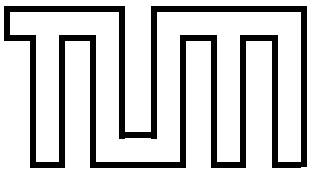


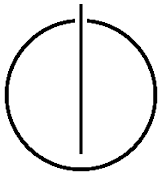
FAKULTÄT FÜR INFORMATIK

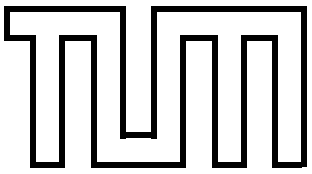
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Informatik

Development of a marker-based visual servo control interface for industrial robots

Alexander Plopski





FAKULTÄT FÜR INFORMATIK

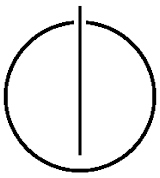
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Informatik

Development of a marker-based visual servo control interface for industrial robots

Entwicklung einer markerbasierten visuellen Kontrollschnittstelle für industrielle Roboter

Author:	Alexander Plopski
Supervisor:	Prof. Dr.-Ing. Alois Knoll
Advisor:	Dipl.-Inf. Thomas Müller
Date:	Mai 15, 2010



Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 3. Mai 2010 Alexander Plopski

Acknowledgments

I'm grateful to Prof. Dr.-Ing. Alois Knoll and the Robotics and Embedded Systems chair from the Department of Informatics of the Technical University of Munich for the opportunity and location to make this bachelor thesis. I would also like to thank Dipl.-Inf. Thomas Müller for his ongoing support during my research. Without his advise and guidance these results could not have been achieved.

Abstract

The space mouse is a common controller for a 6DOF industrial robot. This thesis reviews the applicability of a marker-based visual controller to perform the task of a space mouse. A number of possible approaches for marker detection and transformation calculations are introduced followed by the explanation of the implementation and a review of the stability. The result of this research is capable of replacing the space mouse to control a robot, but a number of critical problems are pointed out as well as possible future improvements to counter these.

Contents

1. Introduction

In the 21st century the production in the modern society is based all around robots. They possess power by far surpassing the human and can complete even the most difficult tasks with unrivaled precision and continue this task close to eternity. Most of them are equipped with a number of sensors, for example visual or torque. They can work in places unreachable or dangerous for humans.

Yet the industrial robots are still not capable of completing every task without being monitored or supervised by humans. Furthermore a number of tasks require the cooperation between humans and robots. For such tasks there is a need for means of cummunication between humans and robots.

But what would be the most efficient way to communicate?
One way that comes to mind instantly is an interface with which the human can control the movements of the robot. This requires an interface, let us say a console, for the supervisor to enter coordinates or directions he wishes the robot to move to. These coordinates or joint positions can be memorized if this task has tot be repeated a number of times. Still this approach is rather inefficient and makes it impossible for the operator to cooperate with the robot.

A second approach can be the implementation of gesture detection to control the robot. But this approach is quite difficult as it requires a gesture for every task. One such gesture could signify a "follow me" order, commanding the robot to track and follow a predefined point. The predefined point would become a marker, a point of interest. Generally, almost any distinct object can become a marker. In retail business the barcode is used as a marker to identify goods, in other applications an IR-emitter marks an object, to mention two examples of possible markers.

In recent developments in the computer game industry two controllers have been introduced to control games, namely the Wii Remote and the Playstation Move. Both can detect rotations and translations, thus it is possible to use them to control a 6DOF robot. But the development of visual controllers, which detect tranformations, does not end with these two controllers. Microsoft recently introduced a visual game controller of its own which is expected to be released this year. In the academic research a number of papers on visual control, such as [20, 25, 13], have been published. With the visual sensors becoming more and more precise in combination with smaller size and lower costs

this research resulted in the field of visual servoing. In general visual servoing sums up all research about robot control based on data extracted from a visual sensor. Currently, visual servoing is subdivided into three fields which result from three different ways of detecting the tracked object: Image based visual servoing (IBVS), Position based visual servoing (PBVS) and the dynamic approaches. While IBVS is mainly 2D-Servoing and PBVS is focused on the object model and therefor a 3D-Servoing approach, the dynamic approaches try to combine both 2D and 3D research.

One research project, which also operates in the field of visual servoing is SFB 453 - "High-Fidelity Telepresence and Teleaction" which aims to analyse and further develop interaction between a human operator and the robotic teleoperator. The goal is to allow the operator to feel present while operating in a distant environment. The subproject "Automated, Robot-Assisted handling of Limp and Deformable Objects" particularily concentrates on recognition, tracking, modelling and prediction of deformable objects.

This thesis observes the applicability of a marker based controller to control a 6DOF industrial robot and introduces the interface required for such a controller. As mentioned previously, there are almost endless possibilities for markers, thus this thesis concentrates on a marker with a distinctive color setup.

Chapter 2 will briefly introduce the existing controllers Wii Remote and Playstation Move as well as the respective approach to detect transformations. In chapter 3 an overview of current approaches to detect and track an object will be shown, and the mathematical background will be explained. Chapter 4 will give an overview of the implementation and is followed by the final evaluation and observations.

2. Existing visual controllers

As mentioned previously there is a number of other existing visual controllers. This section introduces two controllers, that were developed in the game industry but can still be used to control a robot.

2.1. Nintendo Wii Remote

This controller has a built-in ADKL330 acceleration device to sense its movement along the axis's as well as the gravitat. Furthermore it has a PixArt optical sensor to determine the direction the controller is pointing. To use the optical sensor Nintendo makes use of a Sensor Bar. The controller and the bar are shown in Figure 2.1.

The Sensor Bar is equipped with two sets of five LED's each which emit infrared light and is of a fixed size of 20 cm. The controller perceives the light from these LEDs as two bright dots. As the distance between the LEDs on the Sensor Bar and the perceived distance are known the distance between the controller and the Sensor Bar can be calculated by applying triangulation. Furthermore the angle of the detected points is used to calculate the rotation along the Z-axis. The data perceived from the acceleration sensor supports the calculation of all three rotations.

Figure 2.1.: Wii Remote and its Sensor Bar.[1, 2]

Figure 2.2.: The Playstation Move controller.[3]

2.2. Playstation Move

The Playstation controller is shown in Figure 2.2. This controller equips a big ball with light-emitting LEDs on the top. The color of the ball can be assigned individually. To detect this light and thus determine the position Sony uses its own camera, PlayStation Eye. This camera takes 60 frames per second to allow rapid motion detection. Since the size of the original ball is known beforehand the size of the detected ball infers the distance to the controller.

As one can expect, it is not possible to determine any angle using only visual recognition. Therefore this controller equips a linear accelerometer and a three-axis angular rate sensor to track rotations.

3. Object recognition

3.1. Setup of an object recognition system

The brute force approach is to detect an object in the data received from the sensors by matching the model to every possible position. In a picture each pixel is checked wether the model is located at it's position. Of course, this approach is by far inferior to the currently used approaches since it is not invariant to any rotation nor to any zooming or changes in the environment, such as light. Furthermore the runtime of such an algorithm would be very long and therefore unacceptable.

To create a system which is capable of effectively recognizing objects the system of information levels is used. Each level contains an amount of data with a certain level of information. This data can be refined thus a higher level of information is obtained. This is repeated again and again starting at the bottom level with all the data gathered by available sensors, visual, torque and so on, and ending in the highest level of information where the object can be determined.

It generally takes five steps to determine an object. In the first step the data provided by the sensors is recycled to reduce environmental noise and minimize the erroneous sensor data.

In the second step model attributes in the data received are discovered and extracted. Among others such attributes may be color, geometrical forms or firmness. Next, these attributes are transferred into the attributes-space (or the properties-space). That space contains only the extracted attributes.

Finally, after extracting all the necessary attributes into the attributes-space, these attributes are compared to the models may they be it one or multiple. If multiple models are used, these should be stored in a data-base beforehand. This comparison is called matching.

After recognizing and matching the objects to the models their positions can be output.

The explained setup of an object recognition system is not universal. If the experiment deems it necessary a completely different setup can be used, but most systems, as well as the setup in this paper, are similar to this approach. While modeling an object recognition system one must consider available knowledge. Are there multiple models? Are there changes in the environ-

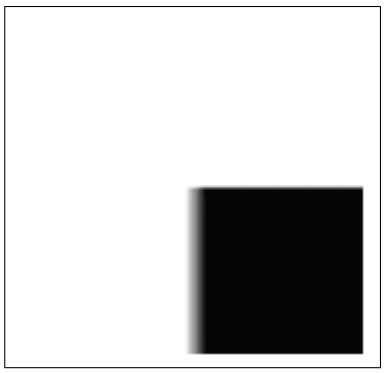


Figure 3.1.: A strong horizontal edge and a weak vertical edge.[16]

ment which require attention? What sensors will be used as well as what sensors are favorable? Of course further questions should be asked while creating such a system. In the following chapters a number of algorithms for attribute extraction (Step 2), which is divided in object based and cognition based, and matching (Step 4), which is separated into model based and vision based, will be introduced.

3.2. Objectbased attribute extraction

3.2.1. Canny edge detectcion

To detect an object in a picture detection of its edges, thus detecting and identifying its geometrical form, can be used. The edge detection algorithm introduced in this section is the Canny edge detection.

For each point in the picture the function f(x,y) represents its gray value. The differentiation of this function is $g(x, y) = (f_x, f_y)$ which are the partial differentiation of f(x,y) towards x and y. This gradient represents the degree of the slope and its direction. Translating this into the picture, the gradient displays the direction and strength of the edge at the point x,y. Two possible edges are shown in Figure 3.2.1.

To calculate g(x,y) the Canny algorithm uses the Sobel method. Hereby the convolutions 3.1 and 3.2 are used.

$$f_x = \begin{pmatrix} p(x-1,y-1) & p(x,y-1) & p(x+1,y-1) \\ p(x-1,y) & p(x,y) & p(x+1,y) \\ p(x-1,y+1) & p(x,y+1) & p(x+1,y+1) \end{pmatrix} * \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad (3.1)$$

$$f_y = \begin{pmatrix} p(x-1,y-1) & p(x,y-1) & p(x+1,y-1) \\ p(x-1,y) & p(x,y) & p(x+1,y) \\ p(x-1,y+1) & p(x,y+1) & p(x+1,y+1) \end{pmatrix} * \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \quad (3.2)$$

Now the strength of the edge $|s(x,y)| = \sqrt{f_x^2(x,y) + f_y^2(x,y)}$ and the edge direction $d(x,y) = \arctan \frac{f_x}{f_y}$ can be calculated.

This detection is further improved by

- smoothing the image before applying the edge detection → the interferences are reduced and the edge detecion gives a better result. The Gaussian filter can be used here.
- edge reduction → In the explained gradient based detection the results for image points next to an edge will often have a very high value. Edge reduction can be used to improve the quality of the representation of the detected edges.
- binarizing the detected edges → this removes theedges with the least strength, thus only the edges of similar strength remain and the amount of bogus edges is reduced.

These four steps, smoothing, edge detection, edge reduction and binarization compose the Canny algorithm.

3.2.2. Hough transformation

The Hough transformation is based on the edge detection which was explained in respect to the Canny algorithm in the previous chapter. This transformation looks for forms made by the edges to determine the figures of interest. These forms can be as plain as a line, a square or a circle. The general Hough transformation can even detect any abstract form of interest.

To find the forms of interest, the Hough transformation requires the x and y coordinates of the detected edges. Further parameters can be the radius of the circle (minimal, maximal) or other necessary parameters. Therefore the detection of lines requires 2 parameters, of circles 3 and of eclipses 5 parameters.

After the parameters have been fixed a search on all points in the space of interest, all pixels of the detected edges, is performed.

The further approach shall be explained with help of the approach to detect lines and the approach to detect circles.

To detect a line the general formula y=mx+n cannot be used, since m and n could become unreasonably huge and are not fixed values. Therefore the hessian normal form is applied. The distance between the line and the base (the

Figure 3.2.: A voting matrix.[16]

center of our space), d, and the angle between the x-axis and the perpendicular of the line, α, and the coordinates of the pixel (i,j) are the parameters for the equation (3.3).

$$d = i\cos(\alpha) + j\sin(\alpha). \tag{3.3}$$

The matrix for the lines would be (d,α) and for each possible i,j and α. For each result value of (d,α) in the matrix would increase. Since a point can be a member of an infinite number of lines the resulting matrix would display a sinus-curve for each point of interest. The detected maxima are the targeted results. The created matrix is called a "Voting-matrix".

A result of such a search can be seen in Figure 3.2.2.

The process to determine circles is similar. The center of the circles is determined with the (3.4) thus a (i,j)-Matrix is created.

$$r^2 = x^2 + y^2 \tag{3.4}$$

This approach enables to detect a number of circles, for example of coins of different size in a picture. Therefore the final matrix would be (x,y,r), of three dimensions.

It should be evident that the dimension of the matrix increases in correspondence to the number of parameters and leads to a great amount of calculations. To counter this a number of improvements have been developed, such as The Fast Hough Transformation.

3.2.3. Contour extraction

An algorithm to detect the contours of objects in a binarized image, an image that consists of 1-Pixels, pixels with density 1, and 0-pixels, pixels with density 0, was proposed by S. Suzuki and K. Abe in [24]. This algorithm derives a sequence of coordinates or the chain codes from the border of an 1-component, a set of 1-Pixels, and a connected 0 component, a set of 0-Pixels. Without losing the case of generality it assumes that the 0-components describe the background or a hole. This algorithm does not explore a way to follow borders, as there was a number of papers dealing with this area, such as [19]. Suzuki and Abe provided the border following with a topological analysis capability. For such a topological analysis the borders are divided into two types, the outer borders and hole borders. Due to one-to-one correspondence of an outer border and a 1-component, as well as a hole border and a 0-component the topological structure of the picture can be determined. The topological analysis is enabled through a unique mark on each detected border and a procedure to determine the parent of the currently followed border.

General declarations

Without loss of generality let the uppermost row, the lowermost row, the leftmost column and the rightmost column, which compose the frame of the picture, be filled with a 0-component. Such a 0-component that contains the frame is called the background. Furthermore a pixel located in row i and column j will be denoted as (i,j) and $F = \{_ij\}$ denotes the density $_ij$ at a pixel (i,j).

For further explanations the following definitions will be used:

Definition 1. A 1-pixel $(i, j) \in S_1$ having a 0-pixel $(p, q) \in S_2$ in its neighborhood is called a border point between the 1-component S_1 and the 0-component S_2.

Definition 2. For two given connected components S_1 and S_2, if there exists a pixel belonging in S_2 for any 4-path from a pixel in S_1 we say that S_2 surrounds S_1. If S_2 surrounds S_1 and there exists a border point between them then S_2 surrounds S_1 directly.

Definition 3. An outer border is defined as the set of the border points between an arbitrary 1-component and the 0-component which surrounds it directly. The border of an 0-component that is surrounded by an 1-component is called a hole border. The hole border as well as the outer border are referred to by ”border”. Furthermore both ”borders” consist of 1-pixels.

As these definitions state for an arbitrary 1-component its outer border is unique. For a 0-component its hole border is unique as well.

Definition 4. The parent border of an outer border between a 1-component S_1 and a 0-cmponent S_2 which surrounds S_1 directly is defined as:

1. the hole border between S_2 and the 1-component which surrounds S_2 directly if S_2 is a hole;
2. the frame of the picture if S_2 is the background.

The parent border of a hole border between a hole S_3 and the 1-component S_4 which surrounds S_3 directly is defined as the outer border between S_4 and the 0-component which surrounds S_4 directly.

Definition 5. For two given borders B_0 and B_n we say that B_n surrounds B_0 if there exists a sequence of borders $B_0, B_1, \ldots, B_n$ such that B_k is the parent border of B_{k-1} for all k($1 \leq k \leq n$).

The definitions 2 to 5 lead to the following isomophoric mapping

- a 1-component $\leftrightarrow$ its outer border;
- a hole $\leftrightarrow$ its hole border between the hole and the 1-component surrounding it directly;
- the background $\leftrightarrow$ the frame.

Suzuki and Abe presented two algorithms in their research. The first one marked all borders and retrieved their topological structure, while the second algorithm retrieved only the outermost outer borders.

First algorithm

This algorithm scans the picture for a pixel (i,j) which satisfies the conditions for a hole or an outer border. For an outer border the previously scanned pixel must be a 0-pixel, while the currently scanned pixel is a 1-pixel. For a hole border the point next to be scanned must be a 0-pixel while the value of the current pixel is ≥ 1. If a pixel satisfies both conditions it is seen as the starting point of an outer border. The newly found border is assigned a serial number, which is denoted by NBD.

To determine the parent of the currently discovered border the serial number of the most recently encountered border, LNBD, is saved. This border is either

the parent of the newly discovered border or has a common parent with it. If the newly discovered border and the most recently encountered border are of the same type, both are hole borders or outer borders, then they share a common parent. In the other case the previously encountered border is the parent of the discovered border. When a new border is discovered a border following algorithm, like [19, 28] or a modification of the Canny-algorithm, is applied and each member of this border is marked with the serial number NBD according to following rules:

1. If the current following border is between the 1-component which contains the pixel (p,q) and the 0-component which contains the pixel (p,q+1), change the value of the pixel (p,q) to -NBD.

2. Otherwise, set the value of the pixel (p,q) to NBD unless (p,q) is on an already followed border.

The conditions listed above prohibit the pixel (p,q) to become a starting point for a new border if it was already assigned to a border previously. After marking all members of a border the algorithm continues the search for a new border until all pixels were scanned.

Second algorithm

The second algorithm is a slight modification of the explained algorithm to detect only the outermost borders. The index of the border LNBD must always be less or equal to 0 and is reset to zero at the start of each new row of the picture. Furthermore the values of ”NBD” and ”-NBD” are substituted with ”2” and ”-2” respectively.

It is easy to see that only the outer borders are detected. Assuming that a row is scanned starting at (i,1) and the pixel (i,j) is a border point where the pixel (i,j-1) is a 0-pixel two possibilities exist where (i,j) is the starting point of an outer border . In the first case all pixels (i,1),(i,2),...,(i,j-1) are 0-pixel. If this is not the case then a border point (i,h) which was encountered most recently must exist. If (i,h) belongs to an outer border where (i,h+1) belongs to the background then (i,j) belongs to a new outer border.

To check this, the index of LNBD is important. Since its index is either 0, 2 or -2 the cases of LNBD=0 and LNBD=-2 satisfy these requirements.

3.2.4. Snakes

The last object based approach mentioned here are the snakes, also known as deformable contours, active contours or minimum energy contours. As these

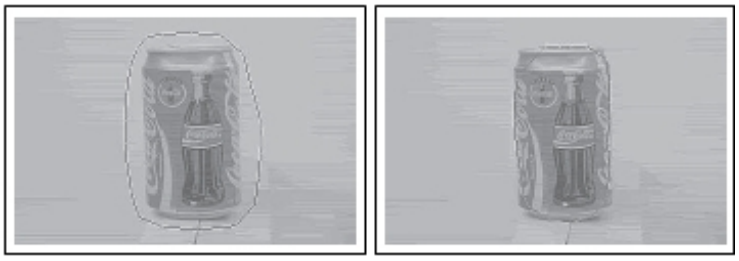


Figure 3.3.: An initial contour left and the resulting contour right.[16]

names suggest snakes are used to determine the contours of an object beginning from an initial contour under use of a so-called energy function. Such a result is shown in Figure 3.3. As it was mentioned before the snake algorithm requires an initial contour to run. If this contour is determined inefficiently, thus allowing further objects or lines between the initialization and the target, this approach may deem useless.

In general the snake algorithm calculates the gradient of the contrast to detect an edge along the contour. By ”pulling” the contour towards the highest contrast a local extreme, a point where the highest contrast is detected, can be reached. At this point the Hildreth-Marr edge, the final destination, is located. Further explanations on snakes and multiple approaches can be found in [14, 11, 27].

3.3. Cognition based attribute extraction

In the previous chapter all approaches concentrated on extracting geometrical shapes to determine the object. But this approach may have a number of disadvantages, because valuable visual information like the color or reflection are not taken into account. Furthermore a model is required beforehand, which may prove to be a hard task. The approaches presented in this chapter will focus on determining objects based solely on visual attributes. The object is seen as a whole which results in avoiding problems such as segmentation errors, loss of information due to limited geometrical attributes and of course we do not need to prepare a model.

3.3.1. Colorsystems

Since the cognition based attribute extraction concentrates on recognition of color parameters and attributes of the object, there is a number of different color systems that can be used here.

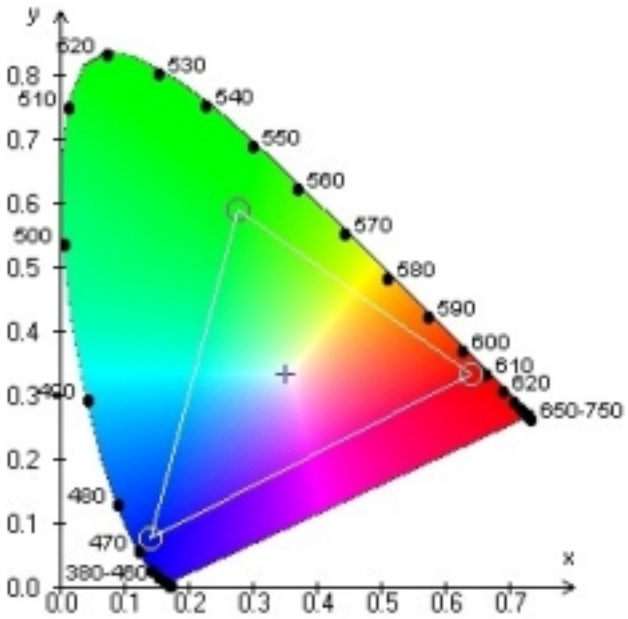


Figure 3.4.: The CIE XYZ System.[4]

The RGB system is the main color system, as perceived by the humans, cameras as well as used in displays and televisions. This system sees the color perception as a combination of red, green and blue. There is a number of diversities for this color space, such as the sRGB and the ECI-RGB systems.

Printers use the CMYK system. This system adds cyan, magenta, yellow and the key color black to create the endresult. This system is not fixed, just like the RGB system, it depends on the medium it is used by. Subsystems are for example the ISOcoated and ISOuncoated.

There is a number of further color systems which are specifications of the RGB system and prove benefitial for different purposes. Two such spaces will be introduced at this point.

The first space is the CIE XYZ color space, shown in Figure 3.4, which was set up by the Commission Internationale de l'Eclairage in 1931. In the 1920s W. David Wright and Hohn Guild independently conducted a series of experiments on the human sight which were the foundation for the CIE XYZ color space. The transformation from RGB is displayed in (3.5).

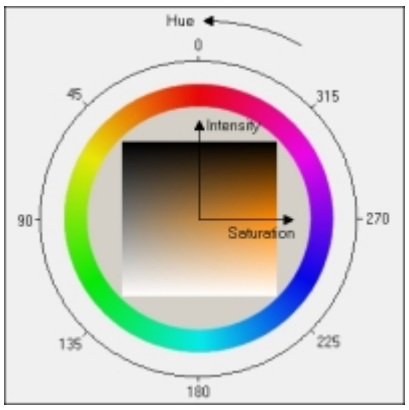


Figure 3.5.: The HSI System.[5]

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \frac{1}{0.17697} \begin{pmatrix} 0.49 & 0.31 & 0.2 \\ 0.17697 & 0.81240 & 0.1063 \\ 0 & 0.01 & 0.99 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{3.5}$$

The final space introduced is the HSI space which is shown in 3.5. HSI is an abbreviation of hue, saturation and intensity. This color space was developed in the 1970s for computer graphics applications. It is a cylindrical geometry where the hue is displayed in 360° around the intensitiy axis and is a set of pure colors. The distance to the center is the saturation. Any change in saturation or intensity does not change the identified color. The reason to use HSI is that modern monitors and televisions can reproduce a wide variety of colors witht he RGB system, but these combinations are inintuitive and the proportions in which R, G and B are used to change saturation or intensity change in such a drastic way, that an easier and less arbitrary way to identify or change colors was necessary. Thus in the mid 1970s PARC and NYIT devised the Hue-Saturation-Value model, which was formally described in [23]. At the same time the HSL system, from which the HSI system was later derived, was devised in [15]. Shortly after first graphic terminals which used these color systems were introduced.

In the HSI color system

- **Hue** is the ``attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors: red, yellow, green, and blue, or to a combination of two of them´´
- **Saturation** is the ``colorfulness of a stimulus relative to its own brightness´´

- **Intensity** is the total amount of light passing through a particular area.

The calculation from RGB into the HSI system is as follows:
If a color is a combination of R,G and B then let

$$\begin{aligned} M &= max(R,G,B) \\ m &= min(R,G,B) \\ C &= M - m \end{aligned} \tag{3.6}$$

and

$$H = 60° \cdot \begin{cases} undefined, & \text{if } C = 0 \\ \frac{G-B}{C} mod 6, & \text{if } M = R \\ \frac{B-R}{C} + 2, & \text{if } M = G \\ \frac{R-G}{C} + 4, & \text{if } M = B. \end{cases} \tag{3.7}$$

Intensity and saturation are defined as

$$\begin{aligned} I &= \frac{R+G+B}{3} \\ S &= \begin{cases} 0, & \text{if } C = 0 \\ 1 - \frac{m}{I}, & \text{otherwise} \end{cases}. \end{aligned} \tag{3.8}$$

Other color systems exist, such as the Hue-Saturation-Lightness System (HSL) or the CIE L*a*b*-color system.

3.3.2. Histogramms

The principle of histogramms is to detect similarities between two pictures to detect objects. Since the histogramms can be quite big and not all colors are of interes,t K-means can be used to group colors thus reducing the size of the histogramm. As a sideeffect such a clustered histogramm is more robust to changes in light. Before histogramm intersection can be used, a number of histogramms of the object have to be prepared beforehand. These should be based on different angles thus allowing determination of an object even in the case of a rotation. Approximately six histogramms are enough to create a stable base.

These histogramms are based on
$\forall i \in [0, x_{max}], j \in [0, y_{max}] : M(i,j) = (r,g,b)^t \Rightarrow H_M(r,g,b) = H_M(r,g,b) + 1.$

The comparison of the model histogramm, H_M, and the picture, H_I compares the grade of coocurrence with equation 3.9

$$V(H_I, H_M) = \frac{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} min(H_I(i,j,k), H_M(i,j,k))}{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} H_M(i,j,k)} \tag{3.9}$$

The positive effect of this approach is that $V(H_I, H_M)$ changes only very little if the target is partially hidden and due to the beforehand prepared histogramms the object can be located even if it was rotated, thus a rotation invariant result is obtained.

Still the drawbacks must not be overlooked. First of all this approach is vulnerable to change in illumination. A change in illumination will often lead to a change in the detected colors thus changing the attribute to detect. Furthermore this approach is not suitable to single-colored or plain-colored objects. Histogramms have a hard time detecting these objects since the histogramm is not specific, especially if the color of the object is detected in the background as well. This can lead to false detection. Last but not least the form of the object is disregarded. Objects of different sizes and forms may lead to similar or equivalent histogramms which is an undesireable result.

To counter these drawbacks an enhanced approach was developed: the Color Coocurrence Histogramms.

3.3.3. Color Cooccurence Histogramms

The Color Cooccurence Histogramms were introduced in [6] and aim to counter the problems mentioned beforehand by taking into account one geometric property. The pixel distance of the object is saved into the histogram. Therefore the Color Cooccurence Histogramm saves the combination of pixels with the same color and the same distance.

In Figure 3.6 pixel $p_1 = (R_1, G_1, B_1)$ and $p_2 = (R_2, G_2, B_2)$ which would be saved in the Histogramm with the distance $(\Delta x, \Delta y)$ are shown. This can be saved as $CH(p_1, p_2, \Delta x, \Delta y)$. Of course if $(\Delta x, \Delta y) = (0, 0)$, so $c_1 = c_2$, then the CH is equal to the normal histogramm. Furthermore the colors in the CCH are clustered for the same reasons as in the normal histogramm, namely to reduce the size, calculations required and make it more robust towards illumination. In order to ignore rotations this approach keeps track of the magnitude $d = \sqrt{(\Delta x)^2 + (\Delta y)^2}$ but not of the direction of Δx or Δy. The quantization of colors and distance is represented as CH(i,j,k), where i and j are two colors and k is the distance range.

To detect the object, a rectangular window is run through the picture and creates a CH for each position. A histogram similar to that of the model is looked for. As a remark, the search window and the model window from which the CH is computed do not have to be of the same size. In fact the search win-

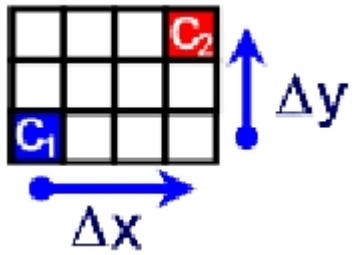


Figure 3.6.: Distance of two points.[6]

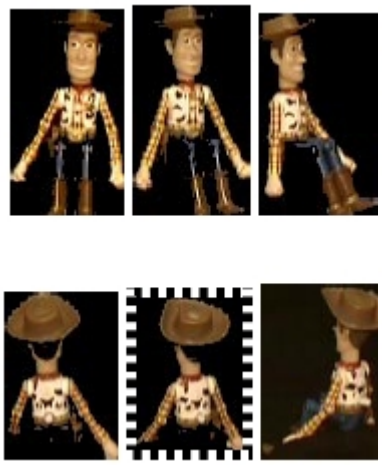


Figure 3.7.: The images taken of Woody.[6]

Figure 3.8.: The found Woody is marked.[6]

dow can be chosen as big as deemed reasonable. The histogramm intersection method can be used to compare the histogramms. If the result exceeds a preset threshold T the object has been discovered.

A search of the doll Woody, from the movie Toy Story, made with 12 histogramms, 8 colors and 12 distances can be seen in Figure 3.3.3.

The CCH is invariant to rotations, partial overlapping and is rather robust to changes in size but it requires a huge amount of calculations, as the search has to be done for each model histogramm prepared beforehand. Similar to normal histogramm detection the CCH encounters problems of detecting simple-colored objects, especially if the color is included in the background, is still a

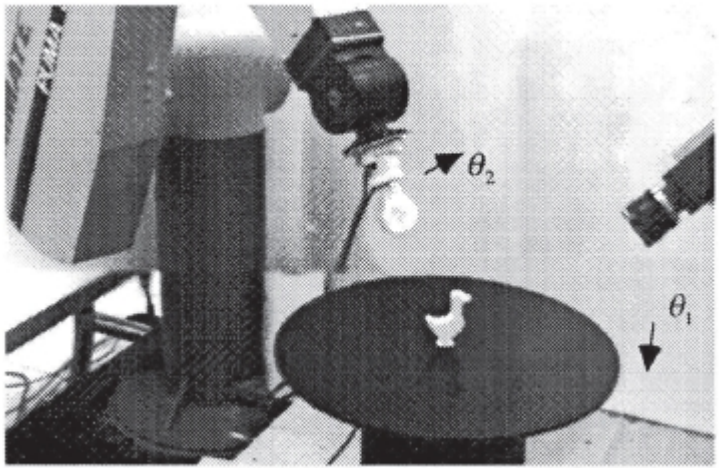


Figure 3.9.: An object is scanned beforehand.[18]

big problem. The CCH counters the problem of objects with different forms but same histogramms by adding one further property, still there exist a number of objects which would result in a similar CH.

3.4. Matching

Each method to detect an object has its own matching methods since they all extract different attributes, like edges for the canny detection and hugh transformation or colors for the histogramms. Two approaches can be differentiated: the model based matching and the vision based matching. For both approaches we can use the direct approach or train a neuronal network.

3.4.1. Vision based matching

In the vision based matching methods the extracted attributes result in a multidimensional vector. As preparation for this approach a number of testresults for the objects we wish to detect must be prepared. Each testresult will lead to a vector in the multidimensinal space and as a whole they create a scatterplot. Of course the scatterplot is different for each object.

To make these scatterplots as accurate as possible the testdata should be made depending on different degrees of freedom. In Figure 3.9 a space which depends on illumination and orientation of the object is created.

The attributes extracted from the picture result as well in a vector in the same space. The scatterplot closest to this vector determines the object detected. In

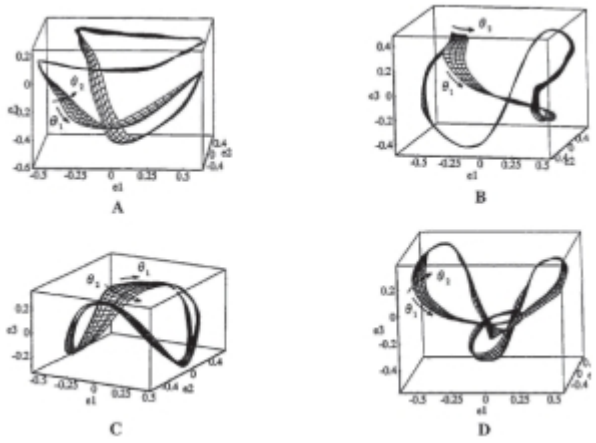


Figure 3.10.: Four different objects create four different Eigenspaces.[18]

Figure 3.10 a number of possible scatterplots which were created by using the Eigenspace method, described in [18], are shown.

If the location and orientation on of the object are the desired results, the calculated vector must be matched not just to the scatterplot but to the closest vector.

3.4.2. Model based matching

For the model based maching no previously prepared images are available. Instead a model or rather a set of attributes and their relations are prepared. These may be colors, distances, forms or other distinct attributes. To compare these models with the extracted data, the model is projected as genuine as possible in the image. On the other hand the extracted attributes can be used to create a model from these. This approach is more difficult and makes use of the graph theory where the model is compared to a semantic network. The use of the A*-algorithm and the Hungarian algorithm to match a model are described in [22].

4. Mathematical calculations

4.1. Geometrical approach

The controller shown in Figure 4.1 has a point invariant to rotation and translation, namely the center of the controller O. Since the picture taken by the camera projects the world onto a 2D-plane, it is possible to calculate the rotation angle around the Z-axis as well as translations in the X and Y directions. The translations are calculated as the distance between the original location of O and its new location O'. For the rotation around the Z-axis a triangle $\Delta OAA'$, where A' is the translation of new location A' of A, is used. Of course, to make these calculations independent to translations in X and Y directions, A' is translated by the previously calculated values into the original space. This process is shown in Figure 4.2. Furthermore Figure 4.2 displays the calculations to detect the rotation around Z.

$$\overline{AA'}^2 = \overline{OA}^2 + \overline{OA'}^2 - 2\overline{OA'}\overline{OA}\cos\alpha \tag{4.1}$$

Applying the well known equation (4.1) results in the rotation around the Z axis.

Detection of translation in Z direction as well as the detection of rotations around the X- and Y-axis depends solely on proportions of the distances as well as on detected areas, if available. If the controller is rotated around one of these axes the proportions of the distance between the center and the respective markers will change thus rotations can be deducted from these relations. If the areas of the markers are available as well, then the proportion of their sizes is a factor for rotation as well.

W.l.o.g. assume that the axis of $\overline{OA}$ is detected as the longer axis of the controller. Since a rotation around X- or Y-axis reduces the distance between the respective markers the longer axis has undergone no or a rotation of a smaller degree. By comparing this axis to its original a relative translation in the Z direction is detected.

This approach is mainly theoretical and in a number of imaginable cases the results are unsatisfying. Furthermore no calculations are possible if the controller has less than four markers or if one marker was not detected.

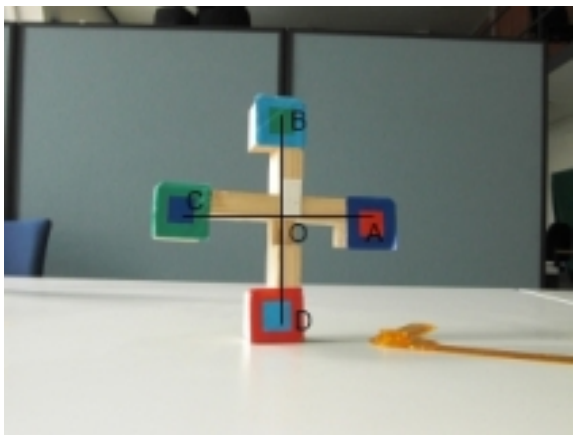


Figure 4.1.: A controller with its coordinate system.

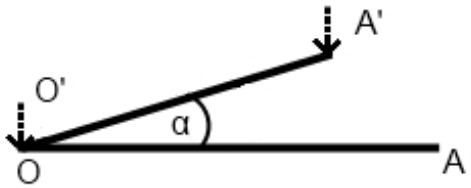


Figure 4.2.: Rotation around the Z-axis.

4.2. Analytical approach

The analytical approach is by far superior to the explained geometrical approach as it is more robust in its calculations. This approach was explained in [7, 12].

4.2.1. General explanations

After initializing the starting positions a transformation will lead to an error $\mathbf{e}(t)$. The mathematical approach aims to minimize the error $\mathbf{e}(t)$. This error is generally defined by

$$e(t) = s(m(t), a) - s^*. \tag{4.2}$$

The parameters in (4.2) are defined as following. The vector m(t) is a set of measurements from the current frame while a is a set of parameters with prior knowledge (e.g. 3D-model or camera intrinsic parameters). These are combined to calculate a vector of k visual features which we represent as s(m(t),a). The desired set of features is represented as s*, which is the identified position of the controller at initialization.

Next a velocity controller to minimize e(t) must be designed. The relation between the time differentiation of s and the camera velocity is necessary. The camera velocity is denoted as $v_c = (V_c, \omega_c)$ where V_c is the linear velocity and ω_c is the angular velocity of the camera frame. The relationship between $\dot{s}$ and v_c is given by

$$\dot{s} = L_s v_c. \tag{4.3}$$

In this equation L_s is the interaction matrix related to s and $L_s \in \Re^{kx6}$. Applying (4.3) on the equation (3.1) leads to the relationship between the camaera velocity and the time variation of the error. The result is shown in (4.4).

$$\dot{e} = L_e v_c \tag{4.4}$$

In this equation L_e equals L_s but is denoted different to represent it's relataion to $\dot{e}$. The velocity v_c is the input to the robot. Since the goal is to minimize the error e a multitude of minimization approaches, such as the lambda approach where $\dot{e} = -\lambda e$, can be used. With this assumption it is evident that

$$v_c = L_e^{-1} e. \tag{4.5}$$

L_e^{-1} is the inverse of L_e if $L_e \in \Re^{6x6}$. v_c can be calculated with equation 4.5 only if $\det L_e \neq 0$. Since L_e is unknown a way to estimate this matrix must be determined.

The interaction Matrix can be used to estimate L_e.

4.2.2. The Interaction Matrix

The coordinates of a point in the 3D-world can be described as P=(X,Y,Z). This point p is projected by the camera on a 2D-plane with the coordinates p=(x,y). This transformation is described by

$$\begin{aligned} x &= \frac{X}{Z} = \frac{u - c_u}{f\alpha} \\ y &= \frac{Y}{Z} = \frac{v - c_v}{f} \end{aligned} \tag{4.6}$$

where m=(u,v) are the coordinates of p expressed in pixel units and $a = (c_u, c_v, f, \alpha)$ is the set of the camera intrinsic parameters. c_u and c_v denote the coordinates of the principal point, in general the center of the image. f is the focal length and α is the ratio of pixel dimensions. The attribute s(m) is the position of the object in the picture thus $\forall s_i \in s(m) : s_i = p_i = (x_i, y_i)$. In the next step the time derivatie of the equation (4.6), that is

$$\begin{aligned} \dot{x} &= \frac{\dot{X}}{Z} - \frac{X\dot{Z}}{Z^2} = \frac{\dot{X} - x * \dot{Z}}{Z} \\ \dot{y} &= \frac{\dot{Y}}{Z} - \frac{Y\dot{Z}}{Z^2} = \frac{\dot{Y} - y\dot{Z}}{Z}. \end{aligned} \tag{4.7}$$

must be calculated. The relationship to the velocity v_c is described by

$$\dot{X} = -Y_c - \omega_c x X \Leftrightarrow \begin{cases} \dot{X} = -Y_x - \omega_y Z + \omega_z Y \\ \dot{Y} = -Y_y - \omega_z X + \omega_x Z \\ \dot{Z} = -Y_z - \omega_x Y + \omega_y X. \end{cases} \tag{4.8}$$

Injecting (4.8) in (4.7) and grouping the terms results in

$$\begin{aligned} \dot{x} &= -\frac{Y_x}{Z} + \frac{xY_z}{Z} + xy\omega_x - (1 + x^2)\omega_y + y\omega_z \\ \dot{y} &= -\frac{Y_y}{Z} + \frac{yY_z}{Z} + (1 + y^2)\omega_x - xy\omega_y - x\omega_z. \end{aligned} \tag{4.9}$$

This equation can be rewritten as the matrix multiplication

$$\dot{p} = L_p v_c \tag{4.10}$$

where

$$L_p = \begin{pmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1 + x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1 + y^2 & -xy & -x \end{pmatrix}. \tag{4.11}$$

The matrix L_p is necessary to approximate the matrix L_e in (3.4). It should be noted that a vector $v = (p_1, p_2, p_3)$ of three points in the detected frame is necessary, thus at least three points must be detected. This is a requirement to control a 6DOF robot. Stacking the equations for the respective points leads to

$$L_v = \begin{pmatrix} L_{p_1} \\ L_{p_2} \\ L_{p_3} \end{pmatrix} \tag{4.12}$$

From here on different approximations can be used to calculate L_e. The first approach is to estimate $L_e = L_x$, the current position of each point. For this approach the Z value of each point must be known, or estimated for each calculation. The second approach is to choose $L_e = L_{e^*}$. This L_e is calculated for the desired position, that means $e = e^* = 0$. For this approach the value of L_e is fixed for all calculations, thus an estimation is not necessary. Recently the approach to use the approximation of $L_e = 1/2 * (L_x + L_{e^*})$ was proposed. Once more the estimation of the value of Z for each calculation is required if this approach is chosen as it uses L_x. Approaches to estimate the depth Z are described in [8].

5. Implementation

5.1. Setup

This thesis was implemented in C++ and uses the OpenCV 1.0 distribution. OpenCV offers a variety of functions and classes for realtime image processing as well as objectdetection. This made OpenCV a natural choice to handle the controller detection.

The robot used for the tests was a ”Mitsubishi V6-S”, a 6DOF robot, and the controller was filmed by a ”lightvision marlin” camera. The setup is shown in Figure **??**. As for the framework for the interface the FlexRF framework [17] was used. This framework was developed to offer access to a flexible environment for real-time robot control. It consists of a number of blocks which can be added as needed to create an operating system. A moderatly complex system is shown in Figure 5.2.

As one can see the goal of this thesis was to develop a generic interface between the camera and the robot. Since the whole communication between the processing unit and the robot is handled by FlexRF this thesis concentrates on marker detection and the calculation of the transformations.

5.2. Markerdetection

To create a stable basis for calculations four markers were installed on the controller. The necessary number to calculate the transformations is three markers, but for the matter of stability a higher number is recommended. An analysis why four markers are necessery can be found in 5.4.2.

The markers were set at the four edges of the controller as seen in Figure 5.3. Their position was fixed for test purposes but general detection at initialization is supported as well.

A combination of cognition based detection and edgedetection was used to detect these markers. The Hough transformation is of no use for the used controller as it detects changes in size up to a specified degree, but rotations which distort the image of the object make it impossible to detect the markers. A possible approach is to use canny detection to determine the borders of the markers and thus determine the markers. This approach is equal to border

Figure 5.1.: The setup of the experiment.

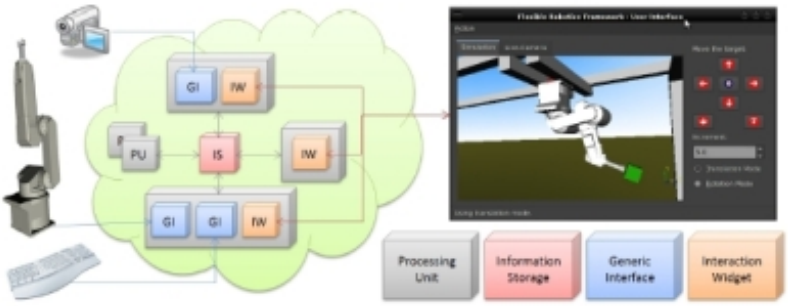


Figure 5.2.: A moderately complex sample application composed from the building blocks.[17]

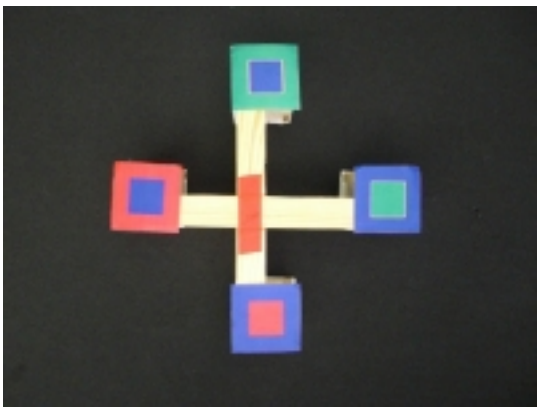


Figure 5.3.: The controller.

detection and following thus the contour detection function implemented in OpenCV can be used for this task as well. Before the contours can be detected, an environment which makes the results invariant to light must be created.

The standard cognition based algorithms, such as histogramms, fail on their own since a number of backgrounds can create histogramms similar to the desired result. This is a major backdraw of this technique. Furthermore it may prove quite time consuming to detect the desired object in its actual size with all possible transformations.

The first approach taken here was to use color normalization. After trying out the normalization method (5.1), which is recognized to give satisfactory results for normalization, as well as the CCN-normalization, which was proposed in [10], the results were by far unsatisfactory or the improvement in detection compared to the method actually used was only minimal. These observations tempt to agree with the results shown in [26, 21]. While both algorithms had good results when used on pretaken videos, tests with live feed could not support their utilization. One must note that both algorithms fail to adapt to an increase or decline in illumination sources.

$$\frac{r}{r+g+b}; \frac{g}{r+g+b}; \frac{b}{r+g+b} \tag{5.1}$$

Thus the picture was first transformed into the HSI space which allowed to skip the task of normalization as the results were satisfactory in regards of stability to changes in direction and intensity of the illumination. After changing the color system with *cvConvertImage()* the image contains only the color data of the original. In OpenCV the original $360°$ were mapped onto a value range of 0-180 to fit the 8-bit image. To ensure a robust detection, a setup which is

rare in possible surroundings was chosen. Each marker consists of two different colors, an outer and an inner color. The OpenCV function *cvInRange()* allows to filter the data of the raw image data for a certain range, the range for green would be 30 to 60 which equals an angle of 60° to 120°. No other color in the markers is filtered into this interval. These new frames contain all the data needed, but it may still be eronerous as there may have been errors in the visualization and filtering. First the image is smoothed with *cvSmooth()*, then *cvDilate()* is applied twice thus increasing the size of the detected areas. This way cases where a marker is not detected, because a part of it was not recognized, are reduced. Finally the threshold function *cvThreshold()* creates a binary image. The threshold was set to 230.

After reducing the image to a binary image, the algorithm explained in 3.2.3 can be used to extract the contours of the inner and outer colors. In OpenCV the function *cvFindContours()* uses this algorithm and extracts a list of contours, where a point of a contour is saved in a sequence and contains the information about the next point. The markers used for the controller are squares and any transformation applies in the same way to each marker respectively as well as to the inner and outer colors. Therefore the proportions of sizes do not change when a bounding box enclosing these respective colors is created. Rectangles of minimal size were used as the bounding boxes for the detected contours. These rectangles are created by *cvMinAreaRect2()*, which returns the center of the minimal rectangular bounding box, its angle, width and height. Furthermore a filter is applied to prevent that the calculation speed in the next step is reduced due to too many boxes of minimal size. The remaining boxes are saved in a list and used to calculate the marker positions.

To detect the location of the marker, the program checks which extracted inner and outer colors match the attributes of the markers. First of all the inner and outer colors are known for the markers. Furthermore the centers of the bounding boxes must be close to each other and size proportions must be within certain boundaries. If at least one of these requirements is not fullfilled, these boxes do not overlap, or overlap insufficiently.

After the search is completed all four markerpositions should be identified. If one marker was not identified then one rotation will be foregone during the calculations, in the case that two or more markers were not detected no calculations are done and the algorithms continues with the next frame.

Issues why the markers may have been overseen will be addressed in 5.4.1.

5.3. Calculation of the transformation

This project was designed to enable robot control without a calibrated camera, nor a model of the controller.

The approach of image based visual serving can be used for the calculations, assuming that the camera is an eye-in-hand camera. As was described in 4.2, three points are required for the calculations. Since four points are available, namely the centers of the four markers, v_c can be calculated a number of times for each possible selection. This number can increase if the center of the figure is added, but it was omitted as the center depends on the detection of the markers.

During the implementation errors while using OpenCV funtions for matrices operations were observed, thus each step was encoded.

The approach $L_e = L_{e^*}$ was used for the calculations. To calculate this value the camera intrinsic parameter f and the value of Z, for the initial position, must be assumed. Both values are reducing the result by a factor. After some try and error f was settled as 72 and Z as 2.

The error, e, is the difference of the x and y coordinates of each point and its original position. With a settled L_e and e the equation (4.5) to calculate v_c can be used. To invert L_e adjoint matrix calculations were used. After all matrices have been calculated the median of the results are calculated to reduce errors as much as possible. Further approaches tried were the majority rule and the median of the maximum. The results were similar to identical therefore the normalization was used. To ensure the proper relation of the detected transformations the rotations and the translations are normalized.

As one can assume the results can be used as direct input to move the robot, but different results for the same type of movement were perceived in different experiments. The calculations detected when the rotations changed, but the values retrieved changed sometimes as well. Therefore a further test was necessary. This test checks the size relation of the detected markers. Since the actual size of the markers is identical the 3D projection rules, which state that an object of the same size will be projected on a smaller area, the further it is from the camera, can be used. Therefore the rotations will be done in the direction of the bigger marker.

To ensure stability of the calculations the values of the rotations are forwarded only when the respective markers were identified. Issues of calculation behavior are discussed in 5.4.2.

5.4. Results

5.4.1. Stability of marker detection

The goal of this project was to create a controller which was stable to rotations of small degrees, any translation on the same plane, as well as translations towards and away from the camera up to a certain degree.

To control the stability of the detection three videos were taken at different times of a day as well as on different days. Each video was taken with the same camera settings and contained all motions that should be detected. Furthermore the videos contained phases where the controller was moved at different speed. Each second 30 frames were taken by the camera. To have a reasonable sample size videos of approximately 30 seconds each were taken and the number of frames where the markers were not detected at all or eroneously was checked. Furthermore the resulted calculations for each visible translation and rotation were checked to detect inconsistency.

When the controller was moved at high speed one marker was lost sometimes. Still, the moment the motion slowed down or was stopped altogether the marker was detected immediately. Furthermore one or more markers were lost when the controller was moved too far away.

The first video was taken at around 10 a.m. and was 616 frames long. Out of 60 frames with errors one marker was missed in 25 of these and two markers were missed in 35 frames. Analyzing these frames showed that these frames were either at a big angle, in rapid motion or very big distance to the camera.

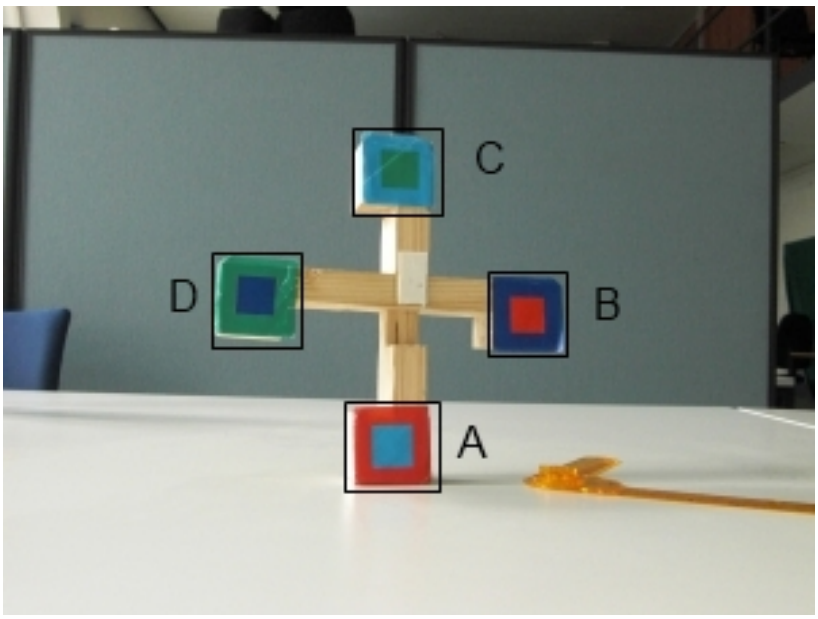
The afternoon video, at 1 p.m., contained 574 frames. Out of these a single marker was not detected in 17 frames and at least two markers were missing in 20 further frames.

In the last video, at 6 p.m., two markers were not detected in 23 frames and one marker in 22 frames. Furthermore three frames had an erroneous detection. All in all 750 frames were used for this test.

In all videos a number of frames with a visible error in the detected size was observed. Still, these differences were in single frames and of small proportions, thus they would have no impact on the longterm rotations.

All in all between 90% and 95% of the frames were detected correctly. If only errors with two markers missing are taken into account then a detection rate of $\approx$ 96% is reached. Furthermore, in cases where the controller was in a position relatively close to the starting position (which is best at about one meter distance to the camera) and in slowmotion the detection rate reaches close to 100%, as was observed during testsessions, while taking videos and is most reasonable to control the robot.

A backdraw of the approach used is that the detection ranges for the colors



must be set anew for a new environment. Furthermore if the difference of light and shadow is too great, for example afternoon and night, the detection fails. The final backdraw is that a number of colors in the background may greatly disturb the detection like strong illumination, which is reflected by the controller, or colors similar or identical with the outer color of the markers. A way to solve this is to have the markers consist of 3 colors, the outer color is consistant for all markers and is similar to a filter to block the background colors. Tests on this approach showed promising results.

5.4.2. Stability of the calculations

If the markers of the controller are denoted as in Figure 5.4.2 and the calculations are applied to it, then four triangular positions to calculate v_e are available, namely ABC, ABD, ACD, CBD. For each of these positions the matrix L_{e^*} is a fixed value. Thus the velocity v_e depends directly on the error e.

$$v_e = \begin{pmatrix} Y_x \\ Y_y \\ Y_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} \Delta x_1 \\ \Delta y_1 \\ \Delta x_2 \\ \Delta y_2 \\ \Delta x_3 \\ \Delta y_3 \end{pmatrix} = \begin{pmatrix} \Delta P_1 \\ \Delta P_2 \\ \Delta P_3 \end{pmatrix}. \tag{5.2}$$

As shown in (5.2), if only one set of three markers without any permutation of the order is used, a number of rotations as well as translations cannot be detected. The detection of these would result from the transformation of the position after a number of iterations. As it was shown in [12] it would take approximately 30 cycles of calculation to get a sufficient result. But since the approach taken in this thesis does not transform the picture, thus the position may remain the same all the time and the calculations must be correct at each cycle.

For this reason four triangles are calculated, where each point takes the position of P_1, P_2 and P_3 once. If only three markers were detected, calculations for three triagles are done. W.l.o.g let A, B, C be detected. In this case $\Delta ABC, \Delta BCA, \Delta CAB$ are used. As shown in (**??**) there is a linear relation between v_e and e. Running a test on the results from the calculations by the algorithm has shown the same result. Furthermore, the translations are detected in all calculations, but the rotations around the x and y axis are superficial only within two calculations while the rotation around the z axis results from all four calculations.

As stated in [[9]] the calculations are correct if the transformation does not approach singularity positions. Still, even though all rotations and translations can be detected the proportions of the calculated movement in each direction and around each axis are not consistent. As such a translation in Z-direction is detected but its magnitude is by far inferior to translations in X- and Y-directions. Among the rotations the rotation around the Z-axis is dominant.

6. Conclusion

In this thesis it was shown that a visual controller can rival existing controllers which rely on vision and further built-in sensors to detect motion. The current approach detects all markers within an acceptable margin of error and is capale of detecting the translations, rotations and any combinition of these with satisfying accurancy. Any motion performed with a space mouse, a commonly used robot controller, can be achieved with the discussed controller as well. The marker detection is quite sensible to position changes thus ongoing correction of submovements is necessary. Still the method of marking the controller edges has a number of flaws, such as weakness to similar color in the background, weakness to great rotation angles as well to large distance and rapid motion. In the future a test with light balls instead of color markers, as they are used in the Sony Controller, may counter these problems. Furthermore a counter measure for a missing marker may be a combination with a model based approach. This approach would require a model of the marker thus impusing further restrictions on the controller. An adaptive, built-in model which is created based solely on the initial position of the controller would be of great interest for further research. Another approach to detect the marker may be to limit the search area by applying an adaptive filter which detects the background thus limiting the search for the markers and improve the stability of the detection even further. This approach would enable the application of snakes to remove any eroneous detection of the marker borders as the number of erroneous marker colors would decrease enough not to cause a slow down thus realtime controlling would still be possible.

The current algorithm enables the user to control a robot directly or from a remote location with a camera filming the robot and the user and transfering the commands to the robot in a different location. A second possible application area may be cooperation between the operator and the teleoperator. The controller was designed by using calculations which assume that the camera is an eye-in-hand camera thus controlling the robot with a camera fixed onto the robot gripper is imaginable as well. This can be achieved by no or only small changes in the code.

Appendix

A. Quickstart Guide

As it was mentioned the parameters for color detection must be defined anew when the robot is used in a new environment. The function *testValues(IplImage* src)* supports the setup of a new configuration. The user is presented with an OpenCV window which contains two trackbars and displays the binary output of the image after applying the color-range threshold, smoothing, dilating and applying the final threshold. The window is shown in Figure A.1. The values of the color-range threshold are set by the trackbars. Normally the upper trackbar contains the lower threshold while the lower trackbar determines the upper threshold. In this case the output is the binary image in the range of [*lower threshold*, *upper threshold*].

If the upper trackbar receives a higher value than the lower trackbar the programm assumes that the user tries to determine the range for red. In this case the displayed image is the combination of the range [*lower threshold*, 180] and [0, *upper threshold*].

After the desired values for color detection as well as the robot axises and the movement threshold values have been set in the configuration file the interface can calculate the resulting transformations. The boolean *doSetTarget* is set by the programm during each cycle and should be checked after the calculation. This variable is set to *true* if the controller was discovered, otherwise it is *false*. This way further control of the robot is possible, for example if the operator uses the built-in FlexRF-interface to set the desired position.

The function to calculate the movements is *newPosition(IplImage* src, core::data::Target current)*. This function will return the new position, as a target, which can be forwarded to the robot. While the programm is executed the initial position of the controller is displayed in the *processed window* of the FlexRFframework as red squares, while the online recognition is displayed as green squares. This enables the user to discover any erroneous detection. The running interface is shown in A.2.

In the case that the user wishes to reset the initial position, the function *resetInitial()* can be used.

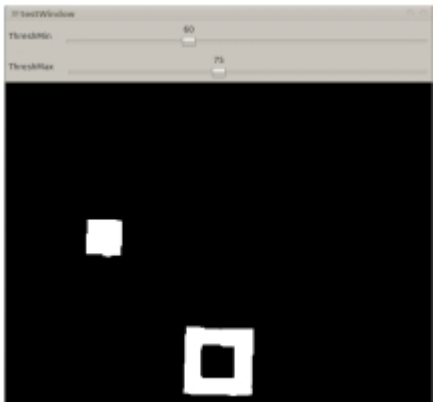


Figure A.1.: The window for color-range detection.

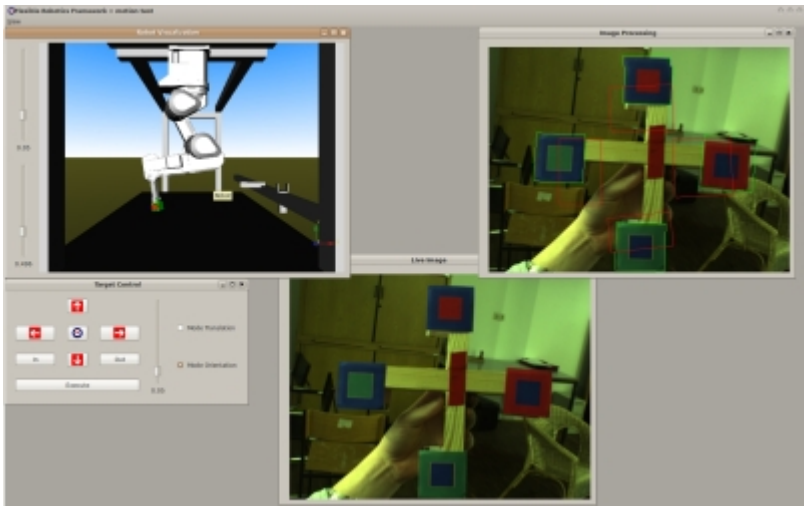


Figure A.2.: The interface and its desktop.

List of Figures

Bibliography

[1] http://en.wikipedia.org/wiki/File:Wii_Remote_Image.jpg.

[2] http://en.wikipedia.org/wiki/File:Nintendo_Wii_Sensor_Bar.jpg.

[3] http://en.wikipedia.org/wiki/File:PlayStation_Move_Final_Design.png.

[4] http://bin.sulinet.hu/inform/szinkoordinata/cie_xyz.png.

[5] http://www.lohninger.com/helpcsuite/img/hsi_color_system.png.

[6] Peng Chang and J. Krumm. Object recognition with color cooccurrence histograms. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, page 504 Vol. 2, 1999.

[7] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *Robotics Automation Magazine, IEEE*, 13(4):82 –90, dec. 2006.

[8] F. Chaumette and S. Hutchinson. Visual servo control. ii. advanced approaches [tutorial]. *Robotics Automation Magazine, IEEE*, 14(1):109 –118, march 2007.

[9] François Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing, 1998.

[10] Graham D. Finlayson, Bernt Schiele, and James L. Crowley. Comprehensive colour image normalization, 1998.

[11] Zhiqiang Hou and Chongzhao Han. Force field analysis snake: an improved parametric active contour model. *Pattern Recognition Letters*, 26(5):513 – 526, 2005.

[12] Seth Hutchinson, Greg Hager, and Peter Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12:651–670, 1996.

[13] M. Jagersand, O. Fuentes, and R. Nelson. Experimental evaluation of uncalibrated visual servoing for precision manipulation. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 4, pages 2874 –2880 vol.4, apr 1997.

[14] Lilian Ji and Hong Yan. Attractable snakes based on the greedy algorithm for contour extraction. *Pattern Recognition*, 35(4):791 – 806, 2002.

[15] George H. Joblove and Donald Greenberg. Color spaces for computer graphics. *SIGGRAPH Comput. Graph.*, 12(3):20–25, 1978.

[16] Christian Große Lordemann and Martin Lambers. Objekterkennung in bilddaten. Technical report, Westfälische Wilhelms-Universität Münster, 2004.

[17] T. Müller and A. Knoll. A generic approach to realtime robot control and parallel processing for industrial scenarios. In *IEEE International Conference on Industrial Technology*, 2010.

[18] S.K. Nayar, H. Murase, and S.A. Nene. Parametric Appearance Representation. In *Early Visual Learning*, pages 131–160. 1996.

[19] Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing*. Academic Press, Inc., Orlando, FL, USA, 1982.

[20] José Santos-Victor. Vision-based remote control of cellular robots. *Robotics and Autonomous Systems*, 23(4):221 – 234, 1998. Intelligent Robotics Systems - SIRS'97.

[21] Gerald Schaefer. How useful are colour invariants for image retrieval. In *Proc Second Int'l Conf. Computer Vision and Graphics*. Kluwer Academic Publishers, 2004.

[22] Simone Schäfer. Modellbasiertes matching (a*, ungarischer algorithmus). Technical report, Institut für Computervisualistik, Universität Koblenz-Landau, 2006.

[23] Alvy Ray Smith. Color gamut transform pairs. *SIGGRAPH Comput. Graph.*, 12(3):12–19, 1978.

[24] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32 – 46, 1985.

[25] Gordon Wells, Christophe Venaille, and Carme Torras. Vision-based robot positioning using neural networks. *Image and Vision Computing*, 14(10):715 – 732, 1996.

[26] Dietrich Paulus Wolfram Hans, Benjamin Kopp. Farbmetrische objekterkennung. Technical report, Universität Koblenz-Landau, 2009.

[27] Chenyang Xu and Jerry L. Prince. Generalized gradient vector flow external forces for active contours. *Signal Processing*, 71(2):131 – 139, 1998.

[28] Shigeki Yokoi, Jun ichiro Toriwaki, and Teruo Fukumura. An analysis of topological properties of digitized binary pictures using local features. *Computer Graphics and Image Processing*, 4(1):63 – 73, 1975.