Object detection and pose estimation of randomly organized objects for a robotic bin picking system

In cooperation with BTH and ThyssenKrupp System Engineering

Authors:
Tomasz Skalski
Witold Zaborowski

Blekinge Institute of Technology
School of Engineering
Department of Applied Signal Processing

Supervisor:  Dr. Nedelko Grbic
Examiner:   Dr. Sven Johansson

29.06.2012
Abstract

Today modern industry systems are almost fully automated. The high requirements regarding speed, flexibility, precision and reliability makes it in some cases very difficult to create. One of the most willingly researched solution to solve many processes without human influence is bin-picking. Bin picking is a very complex process which integrates devices such as: robotic grasping arm, vision system, collision avoidance algorithms and many others. This paper describes the creation of a vision system - the most important part of the whole bin-picking system. Authors propose a model-based solution for estimating a best pick-up candidate position and orientation. In this method database is created from 3D CAD model, compared with processed image from the 3D scanner. Paper widely describes database creation from 3D STL model, Sick IVP 3D scanner configuration and creation of the comparing algorithm based on autocorrelation function and morphological operators.

The results shows that proposed solution is universal, time efficient, robust and gives opportunities for further work.
Contents

Abstract .................................................................................................................................................. 3
1. Introduction ......................................................................................................................................... 6
  1.1 What is bin-picking ......................................................................................................................... 6
  1.2 Goal of the thesis ............................................................................................................................. 7
  1.3 Motivation ....................................................................................................................................... 8
  1.4 Concept of work ............................................................................................................................... 8
  1.5 Thesis outline .................................................................................................................................. 9
  1.6 Expectations regarding final results ............................................................................................... 11
2. Types of bin-picking .............................................................................................................................. 12
  2.1 Feature – based approach .................................................................................................................. 12
  2.2 Model – based approach ................................................................................................................... 13
  2.3 Examples of used approaches .......................................................................................................... 13
3. Solution ............................................................................................................................................... 15
  3.1 Scanner ........................................................................................................................................... 15
  3.2 Database .......................................................................................................................................... 26
  3.3 Comparing Algorithm ...................................................................................................................... 35
  3.4 Pose estimation ............................................................................................................................... 42
4. Results ................................................................................................................................................. 52
  4.1 Test 1: ............................................................................................................................................ 52
  4.2 Test 2: ............................................................................................................................................ 57
  4.3 Summary of the results ..................................................................................................................... 60
5. Conclusion ........................................................................................................................................... 61
  5.1 Future work ...................................................................................................................................... 61
  Content of figures ................................................................................................................................. 63
1. Introduction

1.1 What is bin-picking

Today modern industry systems are almost fully automated. The high requirements regarding speed, flexibility, precision and reliability make it in some cases very difficult to create. Solutions for tasks such as packing, feeding machines with complex parts, sorting still possesses many challenges and are willingly researched and upgraded. These processes are considered as really hard because to solve them many algorithms and devices must interact with each other, giving as a result one compact, robust and functional solution. One of the most widely used solution in past 30 years to solve such problems is bin-picking system - considered as the most complex sorting task used in industry and as a “last piece to achieve fully automated independent process” [1][2][4][6]. It is highly researched since 1980, however not yet fully automated. Creation of such system would give the opportunity to eliminate the human factor from monotonous and dangerous work. It would also give the benefits regarding financial factor, since the machine can work for 24h/7. Hence creation of such system is on really high interest of the industry field. Fig. 1 shows an illustration of a simple bin-picking system.

But what exactly is so called bin-picking? It is an application which integrates robotic grasping arm with vision system and computer: Robot with grasping arm, cameras, object localization and recognition algorithms, collision avoidance algorithms and many other. There are many different solutions for bin-picking systems but general idea of work is as follows: Vision system conjugated with computer looks for a previously known object in a bin, calculate its position and collision paths with a bin, sends the coordinates of the picking candidate and how to grasp it to the robotic arm. The last part of the process is sending the grasped object to the defined localization. All of these operations have to be done with big precision, short enough time and safely, since that are the most desirable factors in industry. That is why creation of functional and viable bin-picking system is such a big challenge. Functional solutions already exist and are used in industry, but there are not yet perfect. In most cases they work only for one type of the object at the time. Bin-picking system can be divided into a few main parts, presented on the Fig. 2.
1.2 Goal of the thesis

The main goal of this thesis is to provide a functional solution for a vision part of a bin-picking system (object localization and recognition). Vision part is by far the most important part in bin-picking system since it serves the role of “eyes” for the robot. It enables the robot to recognize surrounding work environment. Without vision system it would not be possible to localize the object and consequently – obtain a pick candidate. Hence good vision system is considered as a key issue in the whole bin-picking. The most important target is to
create an algorithm which is invariant from distortions, illuminations and has high percentage of recognition success rate. The whole work had to be done using Sick 3D scanner and Matlab environment [7].

1.3 Motivation

Over last few years the computational power is growing exponentially what allowed to implement bolder solutions in the automation field. One of the most important issues in industry is to reduce the aspect of human intervention by improving the quality, precision and necessary resources.

ThyssenKrupp System Engineering in Bremen, Germany is a company which works in the automotive industry for over 60 years and employees over 2000 people worldwide. The product range of TKSE is to invent, create and manage assembly systems for engines and transmissions [8]. Since the automation in such field is a very important and desirable issue, there exist constant necessity to improve it. As was mentioned before, one of the last element which is not fully automated until now is the bin – picking problem. Hence the research department of the company decided to investigate it, because it could be very beneficial in future.

Our motivation was to satisfy the expectations of the company and to present a wide knowledge about already existing solutions. On the other hand we wanted to make a deep research about Bin – Picking system and create a functional fundament and framework for the future work. In the company we had access to the professional equipment like 3D scanners and smart cameras what gave us the opportunity to make our work more realistic than pure software implementation.

1.4 Concept of work

The concept of this thesis was to create the functional object recognition and localization algorithm with the high percentage of success rate for a bin-picking system. Before the work could be started the wide knowledge regarding the bin-picking systems, common solutions and information related to pose estimation issues has to be acquired. Two methods have been taken for consideration. The first method based on feature extraction of
the object in the 3D space by using a stereo vision system. The second one was based on processing raw data from a 3D scanner and compare it with a database in which the object is rotated in different positions[1][2][3][4]. Both methods are explained more detailed in chapter 2. The choice between these two methods was made by comparing possibilities, difficulties and constraints. Finally it was decided to proceed first with the second method since the 3D scanning is a newer technology, but later also a feature extraction of the object is applied. That gives wider spectrum of further modifications and is more flexible.

The idea of work was as follows: Creating the pose reference database using CAD model in Matlab environment, capturing and processing images from the scanner, implementing algorithm for comparing images, localization the best pick candidate to pick, printing out coordinates of the object for a robot. All of these parts will be widely described and presented in the later parts of the thesis.

An addition condition to this work is that it did not handle bolts which are perpendicular to the scanner.

The target of this approach was M8 x 16 bolts. The top and side view of the bolt is presented on Fig. 3 and 4. The diameter of the bolt is 8 mm and the length of the pin 16 mm. The whole bolt is 24 mm long.

1.5 Thesis outline

The Thesis is divided into five chapters. The first chapter is an introduction to the bin – picking approach, our motivation and the concept of the work. In the second chapter two types of bin – picking solving and examples of used approaches were introduced. The third chapter explains the solution of our approach. How the scanner was calibrated and what function it had for the solution, details regarding creating database, precise explanation of the comparing algorithm and our method for object localization and pose estimation of randomly organized bolts. The fourth chapter shows the results of our algorithms and finally chapter five concludes our work.
Fig. 3 Top view of the bolt
Fig. 4 Side view of the bolt

Fig. 5 Basic chart describing the idea of work

Creation of the pose reference database using CAD model

Programming 3D scanner, capturing and importing images into MATLAB environment

Implementation of object recognition and localization algorithm

Visualization of the results and best picking candidate.
1.6 Expectations regarding final results

While implementing any application, creators have always to keep in mind a few key issues and constraints which determines the possibilities of work. There is also need of creating application which does not deviate from the top already existing solutions. In this thesis the following factors have been taken for consideration as the most important:

- Algorithm invariant to illuminations, shadows, distortions
- Short computation time
- High percent of success rate, high accuracy
- Deep knowledge regarding bin-picking systems
2. Types of bin-picking

The Bin – Picking problem as mentioned in chapter 1 has to solve the problem of sorting randomly organized objects from a bin. This objects are often called bulk-material which is shown on Fig. 6, 7 and 8.

Fig. 6 Bak Harpen \hspace{0.5cm} \textbf{Fig. 7} Discs \hspace{0.5cm} \textbf{Fig. 8} Stator Housings

Over the last few years there had been investigate many solutions for the “Bin – Picking problem”. The most common methods can be divided into two approaches.

The first one is Feature-based and the second one is the Model-based method. In the following part both approaches will be described – how they work, advantages and disadvantages and also some examples of existing solutions for both methods.

2.1 Feature – based approach

Feature-based approach is specified on the uniqueness of every part [12]. Every part can be defined by different features which represent them, their structure, type and spatial relation. This approach restricts the problem to distinguishing the main features which characterize the object and to recognize how this features changes by rotating over its own point of gravity in the 3D area. The features could be corners, bores, holes or other specific parts of the object. For example a round hole in an object becomes an ellipse if it is deflected from the point of view of the camera. The relation between two corners can change if the object is rotated from a defined zero position.

This approach needs two cameras. One camera can just give information about the x- and y-coordinates in a plane. The z-coordinate which is the height of the object in a 3D space has to be determined by the second camera using triangulation.
The advantages of such an approach is that the modern image processing allowed to precise and high accurate methods of feature extraction. The image-processing tools available for different programming areas are huge so most of the algorithms did not has to be written by our self but can be implemented directly in the program.

The disadvantages are, that every part need a different solution by itself, depending on what type of part it is because of the difference of features. Another disadvantage is that, this approach had to be made by two high resolution cameras which are sensitive for illumination and occlusion so the lighting has a big influence.

2.2 Model – based approach

Model-based approach is based on creating a database of an object which is the target of the Bin-Picking application, and compare it with a real range image delivered from a 3D scanner [1]. The database is a set of range images of the object rotated in a defined step. It has to cover nearly all possible positions of the object rotated over its own point of gravity in the 3D space (all three axes x-, y- and z-). As only the surface and its depth values in the z-axis are saved in the database, the feature extraction becomes unnecessary.

As mentioned above for this type of realization a 3D scanner is used which captures depth values of the searching area using the triangulation system. Such sensors/cameras achieve better accuracy compared to stereo vision systems with two cameras.

A huge advantages of that approach is the widespread possibility of implementation. A database could be made of nearly all kinds of objects without any feature extracting.

The disadvantage is, that without high efficient hardware the comparison between the database which can be made by thousands of images and the original image will cost high computation time.

2.3 Examples of used approaches

The solution described by [1] is built on the Model – based approach. They split the problem into three stages.
The first is to create offline a reference pose database, built on a 3D CAD model. The model is orthogonally projected into a range map. The resolution of one image stored in the database have to be the same as the input image which is available from scanner manufacture. The second part is a rough pose estimation algorithm which defines the course location of the object. This algorithms contains of two error functions. One is the error between the depth values in the input range image and the reference images in the database, the other error function compares the Euclidean distance transform of the 2D image of real scene and the database. They compute all error function in parallel by using a Graphic Process Unit (GPU) which is 30 times faster than the usage of conventional CPU. The last part is the fine pose estimation which refine the position in the second part. The rough pose estimation is in fact the input for the refinement. This is done using an Iterative Closest Point algorithm (ICP) which compares vertexes from the CAD model with a partial mesh model from the original image to reduce the remaining error.

Another solution described by [3], also built on the Model – based approach. Boehnke creates a virtual range image by simulating a virtual 3D sensor and a virtual scene. Based on this a database of possible positions of the object is created. The next step is to compare the virtual range image with the real range image by using a correlation function. The highest peak of the correlation will be the one with the best match from the database. This is used as an input for the pose refinement algorithm which is handled like [Park, Germann, Breitenstein and Pfister, 2007 & 2010] by using an ICP algorithm.

A different approach is used in [9] where a Feature – based method is implemented for the vision part of a Bin – Picking application. This paper focused on the feature extraction of a specific part – Stator Housing. They are searching for the opening part of this object which is a round hole. In the 3D space it becomes an ellipse depending on the position. The pose could be determined by the elliptic projection in two different cameras. Therefore the edges of every ellipse had to be segmented and grouped and then the pose of the ellipse is estimated.
3. Solution

3.1 Scanner

This chapter begins with a short introduction about 3D scanning and the idea to use this approach in this thesis. Next the workplace will be presented. Then the programming method of the scanner will be presented. Then follow a few images from the scanner and the settings of the scanner.

Capturing good images from vision system is one of the key issue in the whole bin-picking system. Making image with appropriate size, quality causes that further processing are much more easier and more effective. On the other hand, setting good camera parameters are also very important while that allow to eliminate most of distortions which can cause ambiguity in matches. Since the quality of the image has a huge impact on the later comparing algorithm there was a need to choose the device which would be able to make appropriate images. The most common approaches in capturing images in bin-picking systems are 3D scanning and system of two stereo cameras. Since 3D scanning is newer technology and there was a possibility to implement it in the company it was decided to proceed with the SICK IVC-3D scanner. This is a device which base on a unique CMOS chip. It enables to capture 3D images really fast and with high accuracy. This scanner is considered as the “first smart camera in the world that is designed to inspect and measure in three dimensions”[7]. User interface of the scanner enables many operations, divided into categories such as: image processing, communication, regions of interests. Many tools from these categories can be used to enhance the quality of the image or to apply a specific operation to the image for example. edges or averaging filter. There is also a possibility of programming the scanner, which allows to make more sophisticated operations on images. Measurement can be done with two units – pixels or millimeters. That is very comfortable because it is easy to set the desired scale or to check the size of a single object for further adjustments with database. Easy encoder connection enables to capture the images with the good quality in the repeatable way. All of these benefits made the use of the scanner really enjoyable and simple.
Fig. 9  *SICK IVP-3D scanner used to capture real images.* [6]

The scanner consists of the camera and a laser. The idea of work is as follows: The laser draws a perpendicular line to the surface placed below. The camera is searching for a laser beam from an angle. When a laser slices any object the camera measures the distance between each scanned profile and the camera. Then CMOS chip transform these distances into heights.

The number of profiles can be defined in the software. That is very important factor since quality and size of the image depends on it. As the number of profiles are bigger, the scanning is more accurate. On the other hand too big number of profiles can cause distortions in the image. After the height of each profile is acquired the 3D triangulation is performed to create a 3D raw image. The factors like: field of view, resolution, scale, distance from the camera can be set in attached software. Only good adjustment of these settings allow to ensure good quality of the image.

Fig. 10  *Example of scanning an object* [6]
After the 3D image is created it can be saved in a “bank” for visualization or processing. All tools available in the software are made on image banks.

To be able to capture 3D images the workplace has been built. For this purpose the minitec construction has been used. It is very robust, light and in common use in industry so the choice regarding material was quite simple. In this workplace three most important elements can be distinguished: 3D scanner, encoder and a sliding construction. All of these parts are necessary and essential in later work. First of all the general frame with the sliding construction has been built. That was required to move the scanner along the specified axis. The important fact is that the whole construction had to be very rigid. Otherwise captured image would be very distorted or inaccurate. In the next step of creating the workplace scanner has been installed. The scanner has been installed at the top of the sliding construction in the way that regulation of the distance between the scanner and a planar surface was possible. It was really important since the beam of laser must be perpendicular to the surface. The last very important part is encoder. In our case encoder is a programmable device which allows to ensure good quality of the image and to adjust the desired distance of scanning.

Fig. 11 Side view of the workplace
After the workplace has been constructed, all devices have had to be integrated. In this purpose all parts have been connected as follows:

Fig. 13  Diagram presenting the connections between devices
At first connections between the devices and the power module have been set. Both encoder and scanner uses the AC/DC converter 90..260 V AC into 24V DC. This connection was really important since any mistake could cause permanent device damage. Next the encoder was connected into scanner and programmed. Programming of the encoder was really simple since the only value which had to be set was number of impulses. To be sure how many impulses per mm calculates the encoder, the diameter of the moving part of the encoder has been calculated and the number of impulses has been preset to maximum - 8000.

Fig. 14 Encoder used in a workplace

From the following calculations it was possible to obtain the impulses/mm value:

\[
d = 29.3 \text{ mm} \\
\pi \times d = 92,002 \\
\text{Imp} = 8000 \\
\frac{\text{Imp}}{\text{mm}} = \frac{8000}{92.002} = 86.95
\]

Where, \(d\) is a diameter of the moving part, \(\text{Imp}\) is the number of impulses

The number of impulses per mm had to be calculated, since this value is later used to adjust the distance of scanning.

Next step was to enable communication between scanner and PC. For this purpose switch has been used as a connecting device. By setting the appropriate IP addresses of the scanner and
ThyssenKrupp System Engineering

PC the communication was obtained via Ethernet. When all modules have been properly connected to the power supply and configured, the workplace creation has been finished.

Fig. 15  Real image of the workplace

Fig. 16  Real image of the workplace from different view
The idea of work of such created workplace is as follows: First of all the scanner software has been configured. The most important values to set were:

1) Pulses /mm, constant value calculated as 86,916
2) Profile distance
3) Pulses per profile
4) Number of profiles

The first value is already calculated by using the encoder and some simple calculations. The second value must be given by user in [mm] in the scanner software. By changing this value user can preset the mm to pixels ratio. This is very important value since it’s used for scaling purposes. **Example:** If profile distance is equal to 0,5 it means that one mm of scanning is equal to two pixels in the image. Third value is the result of multiplying 1) and 2). For the values calculated above it would be:

\[ \text{Pulses per profile} = 86,916 \times 0,5 = 43,458 \]

The fourth value defines the number of profiles. By changing this value users are enabled to adjust the scanning distance. Scanning distance in [mm] is the result of multiplication 2) and 4). For the values calculated above and value 4) preset to 200 it would be:

\[ \text{Distance of scanning} = 200 \times 0,5 = 100 \text{ mm}, \text{ which is equal to 200 pixels.} \]

Table 1. Some important values to set in the scanner
It is very important to properly adjust values from the Table 1 since even small change can drastically degrade the quality of the image or to introduce unwanted distortions.

After the configuration step it was possible to acquire first images from the scanner. For the purpose of this thesis moving of the sliding construction was performed manually with one quick move. The image was created after the laser crossed the preset distance of scanning.
ThyssenKrupp System Engineering

Fig. 18  Example of the image from the scanner with smoothing filter applied

It is also very important to set the appropriate scale of the object in the scanner (profile distance value) since it must be the same as the objects in the database. Only then the comparison algorithm can give the correct results.

The image shown in Fig. 18 is thresholded. That means the range of pixel values is in the interval 0-1. The highest values are those for whom the distance from camera is the lowest. These values are marked with white color. Values marked with black color can be either surface or missing data. The original image in the matrix form looks as follows (without thresholding):

| 133.5648 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 132.8506 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 131.8056 | 129.5166 | 129.5166 | 129.5166 | 129.5166 | 130.1069 | 130.1069 | 130.1069 | 130.1069 | 130.1069 |
| 132.0038 | 129.5166 | 129.5166 | 129.5166 | 129.5166 | 130.4520 | 130.4520 | 131.3874 | 131.3874 | 131.3874 |
| 132.0038 | 125.8166 | 125.8166 | 125.8166 | 125.8166 | 130.4520 | 130.4520 | 132.0038 | 132.0038 | 132.0038 |
| 0 | 125.8166 | 125.8166 | 125.8166 | 125.8166 | 130.4520 | 130.4520 | 132.0038 | 132.0038 | 132.0038 |
| 0 | 0 | 125.8166 | 125.8166 | 125.8166 | 125.8166 | 125.8166 | 130.4520 | 130.4520 | 130.4520 |
| 0 | 0 | 0 | 125.8166 | 125.8166 | 125.8166 | 125.8166 | 125.8166 | 130.4520 | 130.4520 |
| 0 | 0 | 0 | 0 | 125.8166 | 125.8166 | 125.8166 | 125.8166 | 125.8166 | 130.4520 |
| 0 | 0 | 0 | 0 | 0 | 125.8166 | 125.8166 | 125.8166 | 125.8166 | 125.8166 |
| 0 | 0 | 0 | 0 | 0 | 0 | 125.8166 | 125.8166 | 125.8166 | 125.8166 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 125.8166 | 125.8166 | 125.8166 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 125.8166 | 125.8166 |

Fig. 19  Small fragment of the raw data image
Each pixel of image shown above has its own depth value. These values are described as height of the object in [mm]. This means that the lowest value is the farthest from the camera. Having such raw data enables to execute many image processing operations which will be described in the later parts of the thesis.

As a result of each scanning, raw data along with the highest value of an image is obtained. The image is naturally required for the input of comparing algorithm and the highest value is very important while processing raw data.

Even when the configuration of the scanning parameters has been made in proper way, there are still chances of making images with distortions or illuminations. These two factors are highly unwanted so to counteract them an algorithm has been written. Using it allows to get the real highest point of the object instead of height of random illumination.

Fig. 20 3D visualization from the scanner
In this algorithm we are using the knowledge about distortions and illuminations. Illuminations always appear as small groups of pixels with really big height in comparison to the rest of the image (illuminations are shown with red circles on the Fig. 20). Since it is possible to count a group of pixels with defined height in the image it is possible to distinguish illuminations. The idea of work of this algorithm is as follows: In each iteration virtual surface is defined on the image. The height of this surface is the same as height of the highest object in the image. In the next step, number of pixels sliced by this surface is calculated. Observations and tests showed that the number of pixels of illuminations never exceeds 100. Since this information is possessed all pixels at the current height can be counted. If the number of pixels at current height is smaller than 100 then in the next iteration height of surface is decreased by one. This process is continued until the number of pixels exceeds 100. As a result of this algorithm pixels with the real highest values are marked and this highest value is saved into a table. Then it is possible to send it into Matlab software.

Fig. 21 Result of implemented algorithm
3.2 Database

In this chapter first will be shown an overview about the database. Next the idea of a STL Model and how to import such a Model to Matlab will be presented. An imported part of are the operations on triangles, rotation matrices and scaling which will be also mentioned in this chapter. Next an explanation of extraction of depth values will follow. In the end the visualization of the database will be shown.

In this thesis it was decided to proceed with model based method. Since the creation of pose reference database is the essence of this method, it was required to create such database. Database is a set of images of a single object under different orientations stored in organized way in a memory. It can consist practically of unlimited number of different poses. To the most important advantages of using the database are:

1) It can be created for any type of object for which CAD model is available
2) While changing the object in a bin there is no need to change the whole algorithm like in feature based method.
3) Database is invariant from distortions and illuminations since images are virtual
4) It is very flexible and recently became the most frequently used solution
5) Computation time of the database is irrelevant since it can be created in offline process

The only constraint in creating the database is hardware capability. Database is a key issue in the whole bin-picking system hence the success to a large extent depends on its accuracy of implementation. Database is so important because it is major component in the object localization & recognition algorithm. Creation of such database is not easy task since the raw images in the database have to be identical with the images from the vision system. The task how to make this data the same is considered as one of the hardest in the whole bin-picking system. In this thesis creation of the database is divided into following steps:

1) Importing geometry of the CAD model saved as STL extension into Matlab software
2) Extracting the depth values of the object
3) Filling model with additional points
4) Rotation and scaling operations on the model
5) Saving each rotated pose into a large matrix of demanded size
6) Visualization of the results
First step in creation the database was importing CAD model into Matlab. Since the Matlab does not have any built-in tool for that the information’s regarding this problem has to be acquired. The only available solution at this moment to perform such importing is based on STL models.

STL models are a representation of CAD models, described by faces and vertices. “STL files describe only the surface geometry of a three dimensional object without any representation of color, texture or other common CAD model attributes. The STL format specifies both ASCII and binary representations. Binary files are more common, since they are more compact” [Wikipedia]. The structure of every STL model is as follows:

```
facet normal n_i n_j n_k
outer loop
  vertex v_1_x v_1_y v_1_z
  vertex v_2_x v_2_y v_2_z
  vertex v_3_x v_3_y v_3_z
endloop
endfacet
```

This structure consist of faces and vertices. Each face(triangle) has 3 vertices and each vertex is described by three coordinates - x, y, z. The program works in the way, that for each face it reads the corresponding vertices and then saves coordinates of these vertices into three column matrix. Such created cloud of points can be then connected and visualized by Matlab patch command.
Fig. 23  Model of bolt imported into Matlab software and example of vertices matrix

Fig. 24  Bolt with corresponding vertices description - frontview
While model is imported into Matlab it is possible to process it (all operations has been done on vertices). First of all scaling program has been implemented. Since the original STL model is really small by scaling it is possible to increase the distances between each vertices by a constant value. The scaling is just multiplication of vertex matrix by demanded scaling factor. Scaling is very important for further processing and for adjustments with the scanner images. In this thesis, object is scaled with scale factor equal to 2.

The biggest problems with STL models is that they are described only by limited number of vertices (as can be seen on Fig. 24, 25). Even when the depth values were extracted and saved into a matrix, the object was really poorly described. It was not suitable for further comparing with image from the scanner. Hence the algorithm of adding points into the STL model has been written. As was already mentioned each triangle is described by 3 vertices. Since the coordinates of each vertex are known, calculating the length of each side of the triangle is not a problem. When these distances are already known any defined number of points can be added between two vertices at regular intervals. After this operation each of point can be connected with its equivalent (first point on one side with first point at 2nd side etc.). Moreover, on the lines created in that way, additional points can be added. The idea of work of this algorithm is presented on Fig. 26.
Fig. 26 Triangle with new points.

With red circles marked original points, with green squares marked new points. It was decided to not add points on lines on this particular image to make it readable. However in a program these points are added.

Such operation has been made for each triangle in STL model. That allowed to fill the whole object with a new points. Very important fact was to keep proper depth values. That was possible since each coordinates changes in linear relationship. After implementing this algorithm the object was ready for putting it into the database.

After the scaled object is filled with additional points, one of the last things to do regarding processing is to change the coordinate system and to round values. First the center of gravity of object before rotating is calculated. Then simple program has been written for searching minimum vertex in $x$ and $y$ directions from the center of mass. After adding these values to the vertices matrix it was possible to shift all the values into the positive side of coordinate system.

The shifting had to be done to enable further database creation, since in Matlab user can’t refer to the cell with negative values. After this operation it was possible to introduce rotation operators. Rotation is very important since it enables to obtain primary object in any possible orientation. These objects are then saved in one large matrix as a database. Rotation involves multiplying rotation matrix with the current object matrix (with filled points). As a result of such multiplication rotated object is obtained. The rotation matrices has been made on the base of fundamental linear algebra rights. Rotation matrices has been created in the following way:
\[
R_x(\alpha) = \begin{pmatrix}
1 & 0 & 0 \\
0 & c & -s \\
0 & s & c
\end{pmatrix}
\]
\[
R_z(\alpha) = \begin{pmatrix}
c & -s & 0 \\
s & c & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

Where \( c = \cos(\alpha), s = \sin(\alpha) \), \( \alpha \) is rotation degree set in degrees

Since on the purpose of this thesis object symmetric in y axis was used, there was no need of calculating Ry rotation matrix.

At every iteration shifted to the positive coordinate system vertices matrix with added points is multiplied by these rotation matrixes. To create enough objects for further comparing purposes the rotation has been done in the following way:

Since only two axes had to be rotated, two rotating degrees were set. In this thesis it was decided to rotate x axis to 360 degrees, capturing current object position every 10 degrees. Rotation of z axis was set to 180 degrees with incrementation every 10 degrees. The incrementation of z degree was followed only when for its current value all rotations around x axis were finished. Simple example presenting the idea of work of can be found in Table 1:

<table>
<thead>
<tr>
<th>X rotation value in degrees</th>
<th>10,20,30,40,....,360</th>
<th>10,20,30,40,....,360</th>
<th>10,20,30,40,....,360</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corresponding Z rotation value</td>
<td>0</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

The process is repeated as long as z rotation value reach its set maximum. For the example shown in the table 1 number of created object is equal to 720. Changing rotation degree from 10 to 5 would result in creating 2600 images.

While the rotation algorithm was created there was a need to store each pose with its corresponding depth values. Depth values in STL model are represented by z axis.
scaling and shifting the object into the positive coordinate system it was possible to extract depth values. For this purpose the algorithm has been written. The idea of this algorithm was to capture each depth value from vertices matrix and to save them into a matrix in the appropriate position. This position was described by corresponding x and y values. To be able to do that the rounding of the x and y values has been done since the matrix indexes can be only positive values. A simple example to bring the idea of work of this algorithm is presented below.

Table 3. Example of saving depth values into the matrix

<table>
<thead>
<tr>
<th>Rounded X value</th>
<th>Rounded Y value</th>
<th>Z value</th>
<th>Position of the depth pixel in a matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>15</td>
<td>17,4</td>
<td>(15,10)</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>4,6</td>
<td>(8,2)</td>
</tr>
</tbody>
</table>

After all pixels are added into a matrix, creation of the object is finished. Then the matrix is extended to make place for new objects. This process is continued until all poses are created and saved.
Scaling and shifting the object into positive coordinate system

Adding points into vertices matrix

Extracting depth values from the object shown on the Image 2

Saving the depth map into the 64x64 matrix

Extending the size of the matrix

Rotating the object by multiplying proper rotation matrices with initial depth map

---

**Fig. 27**  General scheme of the algorithm used to create the database

---

**Fig. 28**  Small part of pose reference database
The results of creation the database are shown on the images below.

**Fig. 29** Small part of the database with random rotated objects

**Fig. 30** Database created from points described by pure STL model

The results of creation the database are shown on the images below.

**Fig. 31** Part of the database created for other object

Such created databases allows further comparing with real image from the scanner. Since the size of huge database is known it is possible to refer to any image in the database. Database is required to determine the position of the pick-candidates. That is why it is one of the most important if not the most important part in the whole approach. Hence good its preparation was a key to success.
3.3 Comparing Algorithm

Since the input parameters of the system are known, the next crucial step in this approach is to define the comparing algorithm.

There are many possibilities to do this. The paper by [1] used two error functions, first to compute the cover error, the second to compute the depth error. This gives a rough pose estimation of the object and an initial value for the refinement which is done by using the ICP algorithm. On the other hand [3] used the normalized cross-correlation to compute the correlation between a virtual range image with the real range image.

The algorithm used in this paper also based on the simple cross – correlation of two images. The cross correlation is an common approach used in the field of signal processing.

“In signal processing, cross-correlation is a measure of similarity of two waveforms as a function of a time-lag applied to one of them. This is also known as a sliding dot product or sliding inner-product. It is commonly used for searching a long-signal for a shorter, known feature. It also has applications in pattern recognition, single particle analysis, electron tomographic averaging, cryptanalysis, and neurophysiology.” [13]

According to the definition above, the formulation of the problem for an image instead of a one dimension signal is as follows.

Two input arguments are required for the correlation algorithm. One is the template which is the “shorter-signal” in terms of the definition. The original image in which we are looking for the template is the “long-signal”. Using a sliding-window the template has to be slide over the whole original image. The sliding-window is moved by one pixel per iteration. The number of iterations can be calculated knowing the size of the original image and the template. If the size of the original image is $X \times Y$ and the template $x \times y$, then the number of iterations $N_i$ will be equal:

$$N_i = (X - x + 1) \times (Y - y + 1)$$
For every iteration the function has to calculate the percent how good the template match the original image. In this thesis an already existing function from the Matlab-Toolbox “normxcorr2” is used.

“C = normxcorr2(template, A) computes the normalized cross-correlation of the matrices template and A. The matrix A must be larger than the matrix template for the normalization to be meaningful. The values of template cannot all be the same. The resulting matrix C contains the correlation coefficients, which can range in value from -1.0 to 1.0. [Mathworks – Normalized 2D cross correlation]

The algorithms “normxcorr2” uses the following general procedure [10] [11]:

1. Calculate cross-correlation in the spatial or the frequency domain, depending on size of images.
2. Calculate local sums by pre-computing running sums.
3. Use local sums to normalize the cross-correlation to get correlation coefficients.
The implementation closely follows following formula from:

\[
\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}][\bar{t}(x-u,y-v) - \bar{t}]}{\left(\sum_{x,y}[f(x,y) - \bar{f}_{u,v}]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2\right)^{0.5}}
\]

Where:

- \( f \) is the image.
- \( \bar{t} \) is the mean of the template
- \( \bar{f}_{u,v} \) is the mean of \( f(x,y) \) in the region under the template.

The description above is taken from the Mathworks homepage [Mathworks – Normalized 2D cross correlation].

The next step show an example of the correlation function used to find a simple bolt in the real range image taken from the scanner (Fig. 33).

*Fig. 33*  Example of a bolt made by the 3D scanner and the result of the best match (red dot)

*Fig. 34*  Template from database

The template is an image from the database (shown on Fig. 34) which shows nearly the same position as the bolt in the real range image from the scanner. Using the cross correlation the position where the template correlates the best match, could be determined, based on the plot which is delivered by the “normxcorr” function (Fig. 35).
The highest peak determines the coordinates in the image from the scanner where the template has the highest correlation (in percent). The dot is positioned in the middle of the template. So if the red dot is on the middle of the bolt which we are looking for in the real range image – the match is perfect. The result of the best correlation was ~84%.

The next example show the same image from the scanner but the template had another rotation so theoretically it should not fit the bolt as good as the best match on Fig. 37.

Fig. 35  Plot of the Matlab normalized cross-correlation function

Fig. 36  Example of a bolt made by the 3D scanner and the result of the wrong match (red dot)

Fig. 37  Template from database

Fig. 36 shows the wrong position of the dot.
Fig. 38  Plot of the Matlab normalized cross-correlation function

The highest peak for this template was ~62%. As expected the value is much lower as the best result.

The idea of the correlation is to use the algorithm with every image from the database and find the really best correlation. Obviously the highest percent will be expected in the image which has the same pose as the image from the scanner.

Fig. 39  Example of a small database

On Fig. 39 a small database is shown which is compared with the image in Fig. 33. On Fig. 40 the plot of the best matches for every image from the database is shown.
Fig. 40  Result for all matches in the database

The highest value was for bolt number 25 in the database which was the same as in Fig. 34. This kind of sorting allows to find the perfect match.

A clearly disadvantage of the cross – correlation appears while distortions are introduced to the image. The next example shows what happened with the efficiency of the correlation with some distortions (Fig. 41).

Fig. 41  Image from scanner, manual introduced distortions

The white dot was manually added to the image as a simulation of distortions just to show how the accuracy of the matching decreases. The next figure Fig. 42 shows the matches for the comparison. The best match is ~77% which is 7% lower than for the image without distortions.
Fig. 42 Result for all matches with introduced distortions

Another example of wrong matching could be shown on an image with a few bolts which are very tight. In Fig. 43 are 5 bolts randomly organized. The result shows a complete mismatch. The comparison of the background made the most of this correlation. The result is ~59% and this is the best result for this image.

![Image from scanner, example of wrong matching](image)

Fig. 43 Image from the scanner, example of wrong matching

This is an important task which has an highly influence of the later discussed pose estimation of one bolt from a whole group. The templates which are a fix part of the correlation always has the small background which in the database is just black (zero values). In the real case, where many bolts are positioned very tight to each other a most probably a mismatch will appear. In the next chapter a method will be explained which overcomes this problem.
3.4 Pose estimation

The object recognition and pose estimation is the most significant part of the vision part in a bin – picking approach. That is why these part is also the most complex and often the most difficult fragment. In these paper the solution is in some sense a compromise between the model and the feature approach. In this chapter a solution is explained which allows for an object position and pose detection.

In the previous chapter the functionality of the cross – correlation has been shown. If the input argument of the algorithm is a simply extracted bolt, the correlation is straight forward. The problems appeared when the bolts were situated very tight or one on the other. Obviously this is a common case for a group of bolts in a bin or a heap. So the main questions was, how to made use of the correlation for the pose estimation of one bolt in a group of randomly organized objects?

The answer of that question was to extract just one bolt from the heap, the top most object and correlate that one with the database to determine the pose of it. As long as all data were matrices the extracting or deleting of objects were quite simple. This paper presents a method of grouping the objects into different labels and extracting these, which a most probably a bolt. Later a sorting algorithm defines if it is really a bolt or not. Finally the best match is shown and the object position and pose is determined.

First a small part of the image has to be selected with the possible highest candidate to pick. The 3D scanner after taking an image calculates the height of the highest bolt and delivered this information to the Matlab – program (chapter 3). According to this information all distortions can be eliminated.
Fig. 44 Real range image with distortions  Fig. 45 Real range image without distortions

Fig. 44 and Fig. 45 show the effect of eliminating distortions (blue circle). The important part of this was to eliminate distortions which had a higher value than the top most bolt in the scene. Every distortion which remains and had the same or lower value as the highest bolt would be ignored in the later grouping process.

The next step was to eliminate all these bolts, which are under a defined threshold relative to the highest point.

Fig. 46 Cut the background bolts
This process eliminates distortions in terms of extracting the highest bolt from the image. All the pins of the lower situated bolts disappears which made the extraction much easier. The information about the highest bolt also gave the opportunity to cut the size of the real range image into a smaller one which contains the highest bolt. This is another advantage to eliminate distortions from the image. By searching the highest point the coordinates of part of the top bolt could be determined. At this stage the position of the bolt is unknown so to determine the part of the image which will be cut from the original, a safe distance to every side had to be set. This process had to be done to avoid to cut off a fragment of the topmost bolt. The maximum length of the bolt is 24 mm, the resolution of the scanner and the images in the database are 2px per 1mm. So a safe distance were 50px to every side. On Fig. 47 an illustration of that process is shown.

![Cut out the topmost bolt – illustrating of the cutting window](image)

This method eliminates all bolts or fragments of bolts which are outside the cutting window. On Fig. 48 the result of this cut is shown.
This was the first step in the object localization algorithm for this approach. The goal of this operations were to cut out as most as possible distortions to made the last extraction part as simple as possible.

The next part is separated into two cases. The first one is the case of a simple detected bolt. If the bolt had a good position and the scanner delivered a well image a simple algorithm could be used to extract the topmost bolt. In the second one an sorting algorithm was created which got some best candidates and in the end the best were chosen as the positive match.

The first case will be explained on another image from the scanner. The first processes are the same as described above.

The new image (Fig. 48) is labeled into groups. To do that an already existing Matlab function is used called “bwlabel”. This function returns a matrix containing labels for the connected objects in the image in which this functions was used. Next the size of each label is checked, if the size were smaller than 300px the label was cut off. Normally a bolt has a size between 400-800px depending on the position. To illustrate the labeling another example of a real range image was introduced (Fig. 49).
Fig. 49 Real range image

Fig. 50 Cut highest bolt

On Fig. 51 the result of the labeling is shown. Every color has its own number. For example if the red object is label number one, than every pixel in this label has the value one. This allows later for easier access to specific labels. An important part of this labeling was that if there were at least one pixel which connects two labels, this labels were represented as one. An example of that can be seen on the green label.

Fig. 51 Illustration of grouping the image into labels

Fig. 52 Labels left after eliminating small ones

Fig. 52 shows the result of eliminated small labels from the image.

Next the rest of the image had to be grouped again, but this time the green label has to be split into separate ones. This can be done automated by again labeling the image but first the edges had to be calculated to separate the bolts. Fig. 53 and 54 show how to separate the bolts from each other.
In the left image (Fig. 53) an easy edge detection algorithm was used to determine the edges of the bolts. That was not enough since there were still pixels between the bolts. As mentioned above to separate two labels there could not be any free pixel between the labels. So on the right image (Fig. 54) a dilation filter and a bridge function were applied. This expands the edges and bridges unconnected pixels (“that is, sets 0-valued pixels to 1 if they have two nonzero neighbors that are not connected” [Mathworks]).

Again the labeling function was applied. The program searched for a label which size was in range of 400-800px. If it founds something all indices of that label were saved in a separate image. The values of the same indices but in the original image were saved to the new image which now contained the original depth values of that specific label.
This solution works for simple bolts which are not too distorted. If the bolt lies in the moving direction of the scanner, the head of the bolt could also throw a shadow on the pin. In that case an edge will appear in the middle of the bolt which separates it. That means that non label will be in the defined threshold of 400-800px. If the bolt lies more abruptly the edge threshold of the edge-detection could be too high and the algorithm also separates the head from the pin.

To handle this problem a sorting algorithm has been implemented.

In Fig. 59 a typical example of an image in which the algorithm above wouldn’t work.

A black, thin line is visible between the head and the pin. That will result with an edge, in the labeling process which separates the bolt into two labels. In the next step a sorting algorithm will be introduced which fits that kind of labeling into one bolt.

The image was processed in the same way it was done in the previous steps. The sorting algorithm also operates just on the highest bolt surrounded by distortions (which are fragments of other bolts). First the cut image was labeled, same as previous. All labels which
are smaller than 100px were eliminated. In the first algorithm all labels which were smaller than 300px were deleted but for this approach it's assumed that a bolt is separated so it could happened that an important fragment of the bolt disappears. For the rest of the image an edge detection algorithm was applied but this time the edge threshold was set down twice lower than the former one because now the goal was to split the head from the pin. Because of the fact that there is always a depth variance between this features (head and pin) a low threshold forces an edge between them. Of course also an dilation filter and bridge operation has to be done. The result of that can be seen on Fig. 60.

For the human eye it is obviously which two labels fit one bolt. To let a program decide of that was a little bit more tricky. To follow the idea of finding two labels which fit one object, first a master label has to be defined. This master label must be a part of the highest bolt, it could be the head or the pin, depending on which part is located higher. To set the master label the mean of the depth values of every label were calculated. This allows to determine the highest one which has to be a part of the topmost bolt. On Fig. 62 the grayscale simulates the depth of the label – the darker the scale the higher the label.
For labels which were situated really low, a condition was added that every label which mean depth was lower than 20% of the master label was eliminated. The next step of the sorting algorithm was to calculate the coordinates of the middle of every label. Fig. 63 illustrates that process.

The indices of the pixels in every label are known. To get the coordinates of the middle of the label the minimum and maximum values in the x- and y- axes had to calculate. By calculating the arithmetic mean the coordinate \((x_m, y_m)\) determines the middle point. That process had to be done for every remaining label in the scene. After that the distance between the middle
points of every label could been calculated. If the distance is longer than a defined threshold the label got deleted (Fig. 64).

Fig. 64  Calculating the middle of a label

The threshold was set to 30px because the distance between middle of the pin and the head couldn’t be longer than 15 mm (30px). That limited the image to just a few labels, in this case to the label x6 and x5. In the last step of this algorithm both labels were extracted to different images and correlated with a small database like in Fig. 39. The expectation were that the best correlation would be for the two labels which fits one bolt.
4. Results

Results are very important since they serve the role of validation for the whole project. In this part of the thesis the results of interaction of all algorithms along with its description are presented. The most important factors to measure in this project were: computation times, percentage of matches and recognition success rates. Hence the emphasis is placed to these factors. Two tests have been made to investigate the correctness of the implemented solution:

1) First tests have been made on a stack of 15 randomly positioned screws. The highest bolt is always considered as a best pick candidate. Best match in this test has been removed from the stack after successful matching. The test lasted as long as all screws were “picked up”. This trial has been made by using two different databases. That allowed to explore the influence of the number of images in the database on computation time and percentage of match.

2) In the second approach the randomly positioned stack has been changed after the correct pick-candidate has been designated. This enabled to test the performance and computation times of the algorithm.

4.1 Test 1:

The results of this test are presented on the Fig. 65 - 79. Computation times and percentages of matches can be found in Table 1 and Table 2. For each image from the scanner, corresponding best match from database is also shown.
Fig. 65  Randomly organised bolts 1
Fig. 66  Randomly organised bolts 2
Fig. 67  Randomly organised bolts 3
Fig. 68  Randomly organised bolts 4
Fig. 69  Randomly organised bolts 5
Fig. 70  Randomly organised bolts 6
Fig. 71  Randomly organised bolts 7
Fig. 72  Randomly organised bolts 8
Fig. 73  Randomly organised bolts 9
Fig. 74  Randomly organised bolts 10
Fig. 75  Randomly organised bolts 11
Fig. 76  Randomly organised bolts 12
Fig. 77  Randomly organised bolts 13

Fig. 78  Randomly organised bolts 14

Fig. 79  Last bolt

Table 4. Results of Results of test1 matching for database with 690 images

<table>
<thead>
<tr>
<th>Number of the image</th>
<th>Which algorithm found the object</th>
<th>Best match [%]</th>
<th>Computation time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>79,22</td>
<td>8,76</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>78,91</td>
<td>7,33</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>84,66</td>
<td>4,65</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>87,01</td>
<td>8,13</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>83,63</td>
<td>4,09</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>78,18</td>
<td>6,58</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>80,05</td>
<td>5,95</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>82,95</td>
<td>0,86</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>76,69</td>
<td>9,99</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>81,05</td>
<td>2,74</td>
</tr>
</tbody>
</table>
The orientation of each top-most object has been found with 100% success rate. The results regarding computation times clearly shows how huge impact on this factor has the number of images in the database. The difference in average match between the database with 690 and 2600 images is only 0,79% while computation time drastically increases. This shows how important is selection of appropriate number of images in the database. The average

### Table 5. Results of Results of test1 matching for database with 2600 images

<table>
<thead>
<tr>
<th>Number of the image</th>
<th>Which algorithm found the object</th>
<th>Best match [%]</th>
<th>Computation time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>79,22</td>
<td>25,58</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>79,17</td>
<td>24,87</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>86,03</td>
<td>23,67</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>87,01</td>
<td>22,24</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>88,50</td>
<td>40,25</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>78,18</td>
<td>24,04</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>80,50</td>
<td>20,59</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>83,33</td>
<td>0,733</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>77,23</td>
<td>24,08</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>81,47</td>
<td>6,88</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>78,90</td>
<td>23,58</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>81,69</td>
<td>23,79</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>82,96</td>
<td>4,36</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>82,91</td>
<td>23,77</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>84,85</td>
<td>18,42</td>
</tr>
</tbody>
</table>

**Average match [%] – 82,13%**

**Average computation time [s] – 20,45 s**

**Difference between average matches - 82,13 - 81,34 = 0,79 %**

**Difference between average computation times - 20,45 - 5,523 = 14,927 s**
match for both cases is under 80% what can be considered as a big success since the virtual and real object always differ slightly. Computation times can be shortened by reducing the threshold of searching, for example: stop the algorithm when the match will be greater than 75%. The presented results have confirmed the functionality of the algorithm, its robustness and invariance from illuminations.

4.2 Test 2:

The results of this test are presented on the Fig. 80 - 89. Computation times and percentages of matches can be found in Table 3. For each image from the scanner, corresponding best match from database is also shown.
Fig. 82  Randomly organised bolts 3

Fig. 83  Randomly organised bolts 4

Fig. 84  Randomly organised bolts 5

Fig. 85  Randomly organised bolts 6

Fig. 86  Randomly organised bolts 7

Fig. 87  Randomly organised bolts 8
Table 6. Results of test2 matching for database with 690 images

<table>
<thead>
<tr>
<th>Number of the image</th>
<th>Which algorithm found the object</th>
<th>Best match [%]</th>
<th>Computation time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>81,02</td>
<td>12,80</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>84,17</td>
<td>5,23</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>80,70</td>
<td>3,06</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>82,41</td>
<td>7,96</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>83,98</td>
<td>13,82</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>83,92</td>
<td>8,04</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>79,06</td>
<td>9,51</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>80,26</td>
<td>5,09</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>77,66</td>
<td>0,62</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>75,83</td>
<td>6,58</td>
</tr>
</tbody>
</table>

**Average match [%] – 80,90%**

**Average computation time [s] – 7,27 s**

As in test one, the success rate for this test was also at the highest possible - 100%. This test has confirmed that algorithm works for any randomly organized stack of bolts with high accuracy of matching and in acceptable time.
Chart presented on Image 700 illustrates the percentage of similarity between extracted screw and each image from the database. The biggest pick (marked with red circle) is the best match. X axis represents each object from the database while y matches.

4.3 Summary of the results

In tests presented in this part, emphasis has been put on a three factors: computation times, percentage of best matches and success rate of the algorithm. The results of tests clearly shows that created solution is working fine for the chosen object. Average computation times for the algorithm using database with 690 images were between 5.6 – 7.3 s. That can be considered as a good result since the Matlab software is not optimized for such tasks. Presented solution worked for every possible case with average recognition rate not less than 80%. However changing the number of images in the database to 2600 resulted in increasing the average computation time to 20.45 s. That definitely could be reduced by setting the threshold of matching at the appropriate value. From these tests it can be said that presented solution is invariant from distortions, illuminations and work for occluded scenes. Since that was a goal of this work it can be said that all thesis objectives has been fulfilled.
5. Conclusion

In this thesis vision system application for a bin-picking system has been created. Solving of such complex task was really hard since many interacting algorithms had to be created. Two most important of them were: database algorithm and object localization algorithm. Database created in this thesis enabled to store any object for which CAD model was available. The object can have any shape, size and any properties. The only constrain in creating the database is hardware capability. This database has been successfully used in the whole project for object recognition purposes. The second, essential algorithm concerned the object localization algorithm. Many tests showed that presented algorithm works fast, is invariant to illuminations and works for occluded scenes. Since that was major goals of this thesis it is fair to say that all goals were fulfilled. As a final result functional solution has been created which could be basis for larger project. Bearing in mind that the whole thesis has been done within only five months that shows how big effort was put into its realization.

5.1 Future work

Since the all programs have been implemented in Matlab software first possible action in future is converting programs to other, faster programming language for example C++, C#. Another task which can be done is to make the program more universal since for the purpose of this thesis screw object was used. Very beneficial would be also implementing algorithm on GPU (CUDA) which would enable parallel computation and hence result in reducing the computation time to less than one second.
References


8. ThyssenKrupp Krause Corporation: Information according the company. www.krause.co.uk


## Content of figures

<table>
<thead>
<tr>
<th>Fig.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Example of simple bin-picking system</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>Overall bin-picking application algorithm</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Top view of the bolt</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>Side view of the bolt</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>Basic chart describing the idea of work</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Bak Harpen</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>Discs</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>Stator Housings</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>SICK IVP-3D scanner used to capture real images. [IVP webpage]</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>Example of scanning an object [IVP webpage].</td>
<td>16</td>
</tr>
<tr>
<td>11</td>
<td>Side view of the workplace</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>View of the workplace from the top</td>
<td>18</td>
</tr>
<tr>
<td>13</td>
<td>Diagram presenting the connections between devices</td>
<td>18</td>
</tr>
<tr>
<td>14</td>
<td>Encoder used in a workplace</td>
<td>19</td>
</tr>
<tr>
<td>15</td>
<td>Real image of the workplace</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>Real image of the workplace from different view</td>
<td>20</td>
</tr>
<tr>
<td>17</td>
<td>Visualization of the most important variables</td>
<td>22</td>
</tr>
<tr>
<td>18</td>
<td>Example of the image from the scanner with smoothing filter applied</td>
<td>23</td>
</tr>
<tr>
<td>19</td>
<td>Small fragment of the raw data image</td>
<td>23</td>
</tr>
<tr>
<td>20</td>
<td>3D visualization from the scanner</td>
<td>24</td>
</tr>
<tr>
<td>21</td>
<td>Result of implemented algorithm</td>
<td>25</td>
</tr>
<tr>
<td>22</td>
<td>Structure of STL model [Wikipedia]</td>
<td>27</td>
</tr>
<tr>
<td>23</td>
<td>Model of bolt imported into Matlab software and example of vertices matrix</td>
<td>28</td>
</tr>
<tr>
<td>24</td>
<td>Bolt with corresponding vertices description - frontview</td>
<td>28</td>
</tr>
<tr>
<td>25</td>
<td>Bolt with corresponding vertices description</td>
<td>29</td>
</tr>
<tr>
<td>26</td>
<td>Triangle with new points.</td>
<td>30</td>
</tr>
<tr>
<td>27</td>
<td>General scheme of the algorithm used to create the database</td>
<td>33</td>
</tr>
<tr>
<td>28</td>
<td>Small part of pose reference database</td>
<td>33</td>
</tr>
<tr>
<td>29</td>
<td>Small part of the database with random rotated objects</td>
<td>34</td>
</tr>
<tr>
<td>30</td>
<td>Database created from points described by pure STL model</td>
<td>34</td>
</tr>
<tr>
<td>31</td>
<td>Part of the database cratered for other object</td>
<td>34</td>
</tr>
<tr>
<td>32</td>
<td>Principle of the sliding window using cross – correlation for images</td>
<td>36</td>
</tr>
<tr>
<td>33</td>
<td>Example of a bolt made by the 3D scanner and the result of the best match (red dot)</td>
<td>37</td>
</tr>
<tr>
<td>34</td>
<td>Template from database</td>
<td>37</td>
</tr>
<tr>
<td>35</td>
<td>Plot of the Matlab normalized cross-correlation function</td>
<td>38</td>
</tr>
<tr>
<td>36</td>
<td>Example of a bolt made by the 3D scanner and the result of the wrong match (red dot)</td>
<td>38</td>
</tr>
<tr>
<td>37</td>
<td>Template from database</td>
<td>38</td>
</tr>
<tr>
<td>38</td>
<td>Plot of the Matlab normalized cross-correlation function</td>
<td>39</td>
</tr>
</tbody>
</table>
Fig. 83  Randomly organised bolts 4 ................................................................. 58
Fig. 84  Randomly organised bolts 5 ................................................................. 58
Fig. 85  Randomly organised bolts 6 ................................................................. 58
Fig. 86  Randomly organised bolts 7 ................................................................. 58
Fig. 87  Randomly organised bolts 8 ................................................................. 58
Fig. 88  Randomly organised bolts 9 ................................................................. 59
Fig. 89  Randomly organised bolts 10 ............................................................... 59
Fig. 90  Results of matching for each image ...................................................... 60