EXAMINE VISION TECHNOLOGY FOR SMALL OBJECT RECOGNITION IN AN INDUSTRIAL ROBOTICS APPLICATION

Jonas Martinsson
jmn09009@student.mdh.se

May 20, 2015
This page is intentionally left blank.
Abstract

This thesis explains the development of a computer vision system able to find and orient relatively small objects. The motivations is exchanging a monotonous work done by hand and replace it with an automation system with help of an ABB IRB 140 industrial robot. The vision system runs on a standard PC and is developed using the OpenCV environment, originally made by Intel in Russia. The algorithms of the system is written in C++ and the user interface in C++/CLI. With a derived test case, multiple vision algorithms is tested and evaluated for this kind of application. The result shows that SIFT/SURF works poorly with multiple instances of the search object and HAAR classifiers produces many false positives. Template matching with image moment calculation gave a satisfying result regarding multiple object in the scene and produces no false positives. Drawbacks of the selected algorithm developed where sensibility to light invariance and lack of performance in a skewed scene. The report also contains suggestions on how to proceed with further improvements or research.
This page is intentionally left blank.
## Contents

1 Background ........................................ 1
   1.1 Robot System Products .................................. 1
   1.2 Company products ........................................ 1
   1.3 Tool changers ............................................ 2

2 Motivation ........................................... 2

3 Problem formulation ................................... 3
   3.1 In- and outflow of components ............................ 3
   3.2 Find and orient the terminal pins .......................... 3
   3.3 Pick and place the pins .................................... 4

4 Literature study of computer Vision ................... 6
   4.1 Limitations .................................................. 6
   4.2 Filtering .................................................... 7
   4.3 Filtering technique .......................................... 7
   4.4 Hue Saturation Value ....................................... 7
   4.5 Binary image ................................................ 9
   4.6 Mathematical Morphology ................................... 9
       4.6.1 Dilation ............................................... 9
       4.6.2 Erosion ............................................... 9
       4.6.3 Opening and closing .................................. 10
   4.7 Edge detection and contour extraction .................... 11
       4.7.1 Canny .............................................. 11
       4.7.2 Image moments ........................................ 11
       4.7.3 Hu moments .......................................... 12
       4.7.4 Principal component analysis .......................... 12
       4.7.5 Template matching ...................................... 13
   4.8 Scale-invariant feature transform and Speeded up robust feature 14
   4.9 HAAR classifiers ........................................... 14

5 Requirements .......................................... 15
   5.1 Software related functional requirements .................. 15
   5.2 Hardware related requirements ............................. 15

6 Test setup ............................................. 16
   6.1 Vision test setup .......................................... 16
   6.2 Vision algorithm performance .............................. 17
   6.3 HAAR classifier ............................................ 17
   6.4 SIFT/SURF ............................................... 18
   6.5 Image moment + Template match ............................ 19
6.6 Performance summary ........................................... 20

7 Implementation .............................................. 21
7.1 Program test and algorithm verification ...................... 21
  7.1.1 Fetch image from camera ................................. 22
  7.1.2 HSV filtering + binary threshold ....................... 22
  7.1.3 Mathematical morphology ............................... 22
  7.1.4 Edge detection with canny ............................... 23
  7.1.5 Calculate image moments .............................. 24
  7.1.6 Perform PCA analysis ................................ 25
  7.1.7 Template match ....................................... 26
  7.1.8 Graspability analysis .................................. 27
7.2 Graphical user interface ................................... 29

8 Real world adaptations ........................................ 30
  8.1 Camera .................................................... 30
    8.1.1 Selecting a camera .................................. 30
    8.1.2 Selecting a lens .................................... 30
  8.2 Mapping camera pixels to millimeters ..................... 31

9 Complete system testing ....................................... 33
  9.1 Pixel-mapping test ....................................... 33

10 Summary ..................................................... 36

11 Future work .................................................. 37
  11.1 Pin distribution ......................................... 37
  11.2 Design in- and outflow of components .................. 37
  11.3 Decrease the systems sensitivity to light invariances 37

References .................................................... 38
List of Figures

1. The TC480-SWS Tool changer
2. Two different plastic bodies
3. Terminal pins arranged for the vision system
4. Exploded section view of the plastic body with a pin partially inserted
5. The RGB color-space room
6. The HSV color-space cone
7. Mathematical morphology closing algorithm demo
8. An illustration of the Principal Component analysis
9. The vision system test setup
10. Haar classifier demo
11. SURF demo
12. Template match demo
13. The vision system program flow
14. HSV filtered and thresholded image
15. Image after applying a mathematical morphology filter
16. Image after Canny contour extraction is applied
17. Image with classified objects after HU-moments and area calculation
18. Demonstration of the object-angles computed by PCA analysis
19. Demonstration of the template match algorithm
20. Demonstration of the graspability analysis
21. Demonstration of the graspability analysis algorithm
22. The Graphical User Interface
23. Field of view illustration
24. Outline illustration of the complete system
25. The calibration image for work object definition
1 Background

In the early seventies Björn Weichbrodt and a couple of his engineers started the work on developing a Swedish industrial robot for ASEA by assignment from the CEO of that time, Curt Nicolin[1]. The robot became the first fully electrical and microprocessor controlled industrial robot. The first sold IRB6 was delivered to a small company in southern Sweden, but just a year later, in 1975, ASEA had orders coming in from worldwide. The first international customers where found in USA, Germany and United Kingdom[2]. The benefits of using an industrial robot in production spread worldwide and soon the industry started planning new manufacturing facilities designed to fit the robots. Along with the robot revolution, the need for tools and robot related equipment increased. Companies saw a business opportunity in dressing robots and provide tools for machining. As the industry developed, higher standards had to be reached. The customers demanded more and more complex solutions in order to increase their productivity and production quality.

1.1 Robot System Products

Robot System Products started as a division within ABB robotics, in 2003 two former ABB employees bought the company from ABB robotics. Their motto reads ‘Dressing robots worldwide‘ and the company presentation, quoted from their website[3] reads:

Robotic System Products has a long tradition in developing, producing and marketing robot peripheral equipment for all the major robot brands. The philosophy is to bring standard, high quality products that needs the minimum of installation time and maintenance without adding limitations to the robots performance. With the headquarters based in Västerås, Sweden, Robot System Products has its own representation in many countries and through agencies around the world.

1.2 Company products

Robot System Products main products are swivels and tool-changers(Fig.1) to robots. The swivel would let the robot spin the tool around without risking cable tangling or hose breakage. The tool-changer would let the robot automatically change the processing tool without interaction from the operator. The robot could then accomplish more than one job which could increase the productivity as well as decrease the investment.
1.3 Tool changers

The tool-changers can be arranged to transfer electrical signals, compressed air, water and other fluids and gases depending on the customers requirements. The transferring of signals, gases and fluids between the tool changer and the tool attachment is possible thanks to hose couplings and resilient electric connections. In the event of a tool change, gases and fluids are automatically shut off. The electronic connections are made by a spring loaded connection pin on the tool changer and a flat connection area on the corresponding tool attachment.

2 Motivation

The connection pins on the tool attachment are mounted by hand in the local workshop. This is a demanding job, mostly because of its unilateral nature. The work is also time consuming and in a way so monotonous that a robot could do the job. If the handling of the process becomes easy manageable, anyone could feed the robot with new terminal pins and bodies. For example, the operator places a hundred terminal pins on a collecting space and around ten bodies on a pile in some jig. The robot will place the pins on the bodies and leave the assembled piece in some other jig.
3 Problem formulation

The process should be easy manageable and easy to survey where the actual job takes place in a safe area. The entire system must be dependable and reliable in order to work without human interaction. To get an overview of the problems involved when designing the system, the problems to be addressed can be divided into the following parts.

3.1 In- and outflow of components

The inflow of terminal pins should be handled without the need to pause the entire process and the same goes for the bodies. The pins could be placed on a collecting area with a form that fits the vision system. The plastic bodies (Fig.2) needs to be placed in a way that the robot can reach them.

![Figure 2: Two different plastic bodies, property of Robot System Products](image)

3.2 Find and orient the terminal pins

As required by the company, a vision system will be used to find and orient the terminal pins (Fig.3). The problem consist of evaluating different vision system and technologies as well as find or design complementary parts required by the vision system. Another issue is the calibration of the camera(s) and make a good camera stand for proper vision, aspect ratio etc.
3.3 Pick and place the pins

This is the major part of the work. The robot has to be programmed with data feed from the vision system and, based on camera data, find its way to the pin. Depending on which system is used for localization and orientation of pins, a program that selects a pin from a group of pins may have to be applied. Next stage will be gripping the pin and different gripping technologies has to be evaluated before designing the final gripper structure. When placing the pin in the body, either a jig or another camera can be used. The problem with a jig could be the tolerances in the jig itself and the material. If a camera is used for localizing the mounting holes on the body, the same issues as with the pins is applicable. The pins is mounted in the holes with press-fit tolerances which could require more force applied than the robot could provide (Fig.4), in that case some compression station might be necessary.
Figure 4: Exploded section view of the plastic body with a pin partially inserted, property of Robot System Products
4 Literature study of computer Vision

Computer vision has been used within the manufacturing industry for decades. In it’s early days, small pictures were analyzed with algorithms that took minutes to finish. As the machines on which the algorithms was computed became a lot faster and the research presented faster and computationally cheaper algorithms, more tasks than ever can now be handled with computer vision[11].

4.1 Limitations

The vision part of this project aims to conclude and evaluate different technologies available for vision applications. The explanations of the algorithms is therefore simplified and the interested has to search for further information in the original papers stated in the reference list at the end of the report.
4.2 Filtering

The result from the image analysis mostly depends on how well the picture to be analyzed is filtered. The main goal is to intensify the interesting data and suppress the insignificant data. The filtering technique for digital image processing started in the early seventies by a pioneer in the area named Nasser E. Nahi, a member of the IEEE group[4]. Nasser developed an algorithm for correction of scanned documents and images from a flatbed scanner[5]. In the scanning process, several sources of errors were found that distorted the picture. That includes reflection from spurious objects, inaccuracies in the sensing mechanism or corruption in the transmission between the scanner and other equipment[5]. Nasser’s algorithm is derived from the allegation that the distortion in the picture was scattered according to a statistical pattern. By utilizing the Gaussian distribution of the distortion, Nasser could cancel out the noise within the Gaussian distribution. A few years later, Lee shows the benefits of dividing the picture into smaller pieces of given sizes, before applying the filter. By treating every sub-picture individually he achieved better results than had been done before[6]. This type of filtering had a few drawbacks as it turned out, when applying a Gaussian blur to a picture, some of the interesting features along with the noise where suppressed[4].

4.3 Filtering technique

Today there are many filtering techniques available and described in all sorts of research papers regarding computer vision. Some of them are just improvements of the Gaussian blur previously described, other use totally different techniques or a mix of all. How advanced or complicated filter to choose is not an easy question, however, choosing a filter depending on the conditions of the entire vision system and what the expectations and gains are is a good starting point. Assuming good conditions including white background and good lightning, a simple filter might do the job good enough. The next sessions describes some easy implemented and adjustable filtering structures.

4.4 Hue Saturation Value

When dealing with images in digital analysis color are often described by the Red Green Blue (RGB) scale, a vector with three elements for the colors. Every pixel in the image has an RGB value corresponding to its color. The color of the pixel can be described as the vectors position in the three dimensional RGB room see fig.5.
Colors can also be described in several other systems where Hue Saturation Value (HSV) is one of them. In the HSV color scale the colorspace is not described as a vector in the room but instead as position in a cone[7]. The three colors (RGB) in their base is placed along the circumference of the cone. By applying an angle relative to an axis through the cone, the base color is defined. Saturation is defined as the distance from the cones center axis towards the periphery and value as the placement along the central axis, see fig.6. The benefits of converting the image to the HSV color-space is amongst others, the opportunity to easy choose in which color span the filter shall act. This could also be done in the RGB system but the variances in the image introduced by the vision system often corresponds better to the Saturation and Value properties of the HSV. In other words, the contrast and brightness features in a picture is more likely to correspond to Saturation and Value instead of the base-color (Hue)[8]. To achieve the same result with RGB, all three parameters has to be changed just to set the filter span to suppress or amplify different contrast and brightness variances.
4.5 Binary image

A binary image is, as the name suggests, an image that only consist of two colors usually black and white. The binary image is derived from a gray-scaled image where the level of gray decides whether the pixel should be converted to black or white. If the gray-level lays below some threshold value the pixel becomes white and if it's above the threshold the pixel becomes black. The benefits of this is reducing the amount of information contained in the picture. Less information in the picture results in faster and smaller analyzing algorithms.

4.6 Mathematical Morphology

Mathematical Morphology is one of the earliest filtering-algorithms suitable for object detection systems [9]. The algorithm is very powerful and contains two different operators, dilation and erosion. The operands gives different outputs depending on the order they are applied to the image, the order is best chosen by looking at the condition of the image. The Morphological operators are best applied to binary images where no consideration has to be taken to the underlying color or gray-scale.

4.6.1 Dilation

The dilation operator works with two elements, the original image and a structuring element. By applying the structural element onto the original picture, holes and group of pixels will be joined together, this creates solid elements which rarely appears directly in the image. The election of the structural element is a key variable when designing the filter, a well chosen structuring element will improve the dilation. Observe that any remaining noise like single pixels will also be amplified/enlarged.

4.6.2 Erosion

The erosion operand needs the same ground conditions as the dilation. Erosion is the morphological dual to dilation[9]. This means that instead of applying material to the picture, as done in the dilation, erosion subtracts material by the structuring element, thereby the name erosion. This is beneficial since the erosion will remove single pixels far from nearest neighbor, pixels that are considered to be noise. The nearest neighbor distance is defined entirely by the structural element.
Opening and closing

By combining the dilation and erosion operator, different results could be achieved regarding the original image. By first applying dilation and then erosion gives the operand closing. The closing operator fill holes in objects by the dilation, which also will make the object larger. Then the erosion will restore the the object to its original size by shrinking it, but now as a solid object without holes. Simultaneously will the erosion remove some of the noise in the picture, given a bigger erosion than dilation structural element. The corresponding operation is called opening, where the order of appliance is reversed. The erosion will remove noise and small features that are of less importance, while the object also will be smaller. The object is then expanded to it’s original size by dilation. Fig. 7 demonstrates the mathematical morphology operator closing.

**CLOSING**

<table>
<thead>
<tr>
<th>Original Image</th>
<th>Dilation</th>
<th>Erosion</th>
<th>Structuring element</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Diagram" /></td>
<td><img src="image.png" alt="Diagram" /></td>
<td><img src="image.png" alt="Diagram" /></td>
<td><img src="image.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

Figure 7: A demonstration of the Closing algorithm, picture from [10]
4.7 Edge detection and contour extraction

Edge detection and contour extraction aims to extract useful structural information from the image and at the same time reduce the amount of processed data. It is beneficial to use these kinds of algorithms since they tend to bring forward the features that defines the object. When the contours is used to define the object, the system is less sensible to invariance in brightness or light given a proper filter is adapted before the contour extraction.

4.7.1 Canny

The Canny operator was developed during the 80:s by John. F. Canny[12] and was based on previously developed edge detection algorithms. Canny introduced a number of rules on which his algorithm was created[13]:

- Detection: The probability of detecting real edge points should be maximized while the probability of falsely detecting non-edge points should be minimized. This corresponds to maximizing the signal-to-noise ratio
- Localization: The detected edge should be as close as possible to the real edges.
- Number of responses: One real edge should not result in more than one detected edge.

Canny where able to develop an algorithm that achieved those rules, by using different techniques in steps and let the result evolve during the procedure. Starting with an input image, Canny applies a Gaussian blur filter to smooth out the sharp edges. From the smoothed picture, Canny calculates the image gradient magnitudes without applying non maxima suppression. By applying threshold to classify the edges in strong and weak classes, Canny can use hysteresis to find out which weak edges that is connected to a strong edge. Remaining is the filtering according to the hysteresis values earlier calculated.

4.7.2 Image moments

Image moment can be described as a descriptor in direct relation to the object being analyzed. The Image moment is in analogy with the physical moment describing the dynamical features of the object. The image moment descriptor can be derived in several steps, where the first always establishes the area of the object in a 2-dimensional space. Further derivation of the moment could be used to calculate the center-of-gravity and relative angle in relation to one of the camera axes[14].
4.7.3 Hu moments

To find out whether the object being analyzed has the same features as a reference object, Hu moments is a good practice. Hu moment is a variant or further derivation of the image moments. The regular image moments are sensitive to rotation, translation and scale. Whereas the Hu moments are considered to be rotation, translation and scale invariant. Saying that no matter rotation of the object, where in the image the object is and the scale of the object, the seven Hu moments remains the same as the reference object Hu moments. The Hu moment thereby practice as a good object identification whereas the image moments holds information about rotation, translation and scale[14].

4.7.4 Principal component analysis

Principal component analysis is a mathematical procedure which lays in the field of statistical computing. The aim of the process is to find the principal components of a given data set of variables[16]. The principal components is vectors arranged in the direction where the variance in data is the biggest[17]. Consider an N-dimensional matrix, such a matrix has N principal components, orthogonal to each other. The example Fig.8 below illustrates a 2-dimensional matrix with points. The first principal component (red) is oriented in the direction on where the points varies the most. The other vector, since it is a two dimensional matrix, it has two vectors, lays where the points vary the most along the first component. Principal component analysis is therefore a god tool to find the orientation of an object[15].

Figure 8: An illustration of the Principal Component analysis, picture from [15]
4.7.5 Template matching

Template matching begins with a big image and a smaller template image. The smaller image is slid in both X and Y directions to find where in the bigger picture the template image fits the best. There are several different techniques to determine what is a good fit for this purpose\[18\].

- Squared difference matching method
  $$R_{sq\_diff}(x, y) = \sum_{x' y'} [T(x', y') - I(x + x', y + y')]^2$$

- Coorelation matching method
  $$R_{ccorr}(x, y) = \sum_{x' y'} [T(x', y') * I(x + x', y + y')]^2$$

- Coorelation coefficient matching method
  $$R_{ccoeff}(x, y) = \sum_{x' y'} [T'(x', y') * I'(x + x', y + y')]^2$$

where

$$T'(x', y') = T(x', y') - \frac{1}{(w*h)} \sum_{x'' y''} [I(x + x'', y + y'')] [I(x + x'', y + y'')]$$

$$I'(x + x', y + y') = I(x + x', y + y') - \frac{1}{(w*h)} \sum_{x'' y''} [I(x + x'', y + y'')] [I(x + x'', y + y'')]$$

$I$ denotes the input image, $T$ the template image and $R$ the result.

OpenCV has built-in features for normalized template matching methods, the normalized methods could increase the dependency of the light invariances in the images. The normalized versions where developed by Rodgers[19] and is the same for the above matching methods

$$Z(x, y) = \sqrt{\sum_{x' y'} T'(x', y')^2 * \sum_{x' y'} I'(x + x', y + y')^2}$$
4.8 Scale-invariant feature transform and Speeded up robust feature

In 1999, David G. Lowe presented a new object recognition algorithm based on local image features. Earlier object recognition algorithms had poor results in vision harsh environments. Lowe's algorithm is invariant to scale, translation, rotation and partially to illumination and affine 3-D projection.[20] Lowe was inspired by the vision of mammals and studied how the temporal cortex analyzed images. When the Scale-Invariant Feature Transform (SIFT) algorithm is exposed to a picture, it recognizes or generates on the order of 1000 SIFT keys. Quoted from the original paper is the explanation of how SIFT works[20]:

The SIFT keys derived from an image are used in a nearest-neighbor approach to indexing to identify candidate object models. Collections of keys that agree on a potential model pose are first identified through a Hough transform hash table, and then through a least-squares fit to a final estimate of model parameters. When at least 3 keys agree on the model parameters with low residual, there is strong evidence for the presence of the object. Since there may be dozens of SIFT keys in the image of a typical object, it is possible to have substantial levels of occlusion in the image and yet retain high levels of reliability.

In May 2006 Bay et al. presented their Speeded Up Robust Feature (SURF) detector. It is based upon the SIFT detector by Lowe but the computational time is reduced and the detector is claimed to give better results. The high performance is due to the use of cascade filters where the difference Gaussian is calculated on rescaled images progressively[21].

4.9 HAAR classifiers

In 2001, Viola and Jones presented a classifier for visual object detection based on machine learning. The motivation was a fast and reliable detector, able of the task of face detection[22]. The algorithm developed is based on machine learning and therefore requires training examples, both positive samples where the object wished to be detected is present in the image and negative samples where the wished object is missing. The samples is feed through a training algorithm to generate the finished classifier.
5 Requirements

The vision system needs to correctly classify the objects of interest. The pick and place application relies on the vision system to give it accurate data to process. The vision system consists of both software and hardware components, why the goals and requirements are divided into the following sections.

5.1 Software related functional requirements

The following requirements is set up for the vision system:

- Find all objects placed on the scene (field of view)
  - Crucial for a pick test algorithm and system reliability
- Produce no false positives, only true objects will be detected
  - For reliability and effectiveness
- Give appropriate results regarding the translation and rotation of the objects
  - Crucial in order to position the gripper correctly
- Classify an object as pickable or not
  - Distinguish whether there is enough space for the gripper to grasp the pin.
- Be reliable in different lightning conditions
  - For robustness

5.2 Hardware related requirements

The software components relies on the data feed from the hardware components. Suitable application hardware decreases the complexity of the program code since the errors introduced by the hardware has to be corrected by the software. Also, the software demands some features to function at all. Following hardware requirements are set for the hardware of the system.

- A color camera
  - Most algorithms uses colors in some way to define the objects.
- A lens with no or small distortion factor
  - To not depend on the accuracy of the camera calibration algorithms
- Sufficient lightning
  - A system that sees will not work without appropriate lightning
6 Test setup

The evaluation process starts with derived experiments to test the different technologies available for solving the problems. The goal is finding a suitable test for the different part of the project and evaluate the results in order to know how to proceed. The tests will be designed to see how the technologies fit the requirements for the system.

6.1 Vision test setup

To validate the performance of the different vision algorithms, a test setup with the objects to be detected where placed out in front of the camera. To cover most of the cases the objects where placed in different directions and in different constellations. See Fig.9.

![Figure 9: The vision system test setup](image)

This setup will test the ability of the system to correctly classify the objects orientation, rotation and size. The same template image is used to test all three of the algorithms.
6.2 Vision algorithm performance

This section states the result after a simple evaluation of the different algorithms. The algorithms is gathered from different examples found in the Learning OpenCV book [24] and the examples provided within the OpenCV library.

6.3 HAAR classifier

The HAAR classifier example comes within the OpenCV package as a standard demo project. A program analyzes images with the object in present, positive samples, and images without the object with just backgrounds, negative samples. The program outputs a descriptor file of a trained neural network, which can be used by the vision application. The performance of the neural network depends on the samples the network was trained with, see section 4.9. In this evaluation test, 50 positive samples and 50 negative samples were used to train the neural network. Figure 10 shows the performance of the trained algorithm.

Conclusion of the algorithms performance:
(-) Produces false positives
(+) Can find objects with variances to the originals (different faces)[22]

Figure 10: Haar classifier demo, the image shows detected objects in circles. The classifier produces many false positives.
6.4 SIFT/SURF

The SIFT/SURF demo comes within the OpenCV package as a demo. The program is fed with a template image, seen in the upper left of figure 11, and an image on where the objects is to be recognized. The program finds key-points in the template image and looks for those in the bigger picture, refer to section 4.8. As can be seen in figure 11, the key-points from the template image corresponds to key-points of different pins.

Conclusion of the algorithms performance:
(-) Works poorly with multiple instances of the same object
(+ ) Can detect affined and wrapped objects in harsh environments[20][21]

Figure 11: SURF demo, the algorithm works poorly with multiple instances of the template object. Template object with SURF keypoints seen in the upper left.
6.5 Image moment + Template match

The image moment + Template match algorithm is written with pieces of code gathered from the OpenCV book [24] as well as pieces of code from the examples provided with the OpenCV library. In difference from the previously tested algorithms, the Image moment + Template match algorithm is a collection of features provided by the OpenCV library. Hence more time was required to develop the system for initial testing. The algorithm classifies objects according to size, in figure 12 can be seen that four object falls within the constraints while two objects are outside the size constraints.

Conclusion of the algorithms performance:

(-) Needs a clear environment and image, sensitive to scale and affined transformations

(+) Detects all objects and produces no false positives

Figure 12: Template match demo, the algorithm is able to classify the objects without producing any false positives
6.6 Performance summary

A summary of the results after the minor evaluation is shown in the table below.

<table>
<thead>
<tr>
<th></th>
<th>Detect all objects</th>
<th>Produces no false positives</th>
<th>Correct positioning</th>
<th>Light invariance</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAAR classifier</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>SIFT/SURF</td>
<td>-</td>
<td>N/A</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Image moment + Template match</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

The data from the table concludes that the Image moment + Template match algorithm is the most suitable choice for this kind of application. The light invariance problem introduced by that algorithm could be addressed with proper hardware in form of appropriate lightning. The program do not need to classify other objects than the terminal pins and do not need to find them on different backgrounds, why the HAAR classifier is redundant. The pins are also not projected in afflined transformations since the camera and the pins collecting area are in parallel to each other, why the SIFT/SURF algorithm is also redundant.
7 Implementation

The vision system is written in OpenCV with C++ inside the Visual Studio 2012 developing environment. The OpenCV library consists of dynamic link libraries which is imported into the Visual Studio environment. The OpenCV library and guides to install it is available through the official website http://www.opencv.org.

7.1 Program test and algorithm verification

The program is written with a pipeline architecture with a program start at the top and steps that needs to be executed before continuing to the next step. A figure of the program flow is shown in Fig. 13.

![Program flow diagram](image)

Figure 13: The vision system program flow

The different states of the program is explained in detail at the following sections.
7.1.1 Fetch image from camera

The systems start with the initialization of the input camera. A 30$ of the shelf web-camera is used for testing the algorithms. After initialization the camera will start fetching images. OpenCV returns an image as a matrix of pixel values, a color image in this case consists of a matrix with 1920x1080 (Full HD) x 3 RGB values. Refer to Fig. 9 for an input image.

7.1.2 HSV filtering + binary threshold

The HSV filter is set up in a way to cancel out the uninteresting features such as noise. It works by setting the pixel values outside the defined HSV values to zero and thereby turning them black. The image is then put into gray-scale and thresholded into a binary image, by applying a threshold to the image. The thresholding turns the gray-scaled pixels below the threshold value to zero into black. Pixels above the threshold are set to one and turned into white. Fig. 14 shows a picture of a filtered and thresholded image. The defined HSV values is set by a trial and error approach where the system integrator can make an initial guess and then fine tune the filter parameters.

![Figure 14: HSV filtered and thresholded image](image)

7.1.3 Mathematical morphology

From Fig. 14, it can be seen that further filters needs to be applied since the terminal pins is not solid. Objects with holes or objects that has multiple contours would likely be categorized as multiple objects by the contour extraction phase as well as the image moment calculation. Applying erosion and dilation to the image will help fill in the holes and make the objects solid. The mathematical morphology filter is adjusted by setting the size of the structuring element, in
this case a rectangle, refer to section 4.6. Figure 15 shows the image after the mathematical morphology filter is applied.

![Figure 15: Image after applying a mathematical morphology filter](image)

7.1.4 Edge detection with Canny

By applying Canny to the filtered image, the contours of the objects is gathered in a vector for further analyzing. Fig. 16 shows the image after contour extraction.

![Figure 16: Image after Canny contour extraction is applied](image)
7.1.5 Calculate image moments

From the contours calculated at the previous step, the image moments is calculated. In analogy with the physical moments of a body, the image moments gives information about the objects area as well as the center of gravity. At this stage, the objects is classified into two groups. One group contains the objects that is considered to be terminal pins and one group of all other objects. The identity of an object is decided through comparison of the 7 HU-moments towards reference HU-moments. Since the HU-moments is considered to be scale invariant, the object is also classified against the object-area which has to lay within a defined span. Refer to section 4.7.3. Only the group of true terminal pins is passed to the next step whereas the false group is used in the graspability analysis step. With figure 9 and figure 15 as reference, it can be seen that the image contains 4 true objects and 2 false objects. See figure 17

![Figure 17: Image with classified objects after HU-moments and area calculation](image)
7.1.6 Perform PCA analysis

Principal component analysis is used to get an estimation of the angle of the true objects. Refer to section 4.7.4. Figure 18 shows the objects with corresponding object-angles (blue lines) after the PCA analysis is done. As can be seen in the picture, the PCA analysis just gives the angle relative +/-90 degrees. Two of the pins has the correct angle, meaning that the blue lines point outwards from the body. It is important to get an object-angle in the +/-180 degree span in order to establish where the terminal pin hat is located.

Figure 18: Demonstration of the object-angles computed by PCA analysis
7.1.7 Template match

From the PCA analysis the object-angle is given in +-90 degrees. As an example, the object-angle is set to 45 degrees by the PCA analysis but the true object-angle could as well be 225 (45+180) degrees. To correct an eventual angle error, template matching is used. After creating a template image that defines the important features of the reference object, the template image is rotated to the same angle as the reference objects PCA-angle. A copy of the template image is also rotated the same degrees as the first template but +180 degrees more. See figure 19 for an example. The template match result is then compared between the two and the highest match has the correct angle. The match difference between the two templates is used to establish how good the template image defines the important features of the object. The bigger the difference the better is the template. The zero angle of the template has to be calibrated against the zero angle of the PCA analysis, since it is a static value it only has to be done once for every new template.

Figure 19: Demonstration of the template match algorithm
7.1.8 Graspability analysis

The graspability analysis is performed to establish whether a terminal pin is pickable or not. It is done by defining the area that the gripper needs to grasp the pins. The grasping area is defined in both size and the center of gravity relative the pins center of gravity. Figure 20 shows the grasping area for all five pins in the image.

![Image of grasping area](image)

Figure 20: Demonstration of the graspability analysis

There are several techniques to determine a collision between objects in an image. This program uses contour extraction with the canny operator to detect a collision. The contour extraction algorithm returns information about how many objects found in the image. The number of detected objects is temporarily saved in a variable, for figure 20 that would be 5 objects. For each of the detected objects, a filled bounding rectangle (the grasping area) is painted over the current object. The contour extraction algorithm runs again and the number of objects detected is compared to the previous result. If the later run gives fewer objects than the first, there is a collision between the objects. A simplified version of the code is shown bellow.

```java
object found_objects = ContourExtraction(original_image);
for each(object obj in found_objects){
    image grasp_image = original_image + bounding_rectangle[obj];
    object found_objects_grasp = ContourExtraction(grasp_image);
    if(found_objects_grasp.size() < found_objects.size()){
        obj.isPickable = false;
    }else{
        obj.isPickable = true;
    }
}
```
A visual demonstration of the algorithm is shown in figure 21. As seen in figure 21, the bounding rectangle melts the two top pins together creating one single object.

Figure 21: Demonstration of the graspability analysis algorithm, the left image has five detectable object whereas the right image has four, concluding that there is a collision between objects.
7.2 Graphical user interface

During the development of the program, important data like the HSV filter parameters and the selection of the template image were hard coded into the program code. To change a parameter, the whole program had to be recompiled. This method made it hard to adjust the filters properly and adapt the system to other working conditions. The program was therefore ported to a C++/CLI environment which has support for creating intuitive graphical user interfaces (GUI). In the GUI, the system integrator could change the parameters of the program without recompiling the whole code. The program also included support for saving the setup parameters until the next time the program is started. That worked with help of XML that saves the configuration in files stored locally on the computer. The system also came with the feature of choosing a default startup configuration which set all of the important parameters directly at startup. Figure 22 shows the program from a users point of view.

![Figure 22: The Graphical User Interface developed in C++/CLI](image-url)
8 Real world adaptations

To put the system into production, it has to be customized to fit the parameters of the environment on which it has to act upon. Given from Robot System Products demo robot cell is the distance between the roof and work area. The camera is attached with a bracket in the roof of the cell and the pins will be placed in the work area. The distance between the two is 1000 millimeters. The following sections provides information about the rest of the customizations.

8.1 Camera

Cheap web cameras introduce radial- and tangential distortions to the image. This is due to cost effective manufacturing processes at the factory. The web camera lens is usually made of low quality plastics which results in a lens with poor optic properties. The manufacturing process also tends to mount the lens not perfectly concentric to the camera sensor which also introduces errors in the image. Another problem with the cheap web cameras is the lack of ability to adjust the parameters of the lens, especially the focus and iris[24].

8.1.1 Selecting a camera

The written vision system runs on a standard PC, why it is benefical to chose a standard USB-3 interface for the communication between the PC and camera. Other alternatives includes Firewire (IEEE-1394), Ethernet (GigE) and cameralink. The other buses would require external equipment to be connected between the PC and camera.

The filtering techniques of the program requires a colored input image, why a color camera is more suitable than a monochrome one.

The resolution of the camera is a bit tricky to decide since the price increases along with higher resolution. Higher resolution also means higher precision, saying that a higher resolution gives a bigger field of view with kept object detection precision. When it comes to cost, standard resolutions is usually cheaper than special ones, why the choice fell on a Full-HD (1920x1080) color camera with USB-3 interface from Basler (acA1920-25uc).

8.1.2 Selecting a lens

Web cameras are often made with wide-angle lenses to catch most of the view and wide-angle lenses tend to bring a fish-eye effect to the image, known as radial distortion. In a computer vision system it might be more suitable to choose a lens with another focal length (zoom) than the wide-angle lens, most due to the distance between the camera and the object to be analyzed. High end cameras
designed for computer vision applications rarely includes the lens, the lens has to be bought considering the parameters of the vision system application. As a guidance when selecting optics for a camera, the following recommendations can be applied[23]:

- A lens with short focal length introduces more radial distortions than a lens with longer focal length.
- A lens with shorter focal length than 12 mm usually introduce notable radial distortion.
- The lens should be selected according to the distance to the object and the required depth of field.

Considered the last of the recommendations, there is a formula to calculate the field of view when some of the camera- and object parameters are known[25]. Refer to figure 23 when dealing with the equation:

\[ Y = Y' \ast \frac{L}{f} \]

Figure 23: Field of view illustration, image from [25]

8.2 Mapping camera pixels to millimeters

For the robot to take advantage of the vision system, the vision system units (pixels) must be mapped into millimeters. The field of view calculated in the last section could be used for that mapping. The ratio between pixels and millimeters is calculated through the following formulas:

\[ \text{Mapping ratio}_X = \frac{\text{Camera resolution}_X}{\text{Field of view}_X} \]

\[ \text{Mapping ratio}_Y = \frac{\text{Camera resolution}_Y}{\text{Field of view}_Y} \]

It is trivial that \( \text{Mapping ratio}_X = \text{Mapping ratio}_Y \) since the aspect ratio is kept at 16:9, this gives the equation:
\[ \text{Mapping ratio} = \frac{\text{Camera resolution}}{\text{Field of view}} \]

From the data sheet of the camera[26] and the known parameters of the system, the following data is gathered:

- Camera resolution: 1920x1080 pixels
- \( L = 1000 \text{ mm} \)
- \( Y' = 4.22 \times 2.38 \text{ mm} \)
- Object size = 10 x 4 mm

Given the equations and the system parameters, the standard focal lengths gives the following field of views, mapping ratios, object sizes and maximum number of detectable objects:

<table>
<thead>
<tr>
<th>Focal length</th>
<th>Field of view</th>
<th>Mapping ratio</th>
<th>Object size</th>
<th>Detectable objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 mm</td>
<td>1055 x 595 mm</td>
<td>1.8 pixels/mm</td>
<td>18 x 7.7 pixels</td>
<td>15 000</td>
</tr>
<tr>
<td>6 mm</td>
<td>703 x 397 mm</td>
<td>2.7 pixels/mm</td>
<td>27 x 10.8 pixels</td>
<td>7 100</td>
</tr>
<tr>
<td>8 mm</td>
<td>528 x 298 mm</td>
<td>3.6 pixels/mm</td>
<td>36 x 14.4 pixels</td>
<td>5 300</td>
</tr>
<tr>
<td>12 mm</td>
<td>352 x 198 mm</td>
<td>5.5 pixels/mm</td>
<td>55 x 22 pixels</td>
<td>1 600</td>
</tr>
<tr>
<td>16 mm</td>
<td>264 x 149 mm</td>
<td>7.3 pixels/mm</td>
<td>73 x 29.2 pixels</td>
<td>980</td>
</tr>
<tr>
<td>25 mm</td>
<td>169 x 95 mm</td>
<td>11.4 pixels/mm</td>
<td>114 x 45.6 pixels</td>
<td>380</td>
</tr>
<tr>
<td>35 mm</td>
<td>121 x 68 mm</td>
<td>15.9 pixels/mm</td>
<td>159 x 63.6 pixels</td>
<td>180</td>
</tr>
</tbody>
</table>

Section 8.1.2 stated that a lens with shorter focal length than 12 mm introduces radial distortions to the system, why the first three lenses is discarded. The 25- and 35 mm lenses gives a quite small field of view which will decrease the number of detectable objects, why they are also discarded. Left is the 12- and 16 mm lenses where the 16 mm lens is safer regarding radial distortion and gives more data (bigger object) to the vision system, why the 16 mm lens is the final choice. Following table gives a summary of the parameters of the system:

- Camera resolution = 1920 x 1080 pixels
- Distance between camera and objects = 1000 mm
- Lens = 16 mm focal length
- Field of view = 264 x 149 mm
- Mapping ratio = 7.3 pixels/mm
9 Complete system testing

The system will be tested using Robot System Products demo robot, the ABB IRB 140 mounted in their demo robot cell. The IRC 5 robot controller will be programmed with the RAPID programming language via the robots FlexPendant. Communication between the robot and vision system is handled by a standard RS-232 serial bus connected to a PC. When the RAPID program has reach a state where it needs object position coordinates it sends a command via the comport to the vision system. The vision system lies in background at the PC and polling for commands. When the vision program has received an instruction to fetch coordinates, it runs the program once and returns the coordinates of the first pickable object in the list as a string over the COM-port. The X, Y and rotation variables is comma separated in the string and the RAPID program converts them into three integer variables. Refer to figure 24 for the outline of the system.

![Figure 24: Outline illustration of the complete system](image)

9.1 Pixel-mapping test

The world coordinate frame of the robot is internally linked to a work object in the robot system. Work objects can be defined in multiple by the program integrator. A suitable work object for this kind of application would be related to the camera field of view.
To verify the calculated pixel-mapping ratio and to define the robots work object, a pointer tool with sharp tip is mounted on the robot. A calibration image, see figure 25, is placed under the camera as a guidance to define the work object.

![Figure 25: The calibration image for work object definition and pixel-mapping test](image)

The black circles is easily detected by the vision system which can output the coordinates in relation to the cameras reference frame. By putting the sharp pointer at the origins of the circles the robot will give the coordinates in reference of the work object, thereafter the pixel-mapping ratios can be calculated and verified. The two circles in the middle of the image is an extra test to verify the calculated ratios. The coordinates gathered from both systems (from left to right) is displayed in the table below.

<table>
<thead>
<tr>
<th>Camera reference frame</th>
<th>Work object reference frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>X: 273</td>
<td>Y: 163</td>
</tr>
<tr>
<td>X: 1596</td>
<td>Y: 159</td>
</tr>
<tr>
<td>X: 705</td>
<td>Y: 515</td>
</tr>
<tr>
<td>X: 1198</td>
<td>Y: 515</td>
</tr>
<tr>
<td>X: 290</td>
<td>Y: 784</td>
</tr>
</tbody>
</table>

| X: -2                  | Y: 0                        |
| X: 151.1               | Y: 0                        |
| X: 48.1                | Y: 40.9                     |
| X: 105.2               | Y: 41.3                     |
| X: 0                   | Y: 71.8                     |

It is obvious to see that there are some measurement errors in the table, the first two Y values 163 and 159 can not be mapped into the same destination value 0. The same goes for number two and three but the other way around, 515 can not be mapped into booth 40.9 and 41.3.
Section 8.1 states that the selected high-end camera and lens produce very little radial distortion. Without the distortion, the mapping could be described by one linear equation for each dimension.

\[ f(x) = kx + m \]

and

\[ f(y) = ky + m \]

where

\[ k = \frac{\Delta \text{Camera points}}{\Delta \text{Work object points}} \]

For any arbitrary related set of points.

From the above equation can be seen that the unit of the answer is pixels/mm, the same unit as the theoretical pixel mapping (7.3 pixels/mm) in section 8.2. That will make the comparison between theoretical and tested mapping ratio easier. Using data from the table gives the following equations:

\[ f(x) = \frac{1596-273}{151-(-2)} x + m = 8.63x + m \]

\[ f(y) = \frac{784-163}{71.8-0} y + m = 8.65y + m \]

The difference between the theoretical and the tested mapping ratio is likely due to inaccurate measurement between the camera and objects (L = 1000 mm) in section 8.2. Based on a mapping ratio of 8.63 pixels/mm, the distance L = 840 mm.

The constant m is calculated with the known parameters:

\[ 290 = 8.63 \times 0 + m \Rightarrow f(x) = 8.63x + 290 \]

\[ 163 = 8.65 \times 0 + m \Rightarrow f(y) = 8.65y + 163 \]

The m constant in the equation is nonzero since the origins of the two reference frames not coincides.

The equations is inverted to give an output of mm/pixel, which is the useful equation for the system. The yielded mapping equations is:

\[ f(x) = \frac{x-290}{8.63} \]

\[ f(y) = \frac{y-163}{8.65} \]
A vision system has been developed in C++ with the OpenCV libraries. Different strategies of attacking the problem has been evaluated, SIFT/SURF gave bad results when dealing with multiple instances of the same object and the trained HAAR-classifier gave a lot of false positive results. The image moment and template match algorithms where able to correctly classify and orient the terminal pins given the conditions stated bellow:

- The incoming light is kept at the same level as when the system was calibrated.
- The light is evenly distributed over the whole field of view.
- Shadows and spotlight is kept at minimum level, preferably none of each.
- The terminal pins is distributed or arranged with a bit of space in between.

The calibration error presented in section 9.1 is most likely due to the precision when the robot pointer tool where addressed at the origins of the calibration circles. Some errors in mapping is also caused because of the work object of the robot is not perfectly in parallel with the camera reference frame.

Due to time constraints, different gripping technologies have not been evaluated. Instead is the system prepared to be equipped with a gripper as far as possible. Also is the design of the in- and outflow of plastic bodies not designed and evaluated. The work with distributing the terminal pins for the vision system automatically is also not considered in the report due to lack of time. The system critical arrangement of pins is suggested to be handled by a human hand, since no solution to that problem where derived.
11 Future work

The future work of this project is related to what is stated in the last section and the problem formulation in section 3. If time is found for further research and engineering, the following problems is addressed by this report:

11.1 Pin distribution

The goal is to derive a solution able to distribute the terminal pins in a way that the vision system can recognize them and the gripper can grasp them. Different solutions available at the market includes vibration feeders and conveyor belts. The off the shelf solutions is quite expensive why some own solution with a couple of engineering hours might be more preferable.

11.2 Design in- and outflow of components

For an effective solution, the problem with in- and outflow of booth terminal pins and plastic bodies has to be addressed. The system may even be equipped with jigs for different plastic bodies which will require some sort of safety installation. If the system will be able to handle different bodies, some mechanism to determine the present work body needs to be developed. Either the system integrator writes different programs for different types of bodies and the end user have to choose the right program for that body. A more stable solution would be to let the robot make that decision. The robot could with help of some sensors determine which kind of body that is currently being processed.

11.3 Decrease the systems sensitivity to light invariances

Different of the shelf solutions is presented on the market, including color filters for different light spectras. If the camera is equipped with some color filter, the lightning might have to be adopted to amplify the light in that color spectra. There is also a possibility to equip the collecting area with back-lightning, preferably with a different light spectras than the light that highlights the pins. Improving the sensitivity in a software way includes software implemented filters that could decrease the systems sensitivity to light invariances.
References


[23] Ahlgren, Per, Robot System Products, interview 2015-03-25

