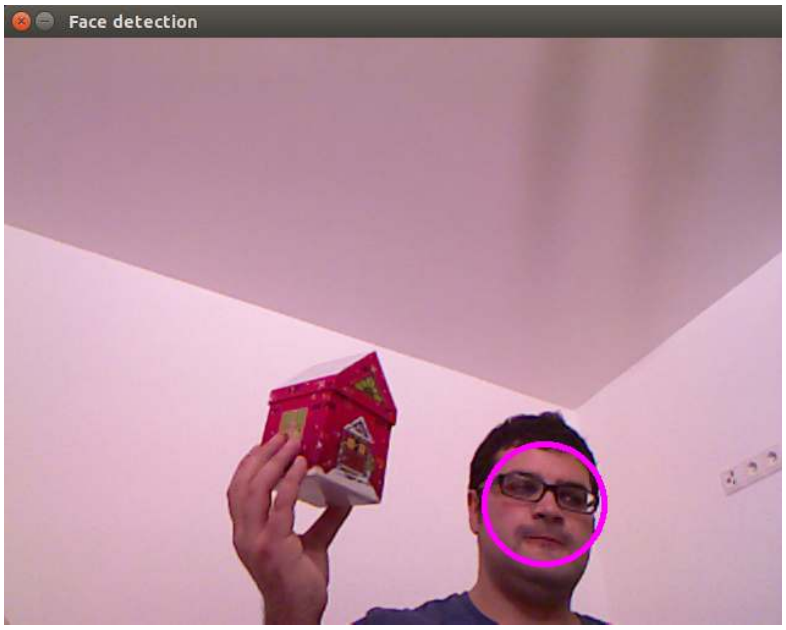


Figure 4.2: Face detecion with object in hand.

$$\sigma_i = \sqrt{\frac{1}{n} \sum_{(x,y)\in face\ region} (I_i(x,y) - \mu_i)^2}, \quad i = r, g\ and\ R \tag{4.4}$$

After we calculated $\sigma_i$ and $\mu_i$ where $i = r, g$ and $R$, we can set the upper and lower limits for the skin color detection as described in Equations 4.5, 4.6 and 4.7. In figure 4.6 the result of our skin detection in based on figure 4.2 is shown.

$$UpperBound_i = \mu_i + 2 * \sigma i, i = r, g\ and\ R \tag{4.5}$$

$$LowerBound_i = \mu_i - 2 * \sigma i, i = r, g\ and\ R \tag{4.6}$$

$$skin = \begin{cases} UpperBound_r > r > LowerBound_r \\ UpperBound_g > g > LowerBound_g \\ UpperBound_R > R > LowerBound_R \end{cases} \tag{4.7}$$

## 4.2 Postprocessing

In figure 4.6 we can see that most parts of the skin are covered by the method, described above. But given that the object has an arbitary texture, parts of the object surface can be falsely detected as skin. Especially we often tend to observe a few skattered skin pixels on object surfaces. To remove this noise we apply an erosion and dilation [Opeb] on the detected skin mask. The result of the ersion and dilation based on figure 4.6 is

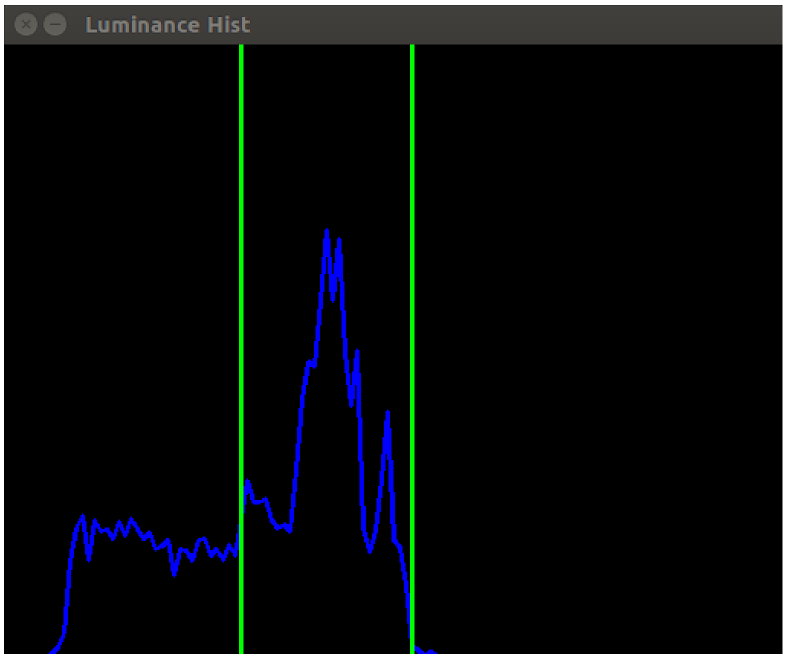


Figure 4.3: Luminance histogram of the face region.

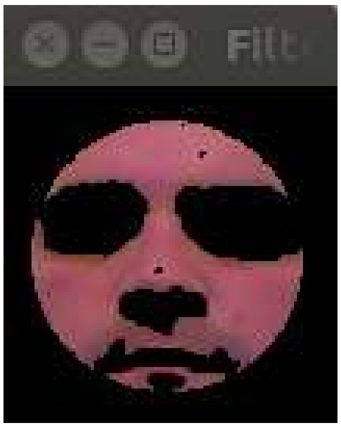
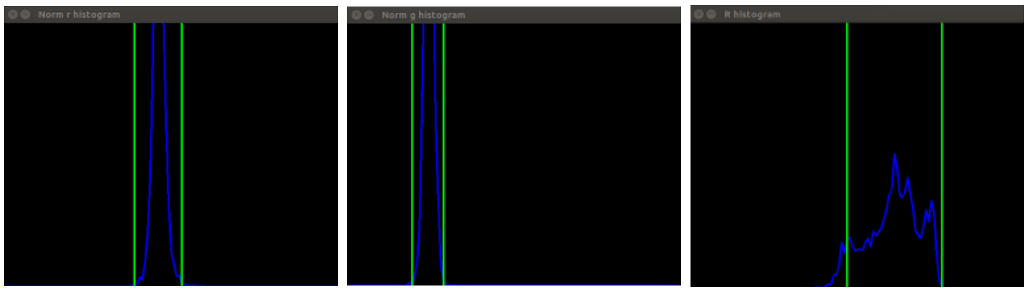


Figure 4.4: Image of the Face region after filtering darker pixels.

shown in figure 4.7. Another problem is the reliability of the skin mask. We can not ensure that all pixels of the hand, are detected as skin. For this reasons we are applying a Closing on the hand mask, which is the same as apply first a dilation and then an erosion on the mask ($erode(dilate(mask))$) [Opea]. After the Closing, all holes between skin pixels are filled up. To ensure to get rid of hand pixels on the border nearby the object, which are not effected by the closing method, we apply again a dilation on the skin mask, which is shown in figure 4.8. After this step we have an oversized hand mask, which is used to remove all hand pixels.



(a) r (b) g (c) R

Figure 4.5: Color histograms (r,g,R) of the face region.

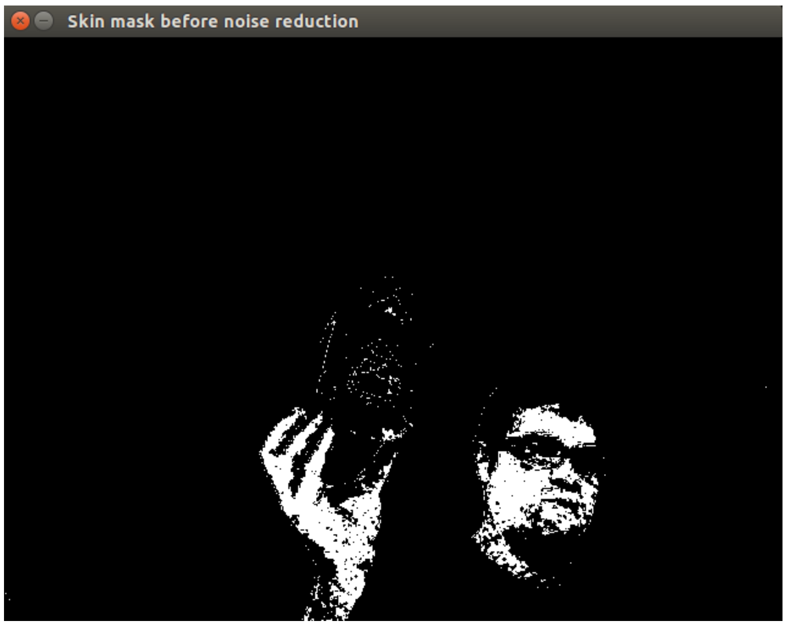


Figure 4.6: Skin detection, where white pixels represent skin pixels.

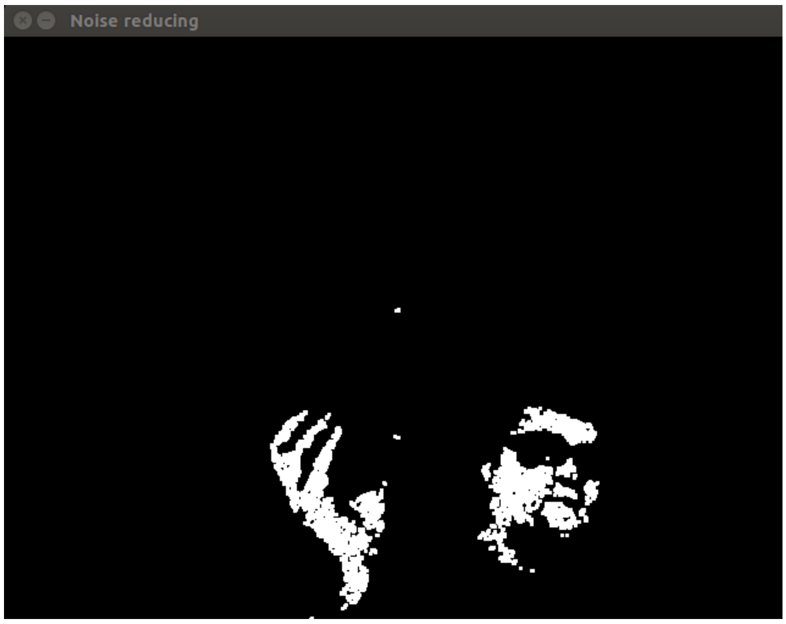


Figure 4.7: Skin mask after applying erosion and dilation.

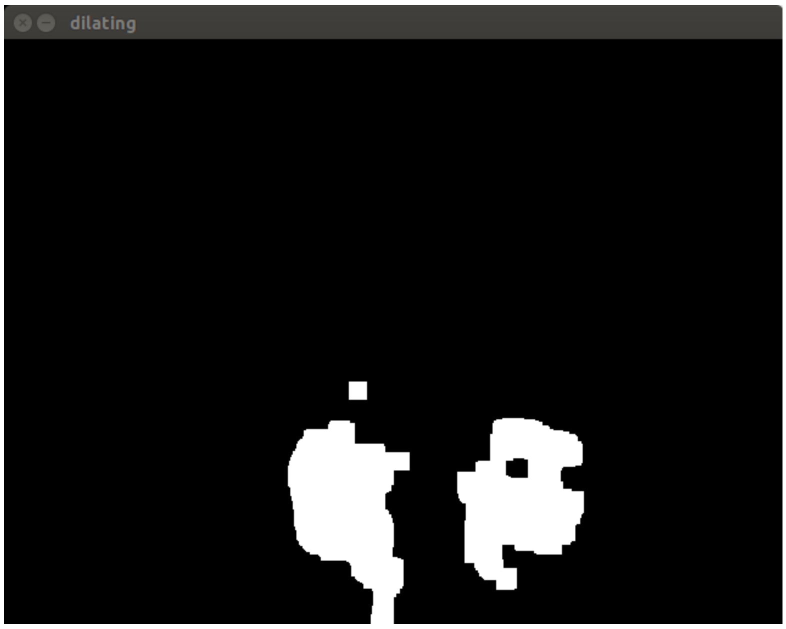


Figure 4.8: Picture of the hand mask after closing and dilating.

CHAPTER 5

# Object Segmentation

Now we have identified the hand of the user, the next step is to segment the object the user is holding. For the object segmentation we use the 3D information of the camera, searching for the nearest biggest cluster in front of the camera. For this purpose we need a full removal of the user's hand. If we apply the skin mask on our original 3D image, we can remove all skin pixels. After this step we get a 3D-image, where the object in the hand is fully separated from the hand.

To segment the object in the hand we apply euclidean clustering, starting from the nearest 3D point from the center of the camera. From this point we check all its pixel-wise neighbours (exploiting the fact that input point clouds coming from the Kinext are always organised as a rectangular grid). If the euclidiean distance is small enough the neighbour is added to the cluster and its respective neighbours are checked, otherwise it is rejected. This process is repeated until no more neigbours can be added. This process is shown in Alg. 5.1. The result of the euclidean clustering on our test scenario can be seen in figure 5.1, where the green circle marks the nearest point to the center of the camera view, which is also the start point of the clustering. After this step we have a first rough segmentation mask o fthe object. A reamaining problem is that some parts (larger than a few individual noise pixels, see Sec. section 4.2) of the object can actually be skin colored and thus falsely removed as skin pixels. To fix this problem we calculate the convex hull of the object mask. to get rid of falsely removed parts of the object. Another problem is that contours are often inaccurate, which is shown in figure 5.2 or parts of the hand are not removed. For this purpose we make use of Grabcut to get a better object segmentation [RKB04].

Grabcut was designed to get pixel-precise object (foreground) contours from some rough initialisation, e.g. samples of fore- and background strokes supplied by a user. In our case the above rough segmentation is used as initialistaion. In figure 5.1 the result of our clustering is shown. Also some parts of the object especially contours are falsely removed. To fix this problem, we dilate the mask, which is calculated by the clustering. After this step we get a bigger mask, which is shown in figure 5.3. Now the contours of the object are included. As side effect, also parts of the hand and the background are included, which will be removed by Grabcut. For this purpose we define all pixels of our original image as sure background except the dilated object mask (see figure 5.3). These pixels are denoted as predicted foreground. After applying the Grabcut-algorithm with one iteration we get an image, where the object is segmented, which is shown figure 5.4.

**Algorithm 5.1:** Euclidean clustering

```
Input: A matrix $Img$, a scalar $startPoint$
Output: A matrix $ObjMask$
$queue.push\_back(startPoint)$;
while $queue.size > 0$ do
    $currentPixel = queue[0]$;
    $queue.erase(0)$;
    // Iterate over all 2D neighbours of the current pixel.
    foreach $Neighbour\ n\ of\ currentPixel$ do
        if $Euclidean\ distance\ of\ n\ and\ currentPixel < maximal\ distance$ then
            // Check if the pixel is already part of the Object mask
            if $ObjMask.getPixel(n) == 0$; then
                // Add the neighbour to the queue and the object mask
                 $queue.push\_back(n)$;
                $ObjMask.setPixel(n) = 1$;
            else
                $continue$;
            end
        else
            $continue$;
        end
    end
end
return $ObjMask$;
```

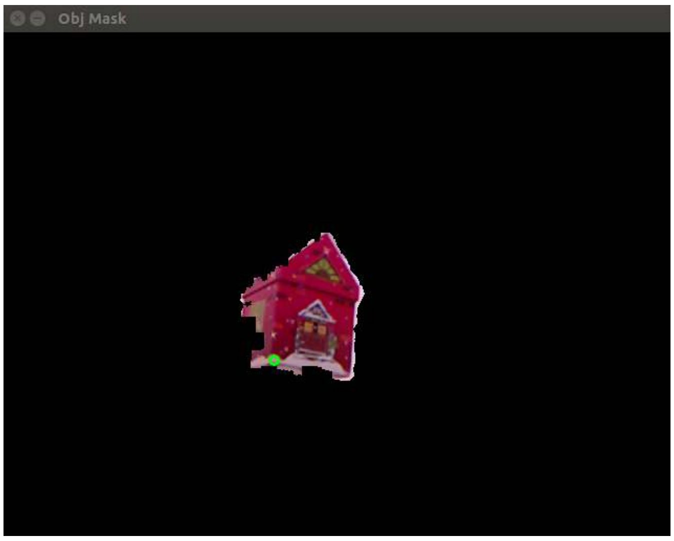


Figure 5.1: Image after applying the euclidean clustering. The green circle defines the start point of the clustering.

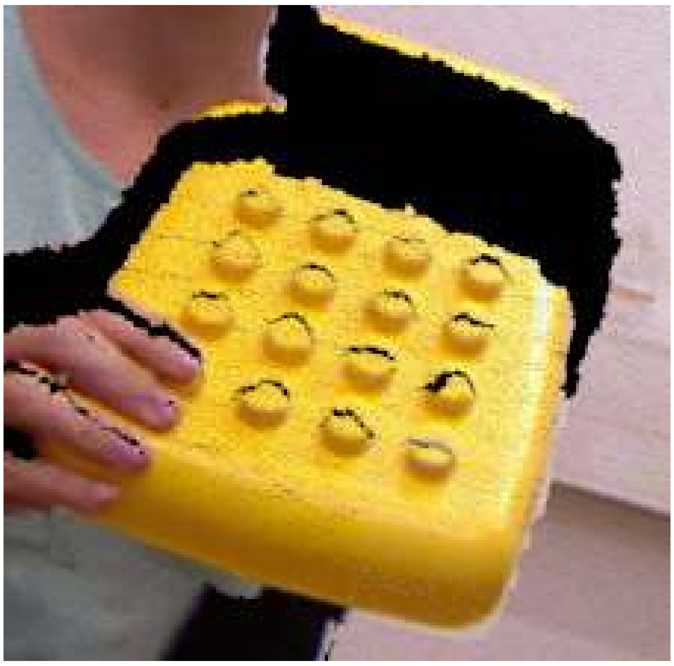


Figure 5.2: Part of a 3D-Image, where we can see that the contour of the yellow object is not accurate.

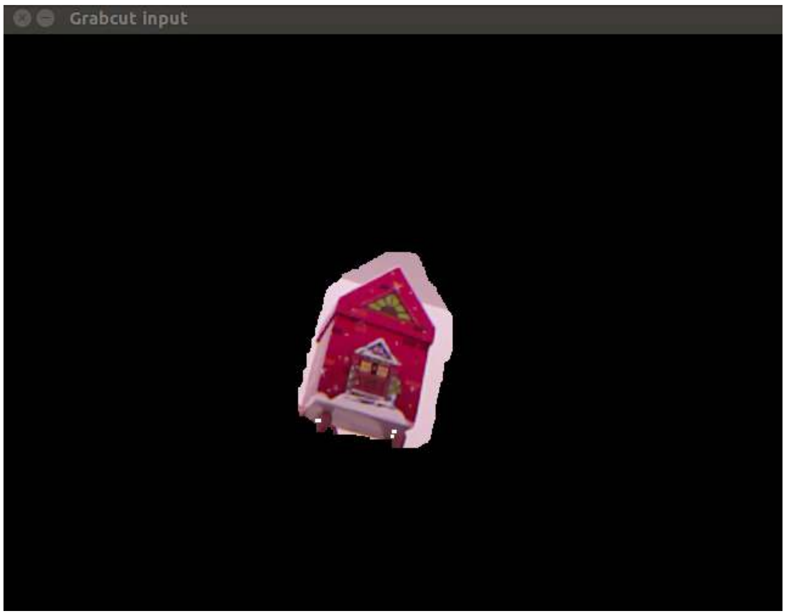


Figure 5.3: Dilated object mask. White pixels are denoted as sure background.

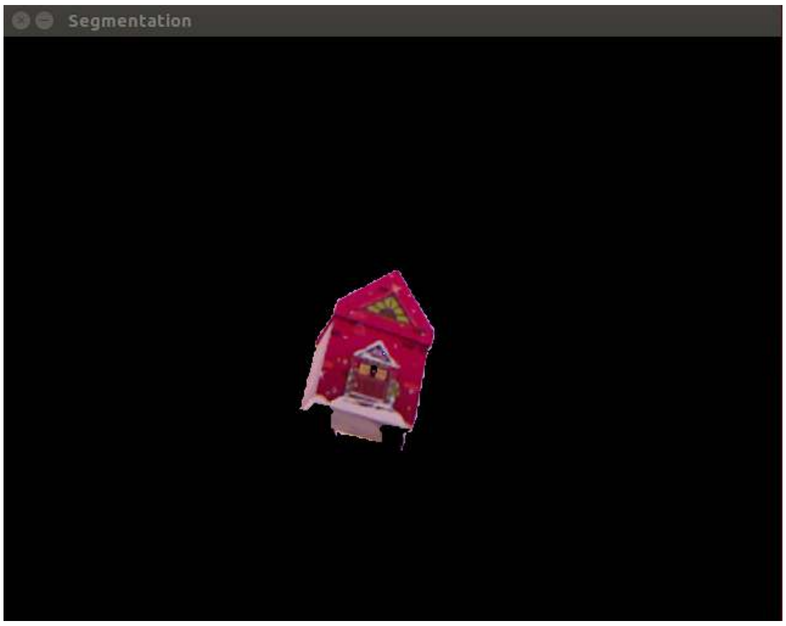


Figure 5.4: Result of grabcut.

CHAPTER 6

# Modelling

After the segmentation we have a 3D image of the object from the current camera view. To get a full 3D model we need several views from different poses, which we can combine to one model. This is done by a method published by J. Prankl, A. Aldoma, A. Svejda, and M. Vincze [PASV15], which is part of the V4R-Library [V4R]. The basic idea of this method is to extract Scale-invariant feature transform (SIFT)-points [Low04] from each view. These interest points are used to map each view to the correct pose in the final 3D model.

For the first view, we use the nearest point as reference point for the euclidean clustering (see figure 5.1). In the further views, we calculate the centroid of the segmented objected, and use the nearest point the centroid as new reference point for the clustering. This allows the user also to move the object, while rotating it in front of the camera. In figure 6.1 the modeling process is shown. The points in the image represents the SIFT points of the object.

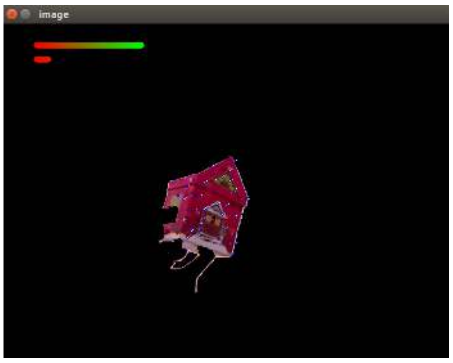


Figure 6.1: Picture of interest points.

CHAPTER 7

# Results

As already mentioned, we have not found any other scientific work for object learning in human hands. Therefore we are not able to compare our method with others. To quantizise our results we introduce our own evaluation method. The learned object will be rated by two different properties:

- Valid points (VP)
- Invalid points (IP)

## 7.1 Valid points (VP)

This property defines how many points of the object are in our final 3D model compared to all points, which would be possible. This means, points which are not seen by the camera (e.g. parts of the object covered by the hand) or not part of the object are not taken into account. The value is calculated as shown in Equation 7.1 and is between 0 and 1. If the value is 1 all points which are seen by the camera are in our 3D model.

$$VP = \frac{Validpoints\ in\ our\ 3D\ model}{All\ possible\ pixels} \tag{7.1}$$

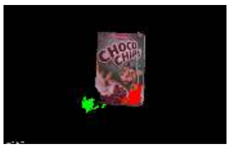
## 7.2 Invalid points (IP)

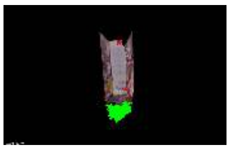
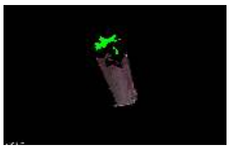
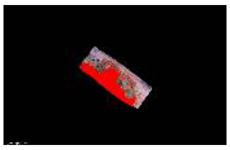
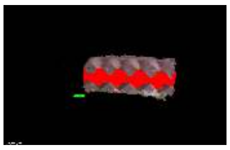
This property defines how many invalid points are in our final 3D model compared to all points in our 3D model. This means, points which are not part of the object (e.g. part of the hand or the background) are defined as invalid points and compared to all points in the 3D model. The value is calculated as shown in Equation 7.2 and is between 0 and 1. If the value is 0 there are no wrong points in our 3D model.

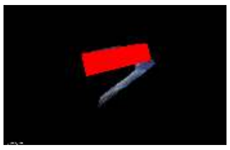
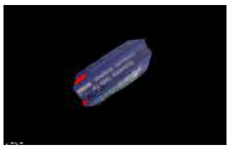
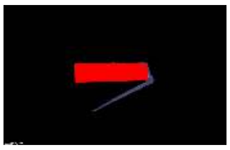
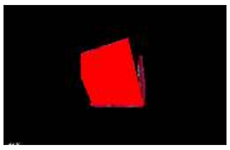
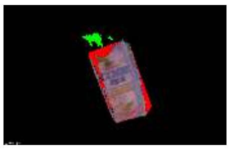
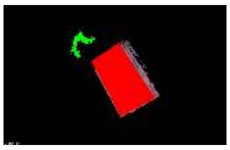
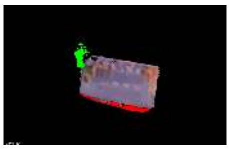
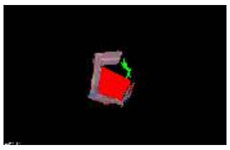
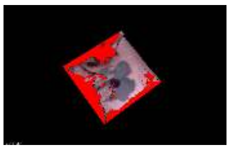
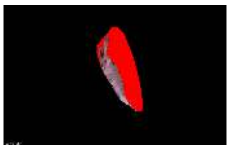
$$IP = \frac{Invalid\ points}{All\ points\ in\ the\ 3Dmodel} \tag{7.2}$$

## 7.3 Evaluation

In table 7.1 we show quantitive results of several objects captured with a Microsoft Kinect RGB-D camera. We ensured sufficient light conditions for the camera and enough space for the manipulation of the object. We also ensured there are no objects between the target object and the camera. For every object we used maximal 3 attempts. Unfortunately to calculate the exact values of our evaluation method, we would need perfect models of our test objects. Also calculating the exact number of possible pixels, would be very difficult. Therefore we estimated the values by taking several specific views of every object. In each view we colored missing parts in red and pixels which are not part of the object, green. Afterwards we calculated the number of valid and invalid points by the corresponding pixels in the view.
A full pixel distribution is shown in figure 7.1. The blue bar represents all pixels in our view, which are part of the object. The green bar shows the number of pixels, which are not part of the object (e.g. hand or background). The brown bar shows, how many pixels are missing in the view. The sum of the blue and brown bar are all possible pixels in the view.

| Object | Cornflakes box | Cornflakes box | Cornflakes box | Cornflakes box |
|---|---|---|---|---|
| View | 1 | 2 | 3 | 4 |
| Image | | | | |
| VP<br>IP | 1.00<br>0 | 0.99<br>0 | 0.99<br>0 | 1.00<br>0 |
| Object | Cornflakes box | Cornflakes box | Cornflakes box #2 | Cornflakes box #2 |
| View | 5 | 6 | 1 | 2 |
| Image | | |  | |
| VP<br>IP | 1.00<br>0 | 0.99<br>0 | 0.95<br>0.04 | 0.98<br>0 |
| Object | Cornflakes box #2 | Cornflakes box #2 | Cornflakes box #2 | Cornflakes box #2 |

| | | | | |
|---|---|---|---|---|
| View | 3 | 4 | 5 | 6 |
| Image | |  | |  |
| VP<br>IP | 1.00<br>0.03 | 0.99<br>0.13 | 1.00<br>0.05 | 0.99<br>0.12 |
| Object | Game box | Game box | Game box | Game box |
| View | 1 | 2 | 3 | 4 |
| Image | |  | | |
| VP<br>IP | 1.00<br>0.04 | 0.75<br>0 | 1.00<br>0.04 | 1.00<br>0.06 |
| Object | Game box | Game box | Egg box | Egg box |
| View | 5 | 6 | 1 | 2 |
| Image | | | | |
| VP<br>IP | 1.00<br>0.04 | 0.96<br>0.00 | 0.99<br>0.01 | 0.97<br>0.00 |
| Object | Egg box | Egg box | Book | Book |
| View | 3 | 4 | 1 | 2 |
| Image |  | | | |
| VP<br>IP | 0.77<br>0.01 | 0.73<br>0.00 | 1.00<br>0.00 | 0.95<br>0.00 |
| Object | Book | Book | Book | Book |
| View | 3 | 4 | 5 | 6 |

| | | | | |
|---|---|---|---|---|
| Image |  |  | |  |
| VP | 0.60 | 0.98 | 0.74 | 0.56 |
| IP | 0.00 | 0.00 | 0.00 | 0.00 |
| Object | Christmas box | Christmas box | Christmas box | Christmas box |
| View | 1 | 2 | 3 | 4 |
| Image | | | | |
| VP | 0.98 | 0.80 | 0.95 | 0.81 |
| IP | 0.00 | 0.00 | 0.00 | 0.00 |
| Object | Christmas box | Christmas box | Tea box | Tea box |
| View | 5 | 6 | 1 | 2 |
| Image |  | |  |  |
| VP | 0.54 | 0.66 | 0.88 | 0.56 |
| IP | 0.00 | 0.00 | 0.05 | 0.06 |
| Object | Tea box | Tea box | Pad | Pad |
| View | 3 | 4 | 1 | 2 |
| Image |  |  |  |  |
| VP | 0.95 | 0.7 | 0.76 | 0.60 |
| IP | 0.05 | 0.04 | 0.00 | 0.00 |
| Object | Pad | Pad | | |
| View | 3 | 4 | | |

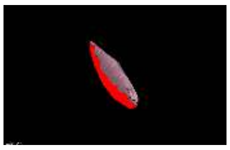
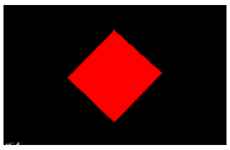
| Image |  |  | | |
|---|---|---|---|---|
| VP<br>IP | 0.75<br>0.00 | 0.51<br>0.00 | | |

Table 7.1: Evaluation of different objects.

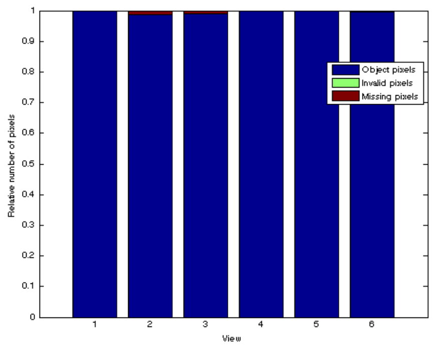
In table 7.2 the overall performance for each object is shown. The values are calculated by the sum of all views for each object. In figure 7.2 the pixel distribution of the overall performance for each object is illustrated.

| Object | Cornflakes box | Cornflakes box 2 | Game box | Egg box | Book | Christmas box | Tea box | Pad |
|---|---|---|---|---|---|---|---|---|
| VP | 0.995 | 0.980 | 0.952 | 0.860 | 0.820 | 0.774 | 0.757 | 0.623 |
| IP | 0.000 | 0.055 | 0.033 | 0.005 | 0.000 | 0.000 | 0.045 | 0.000 |

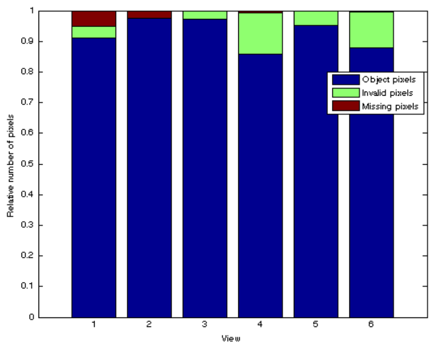
Table 7.2: Overall performance for each object.

Further the qualitative result of our objects are illustrated in 7.3 - 7.10.
In figure 7.3 we modeled a cornflakes box with our approach. It can be seen that our method modeled most parts of the box correctly. Especially these types of object works really good with our method.
In figure 7.4 we modeled a second cornflakes box. In this case we also see some parts of the hand in the 3D model.
In figure 7.5 we modeled a game box. There we can see that some parts of the surface are missing (holes). This happens if some areas of the object are detected as skin.
In figure 7.6 we modeled an egg box with our method. Most parts of the box were learned correctly. We can further see, that there is also a little part of the hand visible in the 3D model. Especially if the skin detection does not detect all pixels of the hand, some artefacts in the 3D model can occur.
In figure 7.7 we modeled a book with our method. It can be seen that the pages of the book are missing. This happens if Grabcut can not differentiate between the background and parts of the object. In this case the pages and the background are white and so Grabcut detected the pages as background. Furthermore it can be seen that it looks like the book would be opened. This is caused by noisy depth information which lead to an angular error. This error is accumulative.
In figure 7.8 we modeled a christmas box. It can be seen that the roof of the box is missing as in the figure before the pages. In this case the roof and the background are white and so Grabcut detected the roof as background. Further can be seen that there is again an angular error, which is caused as in the figure before by noisy depth data.
In figure 7.9 we modeled a tea box. Some parts of the object are missing, because of the size of the object. Most parts were covered by the hand and so not enough interest

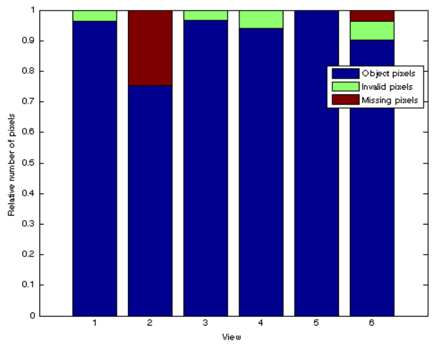
points found. Also parts of the hand are in the 3D model visible.
In figure 7.10 we modeled a pad. It can be seen that most parts of the object are missing. There were not enough interest points available to learn this object.



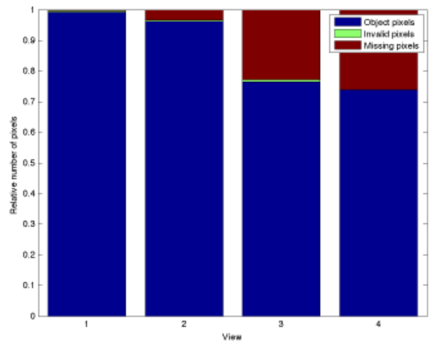
(a) Cornflakes box.



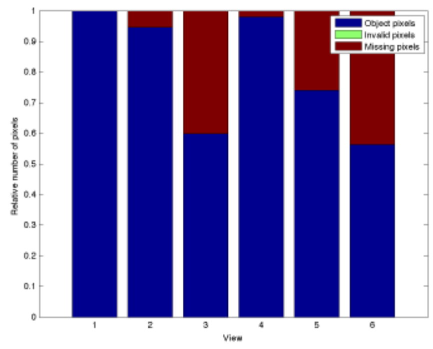
(b) Cornflakes box #2.



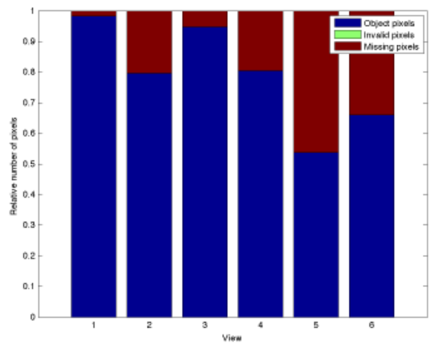
(c) Game box.



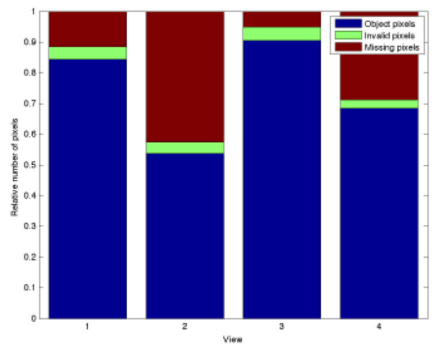
(d) Egg box.



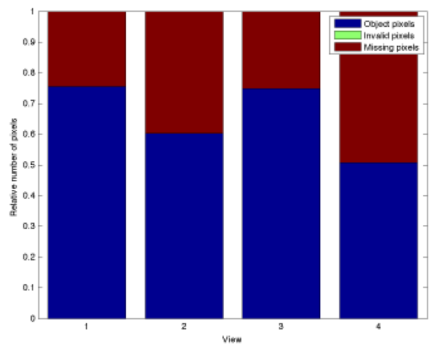
(e) Book.



(f) Christmas box.



(g) Tea box.



(h) Pad.

Figure 7.1: Pixel distribution of all objects for each view.

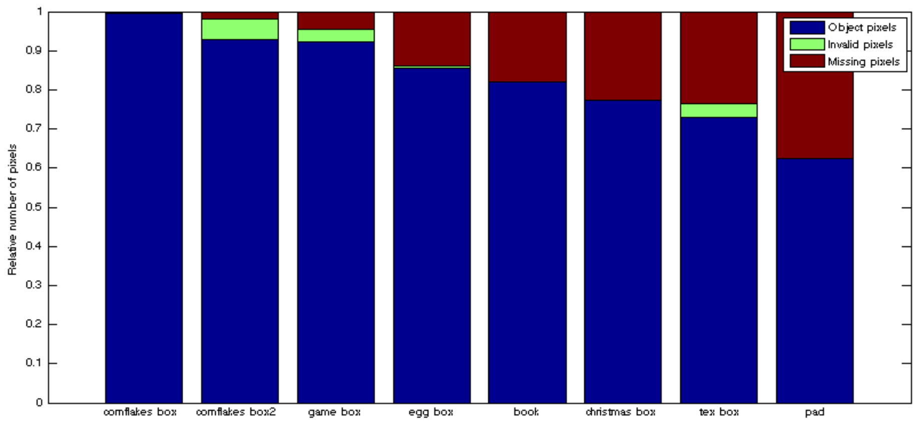


Figure 7.2: Sum of all views for every object.

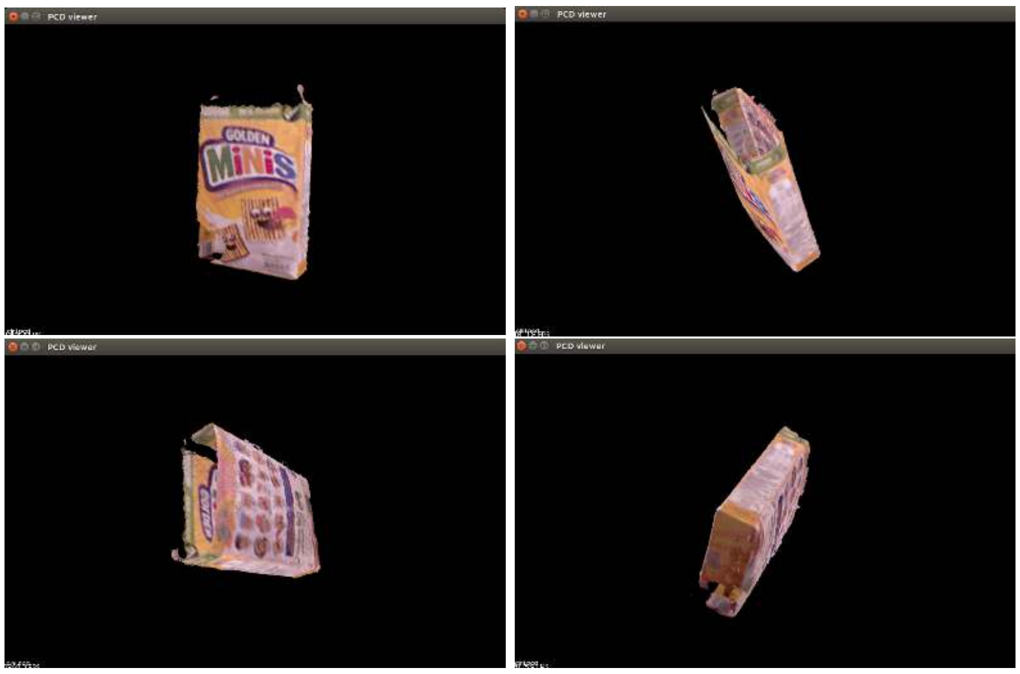


Figure 7.3: 3D model of a cornflakes box, captured with our method.

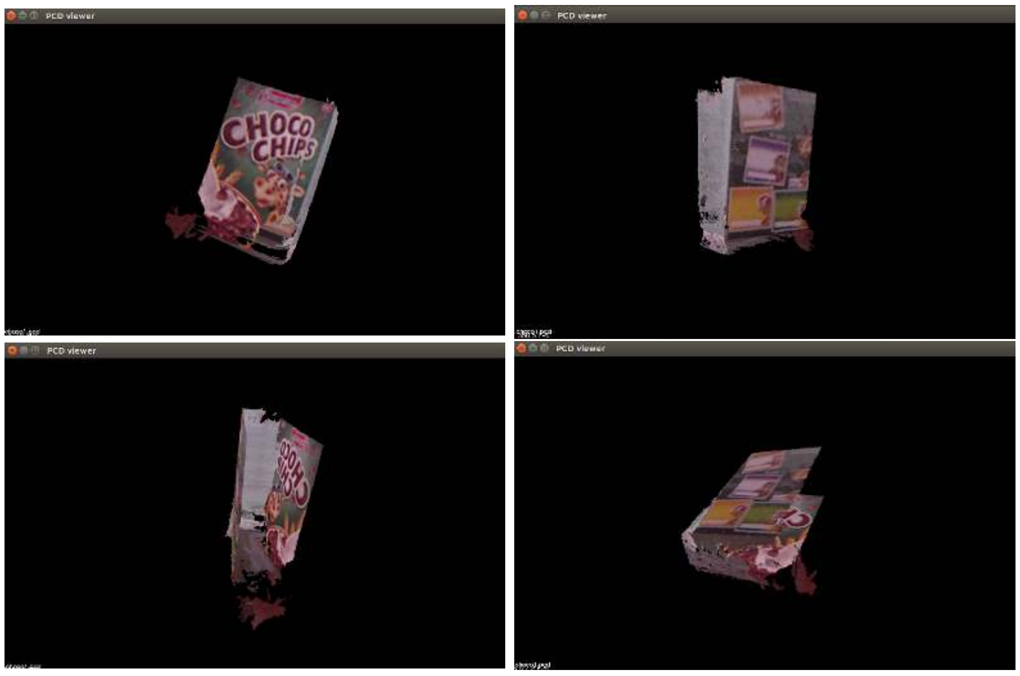


Figure 7.4: 3D model of a second cornflakes box, captured with our method.

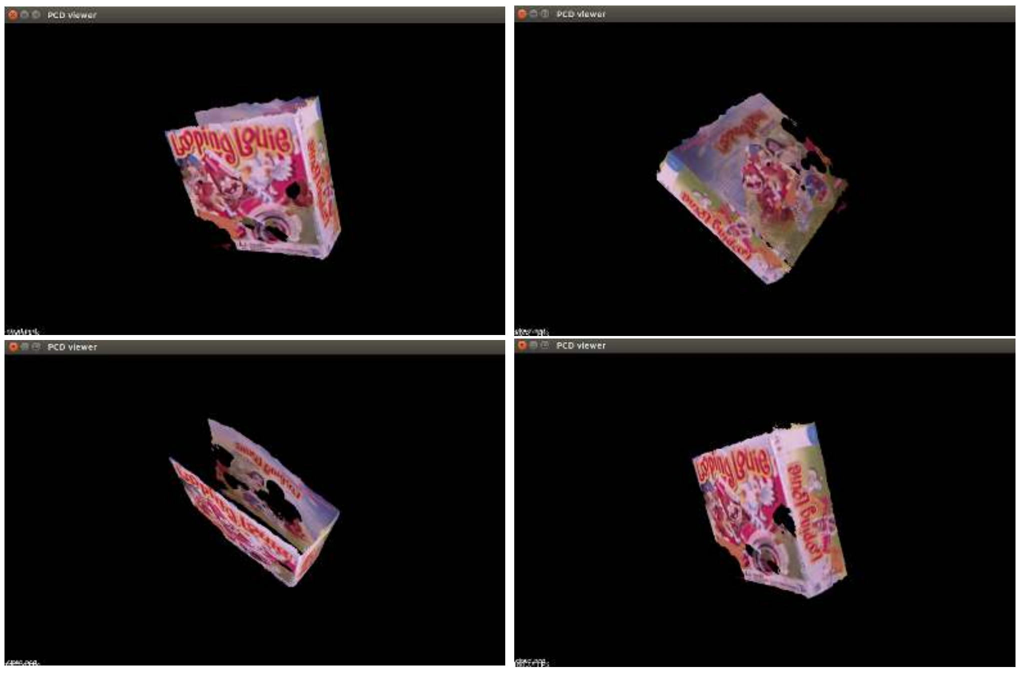


Figure 7.5: 3D model of a game box, captured with our method.

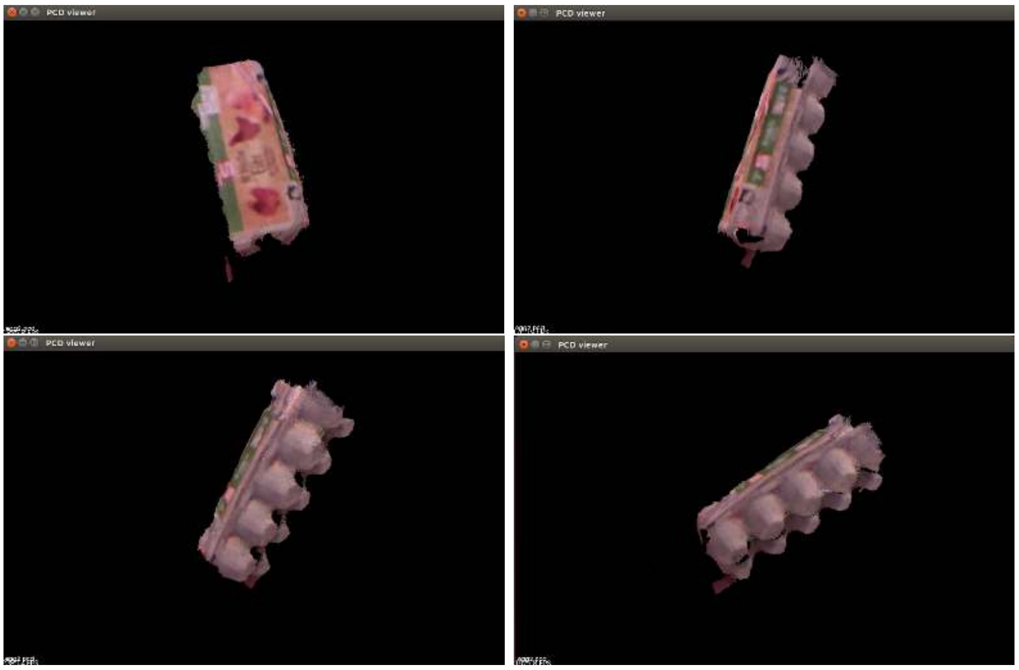


Figure 7.6: 3D model of an egg box, captured with our method.

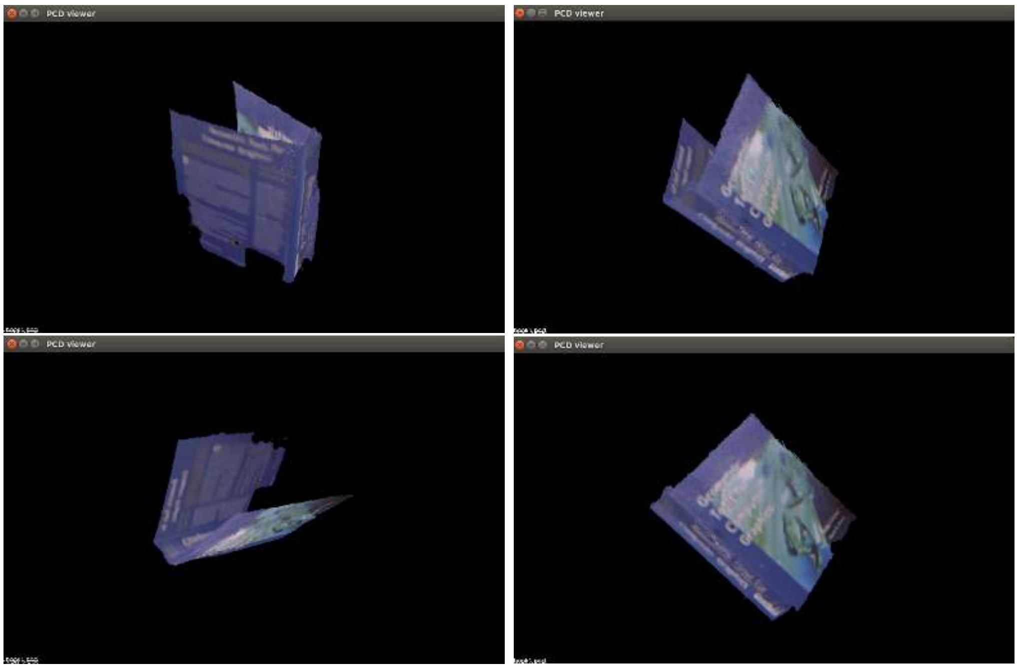


Figure 7.7: 3D model of a book, captured with our method..

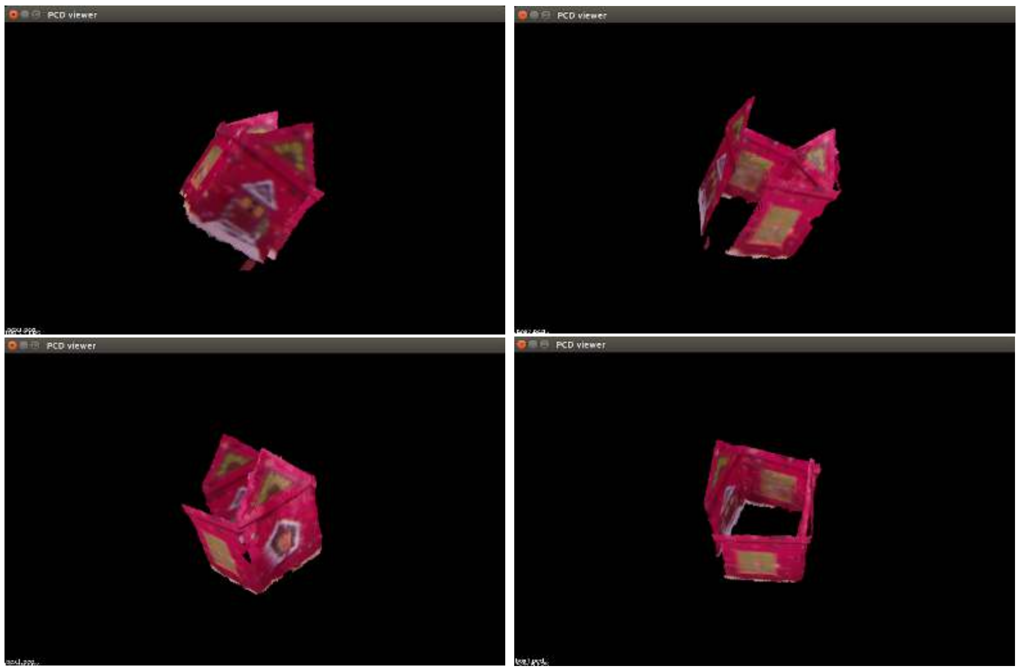


Figure 7.8: 3D model of a christmas box, captured with our method.

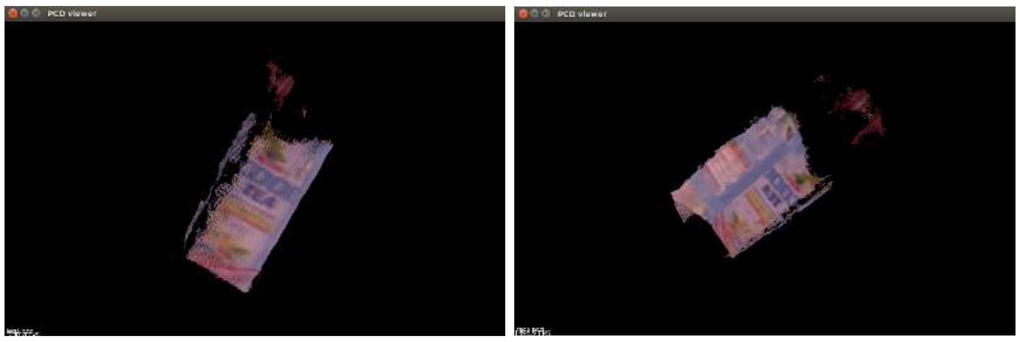


Figure 7.9: 3D model of a tea box, captured with our method.

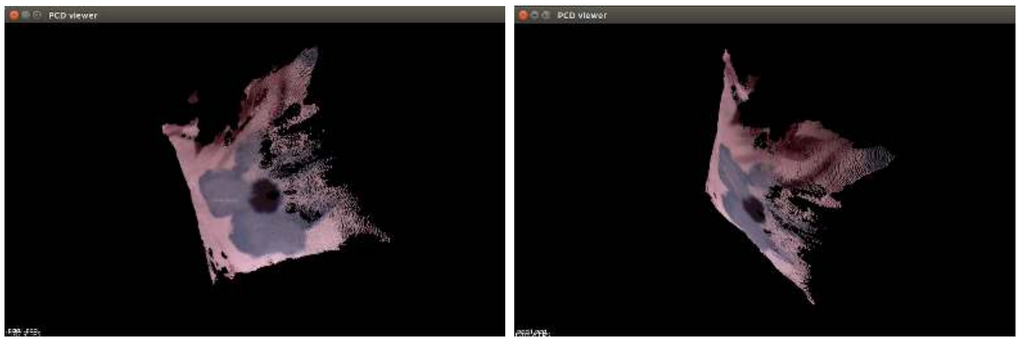


Figure 7.10: 3D model of a pad, captured with our method.

CHAPTER 8

# Limitations

We first look at characteristics of the objects, which produce a good model. But also how the environment and the user impact the quality of the resulting model.

## 8.1 Object requirements

Results in the previous chapter only objects, which have been successfully modeled. We also tried some other objects, which could not be modeled. Therefore we need some requirements, which the object has to fulfill.

### 8.1.1 Size

If the object is too small, not enough SIFT-points are available. Therefore the different views of the object can not be mapped together. Also if we want to model a thin object, we have the same problem. During the rotation, there will be one pose where the thin side of the object faces the camera. At this point maybe no interest points are detected and the modeling process would fail at this point. Another problem, if the object is too small, is that it could not be segmented, because we dilate the detected hand mask. If the object is too small it could be part of the dilation. On the other side, the object can also be too big. It is necessary to hold the object in the hand for the segmentation step and it is also necessary to rotate it in front of the camera to get a 3D model. If this is not possible, our method will fail.

### 8.1.2 Texture

Another important requirement is the existence of SIFT-points. If our method can not find any interest points, it is not possible to map the different views. Especially mostly monochrome and faint objects are very difficult to model, with our method.

### 8.1.3 Reflections

Another problem is the reflection of the surface of the object (e.g. mirror, glass, ...). In this case the RGB-D sensor is not able to get a valid 3D information. Without the 3D information we can not cluster the object. If only few parts or at an oblique angle the object is reflective our method will also succeed. But if the whole object or most of the part are reflective our method will be not able to cluster the object.

## 8.2 Environment

Also the envrionment, which means the light conditions and the user are important to get a accurate 3D-model of the object.

### 8.2.1 Light

In our experiments we ensured a good light condition, which is very important for the skin detection. If there is insufficient light, parts of the background or the object seems to have the same color as the skin and the segmentation could fail.

### 8.2.2 User

Another important limitation is the user. Our skin detection is based on the face. So if the user has a beard or wears glasses, the accuracy of the skin detection is affected. We also tried some experiments with a particular user with and without glasses, which produces a different outcome. Especially without glasses the accuracy of the skin detection was better. Also with a beard the skin detection worked not as good, as without beard. There is also another limitation, if the color of the skin of the user's hand differs too much with the color of the skin of the user's face. Therefore the skin of the hand could not be detected and the segmentation would fail.

### 8.2.3 Camera

In our tests we used the Microsoft Kinect, which is a cheap alternative to professional depth cameras. The Kinect was not designed for such a usecase and so therefore some problems can occur using it. Especially 3D information at near distances are very problematic. In figure 8.1 an original 3D image of the camera is shown. As it can be seen the object in the hand is not fully visible. This is caused by the fact, there is no 3D information available at some areas. In figure 8.2 the back of the same object at the same distance to the camera is shown. There it can be seen, that the object is fully visible. Also if we increase the distance to the camera, shown in figure 8.3 more 3D information is available. But if we increase the distance to the camera, the texture quality of the object is decreased, because the RGB-Image of the Kinect has only a resolution of 640x480 pixels. In figure 8.1 the object has a size of 96x190 pixel. In figure 8.3 the same object has a size of 73x146 pixel. This means in figure 8.3 the object has only 58% of the size as in figure 8.1.

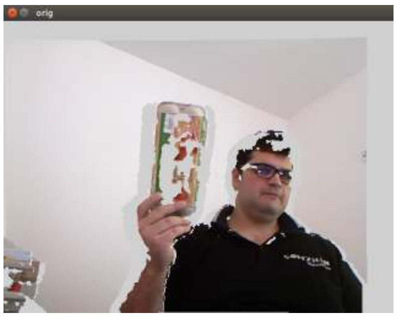


Figure 8.1: Original 3D view of the camera. Grey pixels are denoting areas without 3D information.

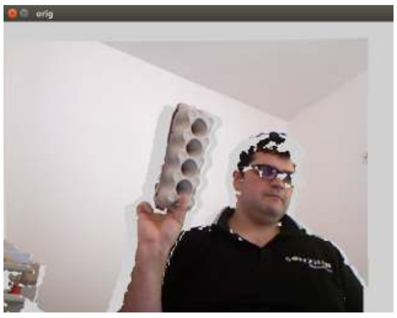


Figure 8.2: Original 3D view of the camera. Backside of the object.

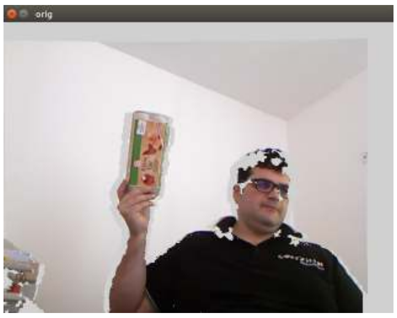


Figure 8.3: Original 3D view of the camera with increased distance from the object to the camera.

CHAPTER 9

# Conclusion and further work

While not solving all problems for all conditions, the presented method allows to obtain satisfactory 3D models for a certain class of objects (sufficiently large and textured). We have shown a working setup, which is a good base for further research on this topic. The process is quite self explanatory and takes no complex feedback for the user. In our case the whole process took about 20-40 seconds for a typical object. In our results some errors in the 3D models can occur (e.g. holes in model, parts of the hand visible,..). Especially the Kinect provides such a bad RGB-image compared with nowadays cameras. Grabcut and V4R would benefit from a better RGB-image. Another improvement would be if Grabcut make use of 3D information. Therefore the algorithm could better distinguish between parts of the object and the background with the same color. Also the hand detection could be improved by combining skin color detection and kinematic models. This could lead to a better hand detection. Afterwards if the hand is detected, the object detection before clustering could be also improved by concluding that the object is in the hand. There is also the problem, that the full 3D model conists of many single 3D views of the object. If at least one view contains part of the hand, it can be seen in the full 3D model. Therefore it would be better to count the occurrence of every pixel and decide at the end if it is part of the model.

# List of Figures

# List of Tables

# List of Algorithms

# Acronyms

| | |
|---|---|
| **CNN** | convolutional neural networks. 7 |
| **HRI** | Human Robot Interaction. vii, ix |
| **ICP** | Iterative Closest Point. 3 |
| **PCL** | Point Cloud Library. 3 |
| **SIFT** | Scale-invariant feature transform. 39, 53 |
| **V4R** | Vision for Robotics. vii, ix |

# Bibliography

[AAD$^+$11] Eren Erdal Aksoy, Alexey Abramov, Johannes Dörr, Kejun Ning, Babette Dellen, and Florentin Wörgötter. Learning the semantics of object–action relations by observation. *The International Journal of Robotics Research*, 30(10):1229–1249, 2011.

[CCH10] Meng-Kai Jiang Chen-Chiung Hsieh, Dung-Hua Liou. Fast enhanced face-based adaptive skin color model, 2010.

[FAB$^+$17] T. Fäulhammer, R. Ambruş, C. Burbridge, M. Zillich, J. Folkesson, N. Hawes, P. Jensfelt, and M. Vincze. Autonomous learning of object models on a mobile robot. *IEEE Robotics and Automation Letters*, 2(1):26–33, Jan 2017.

[Han] Hand tracking and recognition with opencv. `http://simena86.github.io/blog/2013/08/12/hand-tracking-and-recognition-with-opencv/`. Accessed: 2017-06-18.

[KHRF11] Michael Krainin, Peter Henry, Xiaofeng Ren, and Dieter Fox. Manipulator and object tracking for in-hand 3d object modeling. *Int. J. Rob. Res.*, 30(11):1311–1327, September 2011.

[KLK14] Seongyong Koo, Dongheui Lee, and Dong-Soo Kwon. Incremental object learning and robust tracking of multiple objects from rgb-d point set data. *J. Vis. Comun. Image Represent.*, 25(1):108–121, January 2014.

[KMB07] P. Kakumanu, S. Makrogiannis, and N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recogn.*, 40(3):1106–1122, March 2007.

[LN10] Dongheui Lee and Yoshihiko Nakamura. Mimesis model from partial observations for a humanoid robot. *The International Journal of Robotics Research*, 29(1):60–80, 2010.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[LR03] Pisarevsky V. Lienhart R., Kuranov A. Empirical analysis of detection cascades of boosted classifiers for rapid object detection., 2003.

[LzP10] Qiong Liu and Guang zheng Peng. A robust skin color based face detection algorithm. In *2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010)*, volume 2, pages 525–528, March 2010.

[ML] Justus Piater Manuel Lang. Explaining point cloud segments in terms of object models. Accessed: 2018-01-29.

[MZT11] A. Mittal, A. Zisserman, and P. H. S. Torr. Hand detection using multiple proposals. In *British Machine Vision Conference*, 2011.

[NiT] Primesense nite. `https://en.wikipedia.org/wiki/PrimeSense`. Accessed: 2017-08-08.

[NL13] Anh Nguyen and Bac Le. 3d point cloud segmentation: A survey. pages 225–230, 11 2013.

[Opea] Opencv closing. `https://docs.opencv.org/2.4/doc/tutorials/imgproc/opening_closing_hats/opening_closing_hats.html#closing`. Accessed: 2017-07-04.

[Opeb] Opencv ersion and dilation. `https://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html`. Accessed: 2017-07-03.

[PASV15] J. Prankl, A. Aldoma, A. Svejda, and M. Vincze. Rgb-d object modelling for object recognition and tracking. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 96–103, Sept 2015.

[PCL] In-hand scanner for small objects. `http://pointclouds.org/documentation/tutorials/in_hand_scanner.php`. Accessed: 2017-06-11.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.

[RMG97] Yogesh Raja, Stephen J. McKenna, and Shaogang Gong. *Segmentation and tracking using colour mixture models*, pages 607–614. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[SMZ$^+$16] Srinath Sridhar, Franziska Mueller, Michael Zollhoefer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. Real-time joint tracking of a hand manipulating an object from rgb-d input. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.

[Squ] Squirrel project. `http://www.squirrel-project.eu/`. Accessed: 2017-09-13.

[TST$^+$15] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust articulated-icp for real-time hand tracking. *Symposium on Geometry Processing (Computer Graphics Forum)*, 2015.

[V4R] V4r library. `https://www.acin.tuwien.ac.at/vision-for-robotics/software-tools/v4r-library/`. Accessed: 2017-12-04.

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features, 2001.

[VSA03] Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva. A survey on pixel-based skin color detection techniques. In *IN PROC. GRAPHICON-2003*, pages 85–92, 2003.

[YGHS$^+$17] Shanxin Yuan, Guillermo Garcia-Hernando, Bjorn Stenger, Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, Jan Kautz, Sina Honari, Liuhao Ge, Junsong Yuan, Xinghao Chen, Guijin Wang, Fan Yang, Kai Akiyama, Yang Wu, Qingfu Wan, Meysam Madadi, Sergio Escalera, Shile Li, Dongheui Lee, Iason Oikonomidis, Antonis Argyros, and Tae-Kyun Kim. 3d hand pose estimation: From current achievements to future goals, 2017.

[YYGHK17] Shanxin Yuan, Qi Ye, Guillermo Garcia-Hernando, and Tae-Kyun Kim. The 2017 hands in the million challenge on 3d hand pose estimation, 2017.