BELARUSIAN STATE UNIVERSITY

Automation of CMS Phase II Tracker Module assembly

by

Artsiom Bryksa

A thesis submitted in fulfillment for the
degree of Specialist

in the

Faculty of Radiophysics and Computer Technologies
Department of Telecommunication and Information Technologies

June 2017

BELARUSIAN STATE UNIVERSITY

Abstract

Faculty of Radiophysics and Computer Technologies
Department of Telecommunication and Information Technologies

by Artsiom Bryksa

CMS (PS 2S), , . PS , 40 . , . , ,
.

Acknowledgements

, , , , CMS DESY , .

Contents

List of Figures

Abbreviations

AL-CF	**C**arbon-**F**ibre reinforced **AL**uminium
CF	**C**arbon-**F**fibre
CMS	**C**ompact **M**uon **S**olenoid
HL-LHC	**H**igh **L**uminosity **LHC**
LHC	**L**arge **H**adron **C**ollider
PS	**P**ixel **S**trip (module)
SSBA	**S**ensor-**S**pacer-**B**aseplate-**A**ssembly
2S	**2 S**trip (module)

Chapter 1

() – . 2010 2013 . – , (Franois Englert) and (Peter Higgs) 2013 . , – ATLAS, CMS, ALICE LHCb [**?**].

1.1

(Compact Muon Solenoid CMS) – , , () / . 28 15 . 14000 . CMS 3- : , , , – , 3.8 .

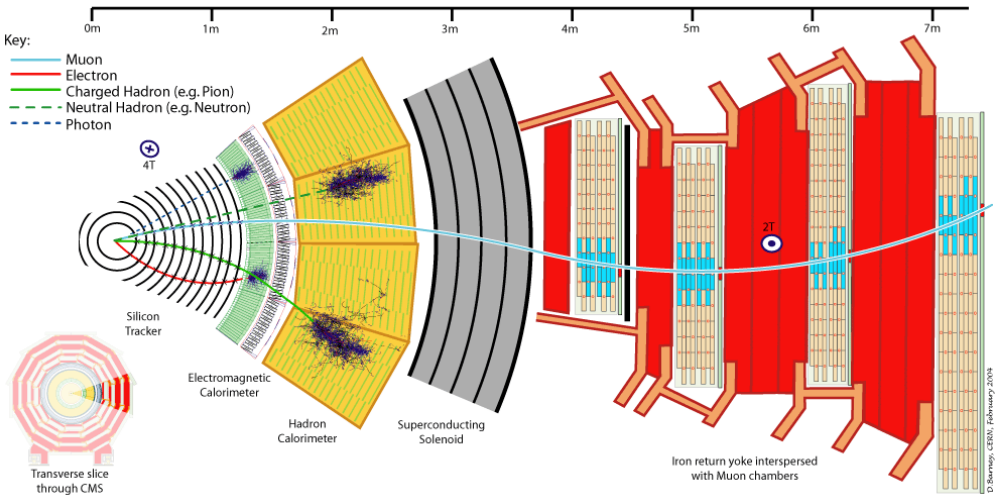

FIGURE 1.1: CMS .

CMS , . **??**.

1.2 CMS

After Phase II Upgrade, the LHC will provide a much higher luminosity. This regime is known as the High Luminosity LHC (HL-LHC). A serious problem presented by these conditions is the enormous data readout rates that exceed far beyond the bandwidth foreseen for the readout electronics. However, the vast majority particles produced in the HL-LHC conditions are not of direct interest for new physics searches

and are characterized by low transverse momentum. Thus rejecting tracker hits related to low transverse momentum particles can significantly reduce the amount of data to be readout. In order to provide momentum discrimination at the hardware level, a 2-layer module design was created. The central idea of the new modules is to provide fast discrimination between low and high transverse momentum particles by estimating the track curvature caused by the magnetic field within the volume of the module itself. For example, particle with high transverse momentum after hitting some pixel/strip at the first sensor layer would hit on of neighboring pixels/strips of the respective pixel/strip on the second layer. While a particle with low transverse momentum would have a more curved trajectory an hit pixel/strips at a displaced position from the first hit. By varying the distance between sensors and number of neighboring pixel/strips required to match hits in adjacent sensors, (2 neighboring strips in the Figure **??**) it is possible to set the transverse momentum threshold for a hit [**?**].

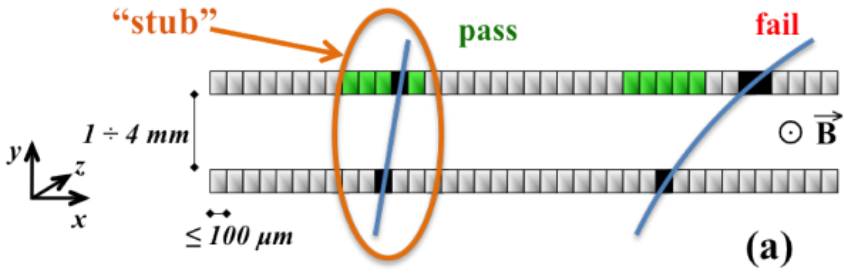

FIGURE 1.2: An example of distinguishing high and low transverse momentum. Particles, which hit any of two neighboring strips or the respective strip itself, would be record as high transverse momentum particles.

Two layer Modules

The CMS Phase-II Tracker will utilize two types of modules, 2S modules and PS modules. To achieve efficient rejection of low-pT (low transverse momentum) particles throughout the Tracker volume, modules in different regions will make use of a few different sensor spacings. For 2S (PS) modules, spacings of 1.8 and 4 mm (1.6, 2.6 and 4 mm) are foreseen. These modules will be used in the end-cap disks as well as the central barrel region of the Tracker. An exploded view of a PS module is shown in Figure **??**.

In the PS module, the sensors are glued to a carbon-fibre reinforced Aluminium (AL-CF) spacers which act as spacers and provide the thermal conductance crucial for the cooling of the module. The two sensors and spacers are in turn glued the carbon-fibre (CF) baseplate. This structure is henceforth referred to as the sensor-spacer-baseplate-assembly (SSBA). This project will focus on the assembly of the SSBA only. The precision requirements of the SSBA are shown in Figure **??**. For the PS module, the sensors must align to within 40 mm measured at the sensors short edge. This corresponds to a rotational alignment tolerance of 0.8 mrad [**?**].

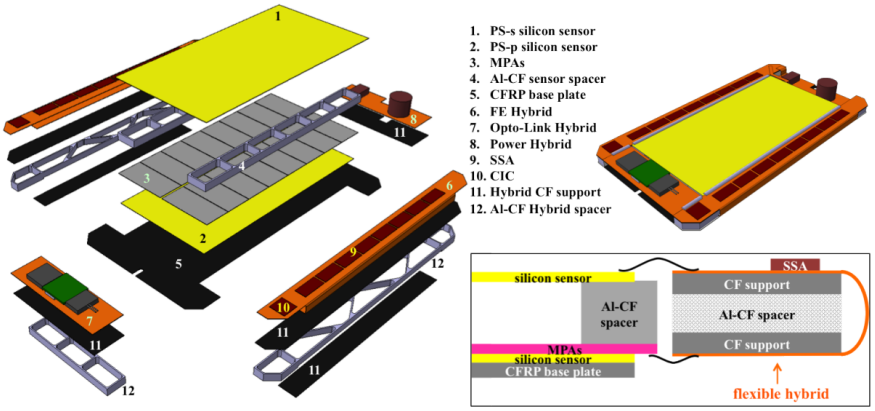

FIGURE 1.3: Exploaded view of Pixesl Sensor Module.

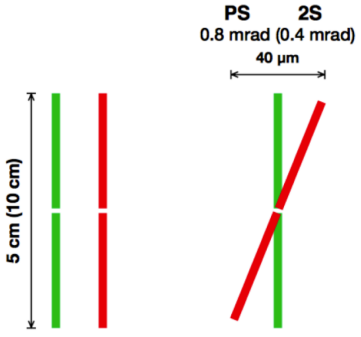

FIGURE 1.4: The precision requirements of the assembly of PS and 2S modules.

Chapter 2

Module assembly

Usual way to assemble such kind of high-precision sensors is a manually assembly with a help of mechanical jig. Alternative way is the proposed automated assembly system.

2.1 Manual assembly with mechanical jig

One option for module assembly is manual assembly with a custom-built mechanical jig (prototype is shown on Figure **??**) [**?**].

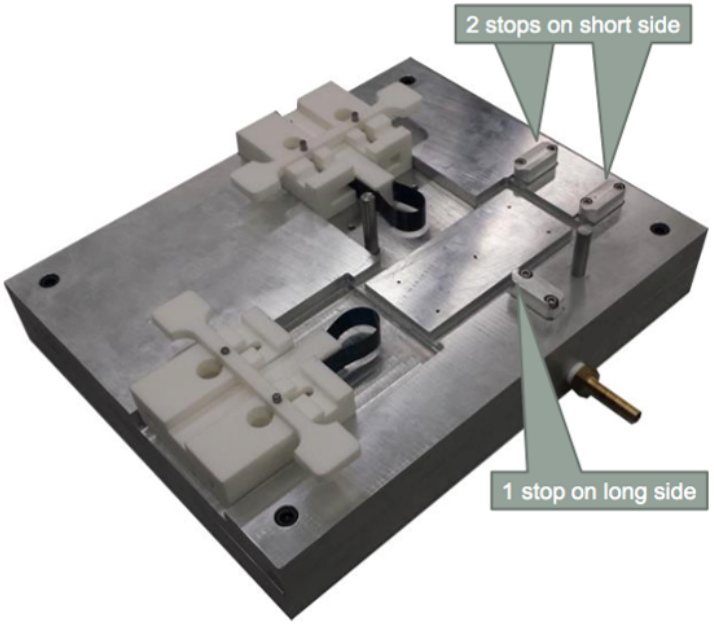

FIGURE 2.1: Prototype of the mechanical jig for module assembly.

With such method of assembly all parts are placed in the jig and glued together manually. To provide required accuracy mechanical jig has 3 reference stops: one along the longer side of the module and two along the shorter side. From the opposite to reference stops sides the module is gently pushed by springs towards reference stops.

Together they provide enough precise positioning of the module's components before and during gluing.

However, such method of module assembly has a number of disadvantages. It is relatively slow and does not scale well for high number of modules to assemble. In addition, such mechanical system has poor repeatability and has no options to control the process. Moreover, it needs very precise machining technologies (several microns precision) to manufacture this mechanical jig, as well as regular calibration of reference stops positions. Finally, this mechanical system need maximum manual handling and highly depends on human operating it. This fact means that even though in theory system can provide required quality of the assembled modules, there will be always more or less several percentage of modules assembled out of required quality only because of a human mistake.

2.2 Automated assembly system

The proposed automated assembly system consists of three subsystems: the motion subsystem, the vision subsystem and the vacuum subsystem (Figure **??**)[**?**].

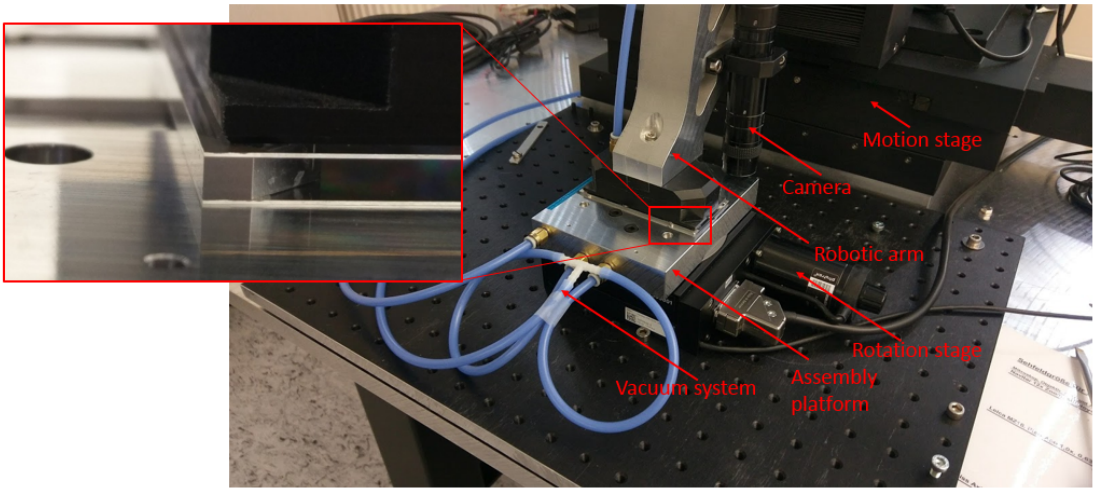

FIGURE 2.2: Proposed automated assembly system.

The *motion subsystems* provides the precise mechanical movements needed to arrange the components comprising the SSBA. Movement of the components is achieved via mounting on the two moveable parts of the motion system: the x-y-z and rotation stages. An custom-built AL tool known as the robotic arm is mounted on the x-y-z stage allowing mounting of components and thus movement of components in Cartesian coordinates. Components placed on the rotation stage may rotate in the horizontal plane with an angle θ. The motion stages are controlled by a motion controller unit. All motion hardware is manufactured by Lang 1 with motion precisions of 4 um and 2 mrad respectively. The *vacuum subsystem* enables the mounting of components to the

arm and rotation stage. It consists of a single pump providing vacuum to four switchable valves which in turn distribute vacuum to independent vacuum lines. The valves are switched to on(off) states by applying a control signal of 12 (0)V. The 12V signals are provided by a relay card. One vacuum line connects to the a pickup tool which is mounted on the arm, others – to the assembly platform. The pickup tool consists of an ESD plastic block housing an inner vacuum chamber which distributed the vacuum to an array of downward facing suction cups which slightly protrude below the bottom side of the tool. A diagram of the pickup tool is shown in Figure **??**.

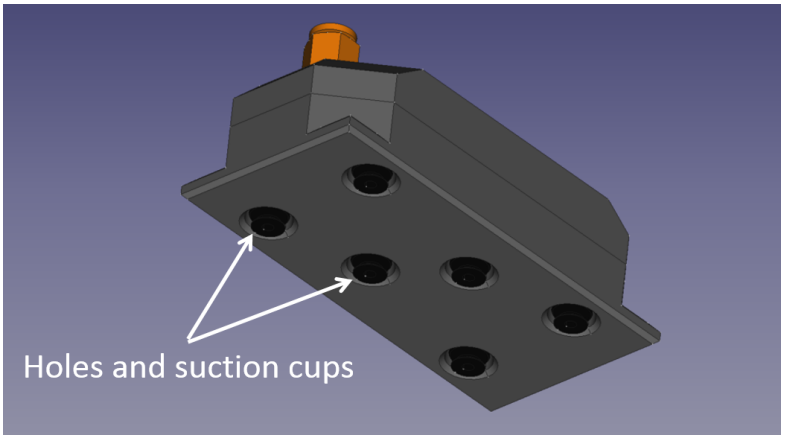

FIGURE 2.3: Pick-up tool.

The pickup procedure is performed by contacting the suction cups with the sensor, switching on the vacuum within the pickup tool and moving the arm directly upwards with the sensor attached. The setdown procedure is performed by contacting the mounted sensor (or baseplate) with the lower surface on which the component will be placed, switching off the vacuum in the pickup tool and moving the arm away. In order to avoid any movement of the component as the arm moves away, the component will be fixed in its setdown position with a separate array of upward-facing suction cups and independent vacuum line. The vision system acquires images of components allow determination of their positions and orientations which is crucial for precise assembly.

The *vision subsystem* consists is represented by high-resolution camera by IDS. The camera is mounted on the arm and is referred to as the mobile camera. It is fixed in a downward-facing orientation. The camera acquires images immediately before pick-ups and immediately after setdowns in order to determine the positions of unmounted components.

2.3 Assembly platform

One of the very important part of the automated assembly system is an assembly platform. the main purpose of it is to fix module components underneath with a vacuum.

The assembly platform should fulfil the following requirements:

1. Fix all necessary module components with vacuum on top of the platform.

2. Provide the possibility of precise placement and orientation for two spacers.
3. To be reasonably light and have center of mass close to the rotation axis of a rotation stage it will be attached to.

In order to match above mentioned requirements, the following design was proposed (Figure **??**):

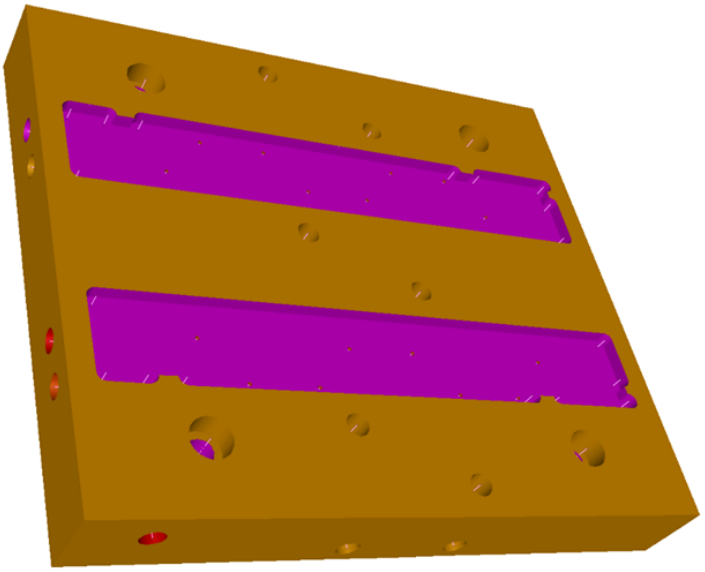

FIGURE 2.4: Design of the assembly platform.

Assembly platform has two inserts with three reference stops each to provide precise placement of spacers. Comparing to mechanical jig, assembly platform's reference stops are part of the platform itself thus do not need a calibration. However, having such inserts makes direct placement of the other flat components (sensors, baseplate) on top of the inserts impossible, because major area of them would have lack of underneath support under the pressure of pickup tool on top while glue curing. To solve this problem it was decided to place these components on the assembly platform perpendicular to spacers. In this case only a small area of components would have no support underneath, which is fine for assembly tasks. Perpendicular components placement on the assembly platform can be easily provided by rotation stage the platform mounted on. The platform center of mass is very close to the rotation axis (less than 1 mm). Hence its weight is around 1 kg thus there will no negative effects on the precision of the rotation stage operations.

The assembly platform houses two independent inner vacuum chambers: the first for spacers holding, the second for holding other flat module components. The chamber for spacers' vacuum system distributes vacuum into the array of tiny holes (0.7 mm in diameter) on the bottom of the inserts. The size and placement of these vacuum holes is determined by the shape of spacers (Figure **??**). These holes do not equipped with a suction cups due to such tiny size. So small suction cups simply do not exist in the market. Additionally, the flatness of contiguous surfaces (inserts bottom and spacer)

cause relatively small vacuum leakage thus providing enough tight vacuum fix of the spacers.

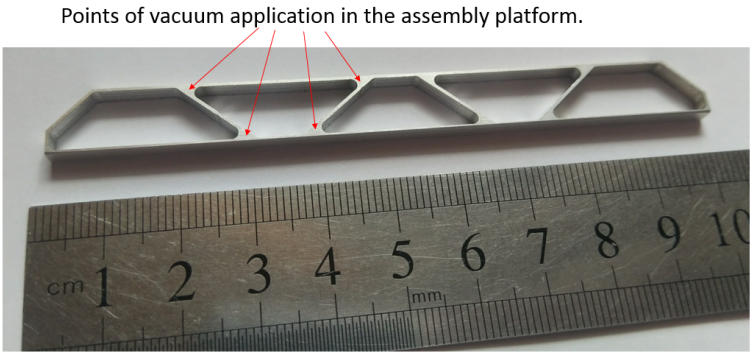

FIGURE 2.5: Al spacer for the PS module.

The second vacuum chamber distributes the vacuum in the array of suction cups to hold other flat module components (sensors and baseplate). It uses the same suction cups as pickup tool. Just as in the pick up tool, they stitch out a bit form the surface of the assembly platform. However, they should not prevent gluing components to spacers, stored in the inserts. That is why the depth of the interests is less than the thickness of spacers thus upper surface of the spacers stored in the inserts is a bit higher than suction cups. In other words, suction cups do not prevent operating with spacers as soon as they are lower than upper surface of the spacers.

2.4 Fast curing adhesive

One of the crucial part of the automated assembly system is an adhesive using in the assembly, to be more – its curing time. Otherwise, it makes a few sense to leave one module for a long time in the whole setup simply waiting the adhesive curing time. For instance, current guideline glue takes around 24 hours to cure. That is why automated assembly needs a technique to avoid such long waiting. One of the proposed decision is using a small amount of fast curing adhesive in addition to main guideline glue. This adhesive should fulfil the following requirements:

1. Provide reasonable bond after about 15 minutes.
2. Have no interference with main adhesive.
3. Have a thin layer (approximately less than 30 um).

Moreover, mentioned above requirements should be completed with as small amount of fast curing adhesive as possible. For gluing tests we used simple rectangular shape Al samples (representing Al-CF spacers) and glass samples (representing Silicon sensors). Reaching listed requirements highly depends on two aspects: first – the way fast adhesive is applied, second – fast adhesive properties.

There are lots of possible way of applying fast adhesive: several fast adhesive drops inside main adhesive layer, several drops close to edges, bevel gluing (fill the bevel space

of the Al sample with fast adhesive), side gluing (put some fast adhesive on sides), etc. For our system we decided to test bevel gluing and fast adhesive drops close to the edge. Even though bevel gluing can provide reasonable bond after 15 minutes, it is way more handy to use just two fast adhesive drops close to the edge. Adhesive was applied as shown on the Figure **??**.

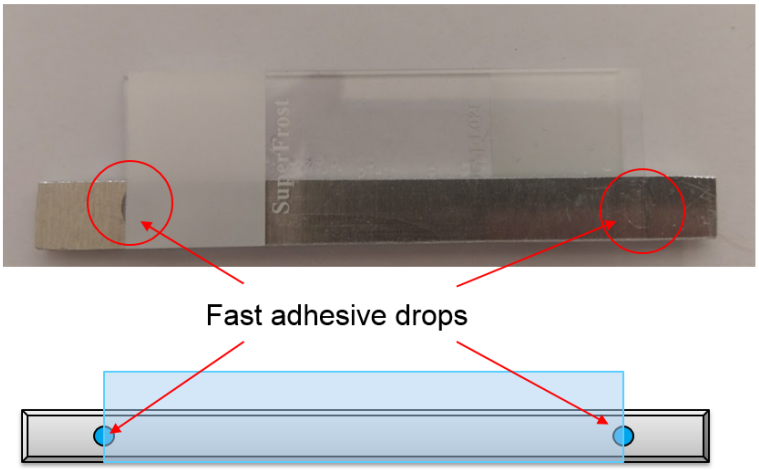

FIGURE 2.6: The way of fast adhesive application during glue tests. Much smaller drops can be used for obtaining reasonable bond.

The second part of the part of the question – finding proper fast adhesive in the market. The main issue of this question is combining two following properties: *fast curing* and *low viscosity*. Low viscosity is an integral property of the adhesive in case it should provide thin glue layer. However, fast curing means that adhesive must become hard quickly which is easier when it is initially has high viscosity. That is reason it is hard to find an adhesive combining both these properties. Nevertheless, there were several candidates for tests:

1. *Polytec EP 660*. According to the datasheet [**?**], its curing time is about 16 hours which is too far beyond the requirements. However, the distributor assumed that the so called *handling time* is much shorter. Unfortunately, it this adhesive did not provide any bond after 15 minutes, as expected. The only advantage of this adhesive is that it is the same vendor as main guideline adhesive what could probably results in some financial benefits.
2. *Loxeal 31-42*. According to the datasheet [**?**], its full curing time is $\approx 20 - 30$ minutes, while handling time is around 3-8 minutes. As a result it successfully provided reasonable bond after 15 minutes.
3. *Wekem WK5*. According to the datasheet [**?**], its curing time is around 5 minutes. It also provided reasonable bond after 15 minutes, but Loxeal adhesive has better quality and easier to operate with. Moreover, Loxeal glue provide thinner glue layer under the same pressure while curing: < 20 *um* while Wekem provides around 40 um (Figure **??**).

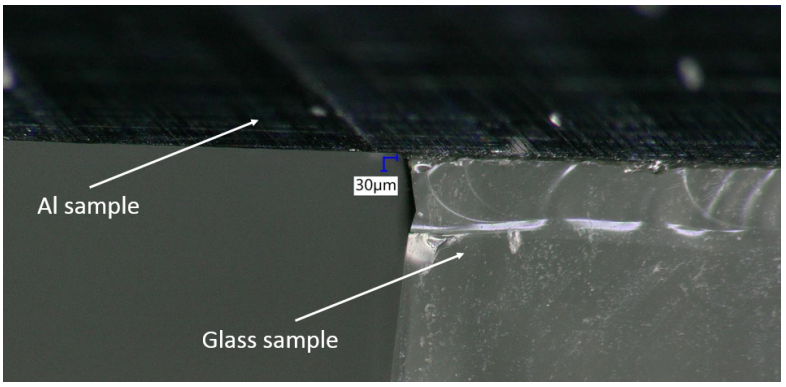

FIGURE 2.7: Glue layer thickness of Loxeal 31-42 under the pressure of around 20 g/cm^2

Talking about interaction with the main guideline adhesive, a series of test was done to check this property of the Loxeal 31-42. Among them were:

1. *Gluing test of two drops of fast and main adhesive in touch.* Tightness of glue bond remained approximately the same after 15 minutes curing comparing to separate fast gluing test.
2. *Gluing test of two mixed drops of fast and main adhesive.* Tightness of glue bond slightly decreased after 15 minutes curing comparing to separate fast gluing test.

To conclude, Loxeal fast adhesive clearly showed better qualities among other adhesives and fulfil all the requirements: it provides reasonable bond after 15 minutes and light pressure (around 20 g/cm^2), has thin glue layer – $<$ 20 um and show no interaction with the main guideline adhesive (touching the main adhesive).

2.5 Automated assembly process

Automated assembly process of a PS module consists of following steps:

1. *Prepare top sensor.* Firstly, put manually the top sensor to the platform and fix it with the vacuum. Next, detect its location and orientation with a help of camera and special software and save the data. Correct the orientation of the sensor with a help of rotation stage so thus it will be parallel to the X-axis of the motion stage. Finally, pick it up with the pickup tool and leave it attached to it. The vacuum on the pick up tool will remain applied during the whole assembly process beginning from this very first step.
2. *Prepare spacers.* Once the top sensor removed from the platform and attached to the pickup tool, the platform should rotate by 90 degrees so that spacers can take their spots with the correct orientation relatively to the top sensor. Next, gently pushed to the reference stops and got fixed with the vacuum from the platform. Then the system should find and locate the reference marker on the platform thus being able to deduce the location of the spacers.

3. *Glue top sensor to spacers.* Once both top sensor and spacers are ready and their locations are identified, the software can easily calculated the path for the pickup tool with attached top sensor on it. Next is to put the main adhesive and several drops of the fast adhesive on the spacers and move the pickup tool to the calculated gluing position. Special attention should be paid for Z coordinates of pickup tool moving as soon as it directly influence the thickness of the glue layer.
4. *Prepare bottom sensor.* After about 15 minutes the fast adhesive drops are cured and provide reasonable bond to remove just glued spacers with top sensor. To do this pickup tool should simply move upwards with vacuum remaining applied. After the platform is cleared it should rotate by 90 degrees so that the bottom sensor can be placed and fixed with the vacuum. Next, just as for top sensor, detect its location and orientation with a help of camera and special software and save the data. Correct the orientation of the sensor with a help of rotation stage so thus it will be parallel to the X-axis of the motion stage which means parallel to the top sensor.
5. *Glue bottom sensor to spacers + top sensor.* All the required components for this step are fixed and theirs locations are identified. Hence the software is able to calculate the path for the pickup tool. Finally, pickup tool can move to the gluing position after applying main and fast adhesives.
6. *Prepare baseplate.* After about 15-minutes glued structure can be lifted up with the pick up tool and its vacuum still remaining applied. Next is placing the baseplate paying attention for three reference pins and fix it with vacuum. These pins are not exist in the current version of the assembly platform as soon as gluing baseplate to the sensor-spacers-sensor (bare module) is not a high priority for the current state of the project. The reason for that – baseplate gluing do not required so high accuracy as bare module.
7. *Glue sensor-spacers-sensor structure (bare module) to the baseplate.* As soon as the software has already known the position of the reference marker on the assembly platform, hence it is able to calculate the location of the baseplate and the gluing position of the pickup tool with bare module attached to it. Finally, the pickup tool moves to the gluing position after applying main and fast adhesives.
8. *Automated assembly is done.* After about 15 minutes fast adhesive provides enough bond and the assembled module (a part of it to be precise) can be removed from assembly platform and left for 24 hours to let the main adhesive cure. However, baseplate's reference pins fits the baseplate very tight thus manual removing can likely cause break. That is why it is better to remove the module with a pick-up tool as soon as it can provide perpendicular lift which is safer.

Chapter 3

Control Software

In order to control the whole automated assembly system a PC-based Qt application was developed. All the necessary hardware is connected to the PC with USB interface.

3.1 General structure of the application

The control of the motion, vision and vacuum subsystems is integrated in a single software application henceforth referred to as PSAuto (from automated assembly of PS module). It is entirely written in C++ and utilises the Qt framework version 4.8.7. A schematic illustrating the integration of the subsystems with PSAuto is shown in Figure ??.

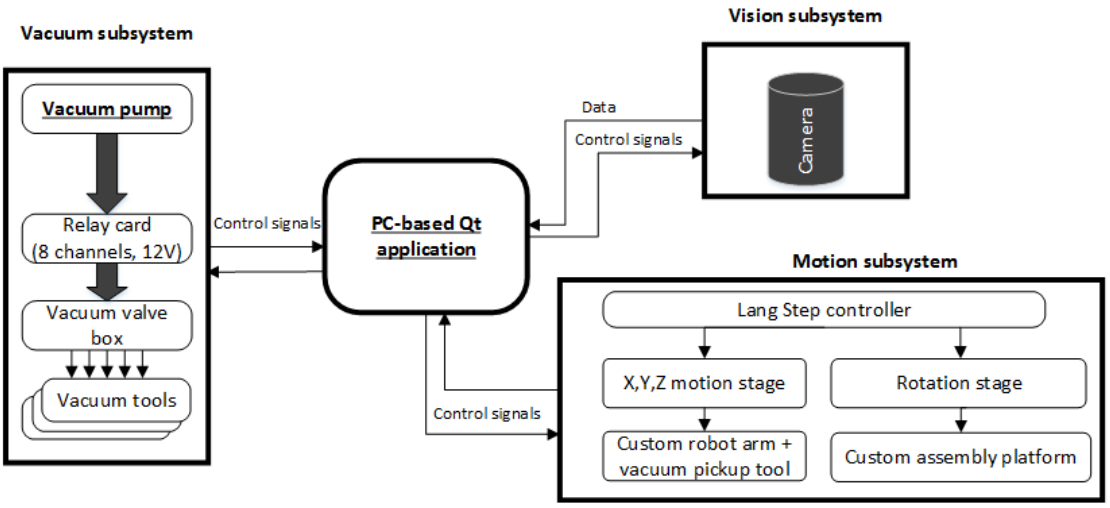

FIGURE 3.1: A schematic view of the integration of the motion, vision and vacuum subsytems via the Qt-application.

3.1.1 Model View Controller architectural pattern

The architecture of the application is based on Model-View-Controller (MVC) architectural pattern. It considers there to be three main types of objects: *model* objects,

view objects and *controller* objects. When designing an application, a major step is choosing or creating custom classes for objects that fall into one of these three groups. Each of the three types of objects is separated from the others by abstract boundaries and communicates with objects of the other types across those boundaries. The pattern defines not only the roles objects play in the application, it also defines the way objects communicate with each other [**?**]. The key point of MVC is that View and Controller depend on Model, but Model does not depend on them. The interaction of these three types is schematically shown in the Figure **??**.

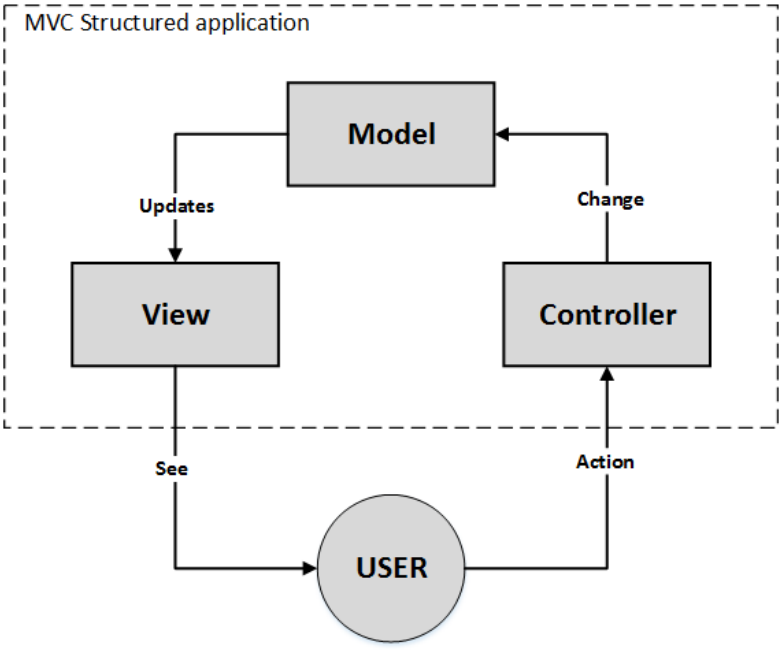

FIGURE 3.2: Model, View and Controller (MVC) relative to user.

Shown in the Figure **??** interaction is a classical MVC architecture. There are various realization of this diagram in terms of interaction between three main object types of MVC. The main reasons of it are various application types and their realizations. For instance, in the PSAuto application this diagram will look like in the Figure **??**.

Comparing to classical MVC structure from Figure **??**, the Model do not directly inform/update the View. This information passes through Controller. A controller object acts as the intermediary between the Model and the View. Controllers are often in charge of making sure the views have access to the model objects they need to display and act as the conduit through which views learn about changes to the model. Controller objects can also perform set-up and coordinating tasks for an application and manage the life cycles of other objects [**?**].

Depending on the application logic and demands, Model, View and Controller follow special properties and rules.

Model is central component of the system. It directly manages the data, defines the logic and rules of the application which manipulates the date. Moreover, like in our case, it regulates the interaction between the application and hardware. For instance, there is

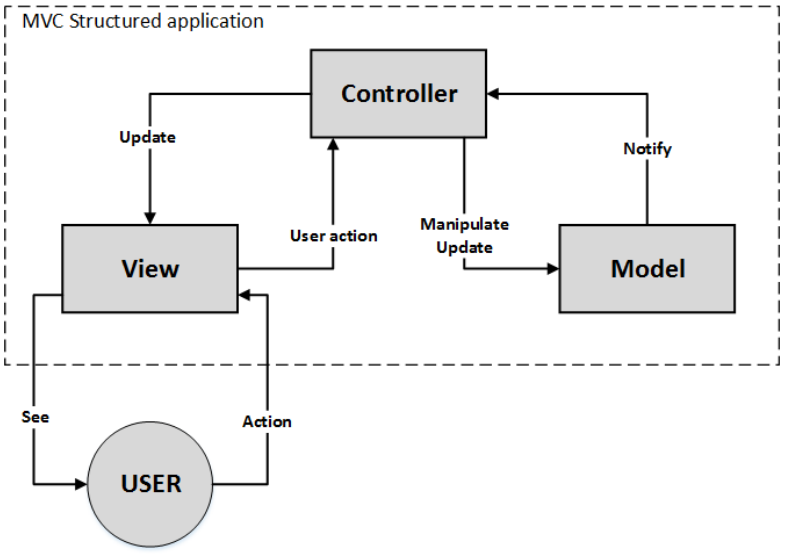

FIGURE 3.3: Model, View and Controller (MVC) relative to user.

a Model class *ConradModel* in PSAuto application. It is responsible for communication with a relay card for turning on and off vacuum lines. This class completely meet MVC rules as a Model object because it does not depend on any other class, while others depend on it using its possibilities for their functions.

View unite all objects responsible for visual part of the application: windows, tabs, labels, forms, buttons, etc. They know how to display the data from the application's model and also give a user the possibility to edit the data. For instance, one task of *AssemblyModuleAssembler* class is to display the information about Vacuum lines of the system, as well as control elements for it. However, it only display the control elements, but not process the user's actions.

Controller is mainly responsible for accepting input and processing it, generating commands for the Model or the View. For instance, *ConradManager* provides all necessary functions to control the vacuum lines and to get the current status of it. It is worth noting that very often there is no direct connection between the Model and the View, like it is shown in the classical MVC architecture in the Figure **??**. Instead of it, the Controller acts as the intermediary between the application's model objects and its view objects.

3.1.2 OpenCV library

A lot of algorithms needed for the application were found in the open-source library — OpenCV (*Open Source Computer Vision*). It is a cross-platform library of programming functions mainly oriented on real-time computer vision. It was originally developed by Intel's research center in Nizhny Novgorod (Russia). Later it was supported by Willow Garage and is now maintained by Itseez. The library is free for use under the open-source BSD license [**?**].

OpenCV was designed for computational efficiency and, as was already mentioned, with a strong focus on real-time applications. It is written in optimized C++ and can take advantage of multicore processors. There is also a possibility for further automatic optimization on Intel architecture with Intel's *Integrated Performance Primitives (IPP)* libraries, which consist of low-level optimized routines in many different algorithmic areas [**?**].

One of the main goals of OpenCV is to provide a simple-to-use computer vision infrastructure that helps people build fairly sophisticated vision applications quickly. In its libraries one can find over 500 functions that span many areas in vision. In the list below one can see main features of the OpenCV library [**?**]:

- Image data manipulation (allocation, release, copying, setting, conversion).
- Image and video I/O (file and camera based input, image/video file output).
- Matrix and vector manipulation and linear algebra routines (products, solvers, eigenvalues, SVD).
- Various dynamic data structures (lists, queues, sets, trees, graphs).
- Basic image processing (filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids).
- Structural analysis (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation).
- Camera calibration (finding and tracking calibration patterns, calibration, fundamental matrix estimation, homography estimation, stereo correspondence).
- Motion analysis (optical flow, motion segmentation, tracking).
- Object recognition (eigen-methods, HMM).
- Basic GUI (display image/video, keyboard and mouse handling, scroll-bars).
- Image labeling (line, conic, polygon, text drawing).

The application of the OpenCV libraries will be described more detailed in the next section.

3.2 Pattern recognition

Pattern recognition is a crucial part of the whole automated assembly system, in particular – vision subsystem. It provides the software with the information where all the components of the modules are situated in the space and theirs orientation. Within this information the software is able to calculate where to move each component of an assembling module [**?**].

An important task performed by PSAuto is the determination the *location* and *planar orientation*. By definition, planar orientation is the rotational orientation of the sensor in the horizontal (X-Y) plane of the sensors during the assembly process. This is generally achieved in two basic steps: the independent determination of *localised*

positions and orientations of the four markers at the corners of the sensor and a global fit to these positions from which the final position and orientation of the entire sensor is extracted [**?**].

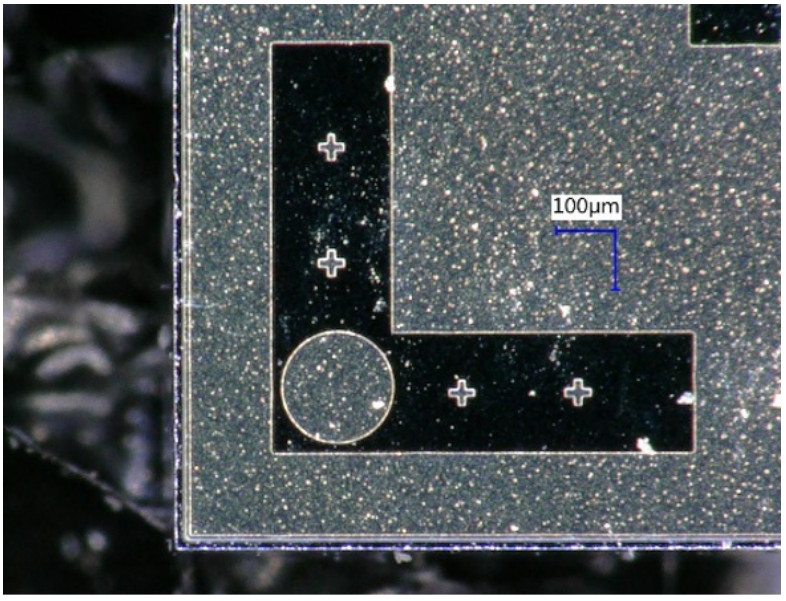

FIGURE 3.4: Fiducial marker on one of the corner of a PS sensor.

This is done by processing the images of *fiducial markers* at the corners of the sensors (Figure **??**) acquired by the vision system with a Pattern Recognition algorithm. The markers are precisely positioned with respect to the strips or pixels or the sensors. Hence precise alignment of the markers ensures precise alignment of the pixels or strips.The algorithm takes raw images as input and returns the position and orientation of fiducial markers located at the corners of the PS sensors. The pattern recognition algorithm utilises the *OpenCV* package. The steps comprising the pattern recognition are now outined:

1. Pre-processing of raw image.
2. Determination of positions and planar orientations of fiducial markers.
3. Location of other corners and extraction of final position and orientation.

3.2.1 Pre-processing of raw image

The raw images from the camera are first converted from colour to black and white which is known as *grayscaling* in OpenCV. The pixels comprising the grayscaled image contain intensity information only in the form of a single number ranging from 0 to 255 describing the darkness of a shade of gray as opposed to the intensity and colour information contained in the pixels of a colour image. As the corner markers are based on simple shapes and not colours, the colour information is not helpful and is thus disregarded. The grayscaled image is then converted to a *binary* image in which each pixel is either black or white in a process known as *thresholding*. Thresholding simply converts each

pixel of the image into a white(black) pixel if the intensity of the pixel is above(below) a pre-defined threshold (). Thresholding serves to reduce the differences between images of identical sensors due to random *noise* arising from dust and random differences in the surfaces of the patterned sensors. Examples of grayscaled and thresholded images of the same corner marker of a dummy PS sensor is shown in Figure **??**. The optimal threshold will depend on the ambient lighting conditions around the assembly area and and the contrast of the fiducial markings on the sensors. Currently, typical values of threshold are around 90.

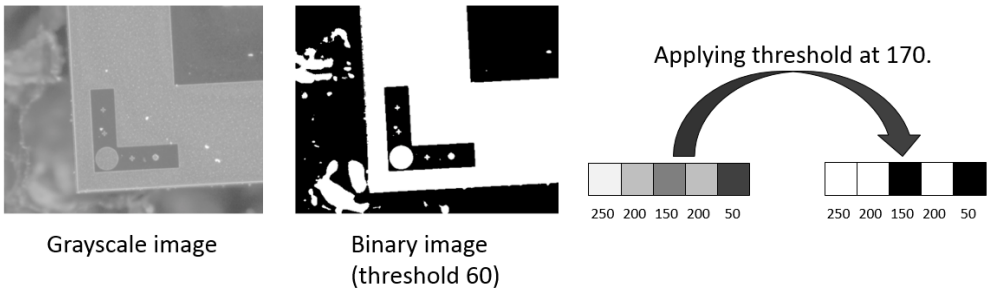

FIGURE 3.5: Threshold application on fiducial marker of dummy PS module.

3.2.2 Determination of positions and planar orientations of fiducial markers.

The position and orientation of the fiducial marker within the thresholded image is determined using a standard image processing technique known as *template matching*. In template matching, parts of an master image that closely resemble a template image are located. In this case, the master image corresponds to the thresholded image of the fiducial marker and the template image corresponds to a thresholded image of a fiducial marker.

Template matching proceeds by iteratively superimposing the template image at each point of the master image and calculating a metric which describes the similarity of the template image and the portion of the master image with which it coincides. The OpenCV package provides multiple options for the metric. Similar results are observed for each possible metric with the chosen metric based on the normalised squared difference between the intensities of coincident pixels of the master image and superimposed template:

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2}{\sum_{x',y'}\sqrt{\sum_{x',y'}T(x',y')^2 \cdot \sum_{x',y'}I(x+x',y+y')^2}}$$

where I denotes *master image*, T – *template image* and R – *resultant metric*.

This metric is referred to as CV_TM_SQDIFF_NORMED in OpenCV. The point in the master image where the metric reaches a minimum represents the most probable location of the fiducial marker. In Figure **??** the determination of marker position with

template matching and its result using images of the dummy sensor is illustrated. The most probable location of the marker as determined by the algorithm is indicated by the white rectangle. The observed location matches closely with expectation.

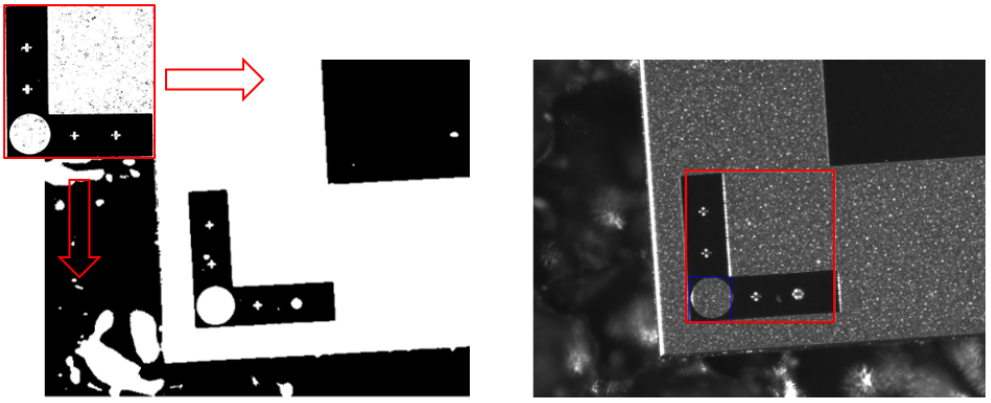

FIGURE 3.6: The technique of template matching is illustrated on the left image. The red arrows indicate the iterative calculation of a metric at each point of the master image. The result of a template matching routine on using test images is shown on the right image. The most probable location of the marker in the master image is outlined by the red rectangle.

In order to deduce the orientation of the marker in the plane transverse to the optical axis of the camera, the matching procedure is repeated iteratively with different rotational transformations applied to the master image. For each iteration, the minimal value of the metric is recorded with the minimal metric value across all iterations denoted as α. The planar orientation of the sensor is estimated as $-\alpha$. In Figure **??** a schematic illustrating the determination of the orientation is shown. A graph of the resultant minimised metric values versus the size of the angular transformation applied to the master image is shown on the right image of the Figure **??**. The graph corresponds to a test extraction performed with images of a dummy sensor where the sensor in the master image had a planar orientation of ≈ 3.5 degrees. A clear minimum is observed at ≈ 3.5 demonstrating the method's validity. More precise determination of the sensor orientation can be achieved when factors such as ambient light conditions, image focus and marker design are further optimised. The best accuracy reached during tests was ≈ 0.025 degrees.

3.2.3 Location of other corners and extraction of final position and orientation.

The procedure described in step 2 is repeated at each sensor corner. The planar orientations determined at given corner are used to set the direction of movement needed for the motion stage to automatically travel to an adjacent corner. The final position and orientation of the sensor is determined by a χ^2 fit to the four (x,z) points. The orientation determined from the fit is cross-checked with the estimations of the orientation at the

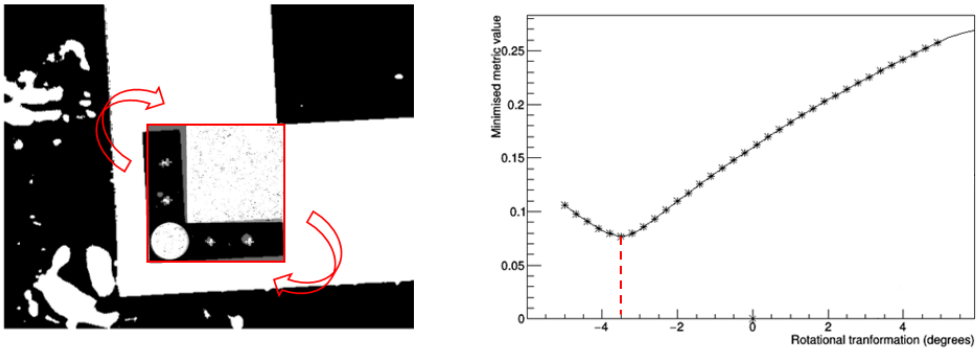

FIGURE 3.7: A schematic illustrating the estimation of the sensors planar orientation is shown on the left image. The red arrows indicate the iterative rotational transformations applied to the master image. graph of the minimised metric value versus the angular transformation applied to the master image is shown on the right image. A clear minimum at ≈ 3.5 degrees is observed.

corners which accuracy could be not far from χ^2 deduced. If there is agreement between the fit and four corner orientations, the fit results are used.

Detailed Flow Chart of the pattern recognition is shown in the Appendix **??**.

3.3 Application functionality

The main window of the application has several buttons and check boxes on top of the window and a number of tabs including: Finder, Threshold, Assembly, Autofocus, Motion Manager and others.

1. *Finder.* A simple tab the only purpose of each is to show the last acquired image of the camera. This tab also provide a possibility to save current image.

2. *Threshold.* The result of the Threshold operation highly depends on the ambient light thus each new test required threshold value calibration before starting it. This tab is specifically created to control the applying threshold value. It provides the possibility to configure this value and gives an immediate respond by applying Threshold operation on the last acquired image and showing the result under the grayscale original image. The screenshot of the Threshold tab is shown on the Figure **??**.

3. *Assembly.* The primary tab of the application (Figure **??**). It contains four image boxes on the left, control tools on the right and motion stage real time status information on the bottom. Left-top image box contains last acquired image. Left-bottom image box – Thresholded (binary) image of it. Right-bottom image box shows the template image. Finally, right-top image box represents the graph of the last pattern recognition metric distribution of the template matching metric along theta value.

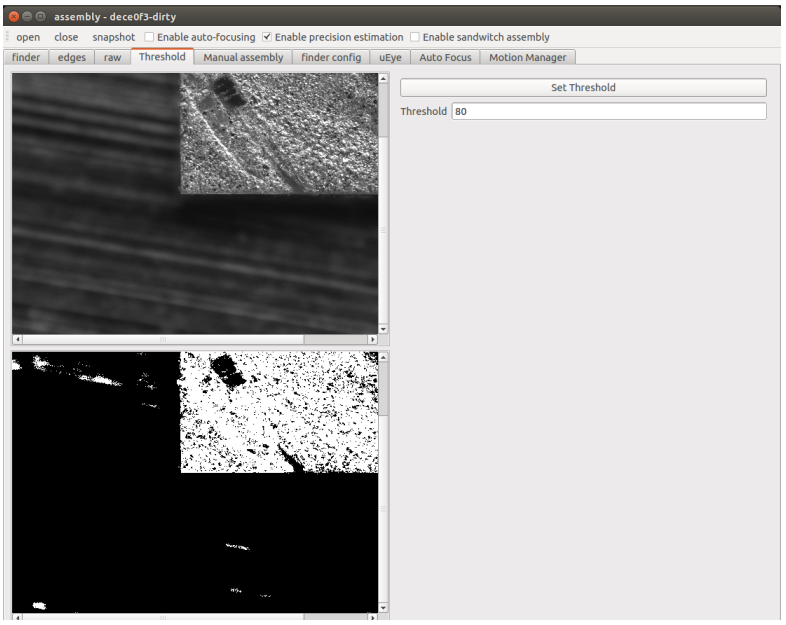

FIGURE 3.8: Screenshot of the Threshold tab of the PSAuto application.

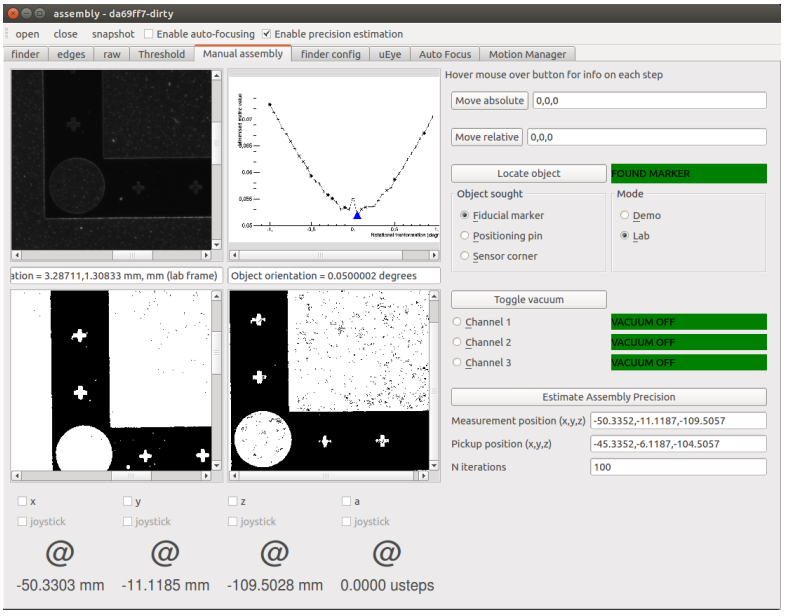

FIGURE 3.9: Screenshot of the Assembly tab of the PSAuto application.

On the right side of the tab there are a number of control tools (beginning from the top one):

- Two buttons to move the pickup tool to the absolute or relative position which is written in the forms beside buttons respectively.
- A set of radio-buttons choosing type of markers on samples and the pattern recognition mode.
- A set of buttons to control the relay card of vacuum lines. Beside each button there is a real time indicator of the current vacuum line status.

- The last tool in this tab provides the possibility to conduct tests of the system. It consists of several forms for input test information and a button to start the test.

4. *Auto focus.* Another nice tool of the application is autofocus. It place an essential role in the pattern recognition as soon as its accuracy directly depends on the quality of acquired image. There are two features provided by this tab: find the focus position of the camera and move to this position. To find the focus position of the camera it is required to set the step size along Z-axis and the number of steps. As smaller the step is, the more precise focus position will be identified. The idea of auto focus detection is similar to best theta detection in the pattern recognition. For each step position it saves the metric data representing the blur of the acquired image thus in the end the software can find the position with least blur on the image comparing saved metrics for each image.

   This algorithm can also be used for another purpose. As soon as the camera is constantly fixed on the robotic arm, it is possible to make relative Z-axis distance measurement. For instance, that can be used for measuring the glue layer thickness of an assembled modules.

5. *Motion Manager.* This tab provides the control of the motion and rotation stages. Among its functionality are:

   - Independent control of X,Y,Z axis and rotation stage.
   - Self-calibration.
   - Real time monitoring of the status of stepper motors of each axis and rotation stage.

Chapter 4

Precision estimation tests and the first prototype

The automated assembly system has a number of properties in terms of precision:

1. Motion stage movement repeatability.
2. Image acquiring repeatability.
3. Precision of pattern recognition.
4. Possible movements of a sensor while picking them up and down with the vacuum pick up tool.

In order to investigate this properties a series of tests were done.
Real sensors will be very thin (around 200 um). This fact makes them very fragile. Even though dummy sensors, which will be used for further experiments, is a bit thicker (around 300 um), they are still too fragile for the first tests, because the bottom surface of the pick up tool and the plane underneath testing samples are not yet parallel enough. That is why for the first pick up and down tests we used glass samples. They have the same dimensions and represent close enough the properties of silicon sensors. Moreover, they are much cheaper, so that in case of test failure (sample break) it will not be such a big problem as if silicon sample crashes. Despite all mentioned above, none of glass samples where crashed.
Even though we did not do the pick up test with silicon samples, there is still an opportunity to get some information of the pick up and down precision of the silicon samples without direct tests with them. By making a full range of tests with glass samples, we will be able to say how pick up and down influences the precision. Based on this results we will be able to approximately predict the precision of pick up and down tests with silicon samples. Later, when parallelness of the bottom surface of the pick up tool and samples will be provided, we will be able to confirm the results of the prediction.

4.1 Pattern recognition precision tests

For investigation of the pattern recognition precision the following tests were done. During these tests samples were not moved, so that the additional errors by vacuum pick up and down can be excluded.

4.1.1 Pattern recognition on the painted corner of a glass dummy

In the very first test we investigated the pattern recognition on the corner of the sample. Thin pieces of glass with a silver painted corner (Figure **??**) were used for the tests as an approximation of a silicon sensor. Silver painted corner was used as a marker for pattern recognition to be found in the acquired image.

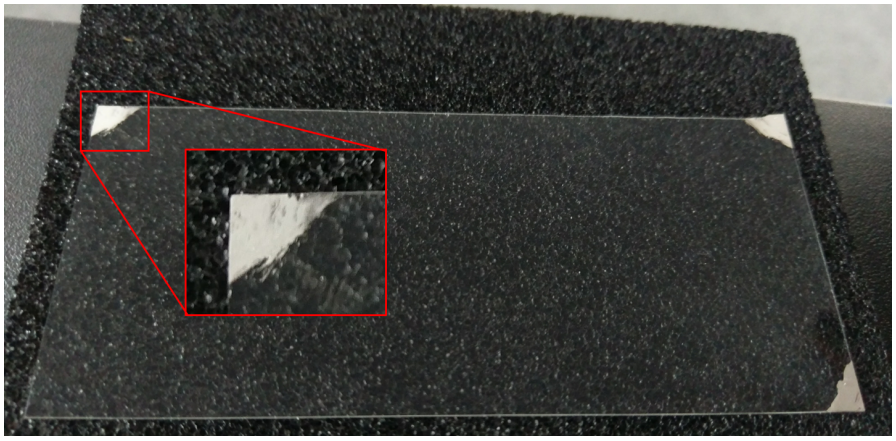

FIGURE 4.1: Glass sample with silver painted corner.

The step-by-step outline of this test is listed below:

1. Move to the image acquiring position.
2. Acquire image and run pattern recognition.
3. Move aside for 5 mm in all three axes.
4. Move to the image acquiring position.
5. Acquire image and run pattern recognition.
6. Save data of the current iteration and go to the next one.

After each step software saves the difference between measured coordinates before and after moving the arm with the camera. The distributions of these values are showed in Figure **??**, Figure **??** and Figure **??** for X-axis, Y-axis and theta, respectively. The test had 100 iterations done in a row.

Looking at the Figures **??** and **??** one can see that the X, Y detection of the pattern recognition has precision of around 1-2 um, while the theta detection results do not look so precise. There are several reasons of such behaviour. The main one them is shown on the Figure **??**.

Silver painted surface is not flat in 10 um scale. Due to this roughness different amount of light reflects to the camera from different points along the painted surface.

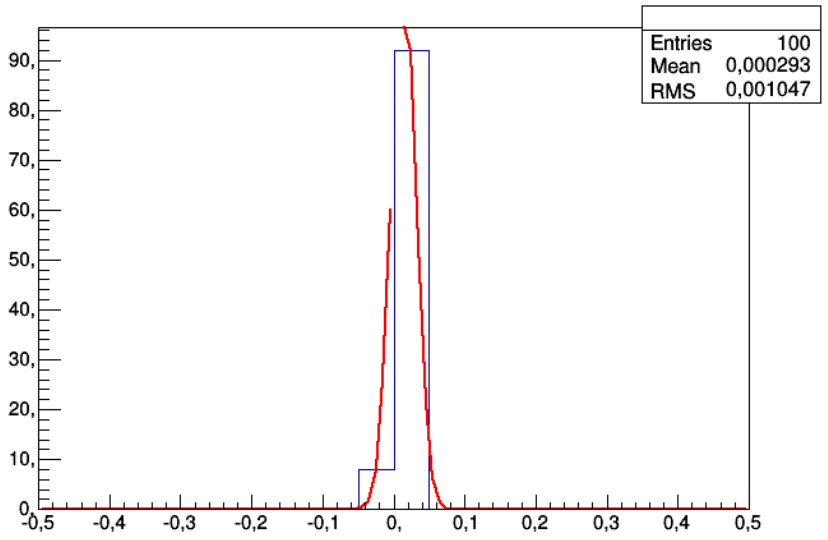

FIGURE 4.2: Distribution of the difference between detected X coordinate of the master image before and after moving the arm in each iteration. $\Delta X \approx 1\ um$.

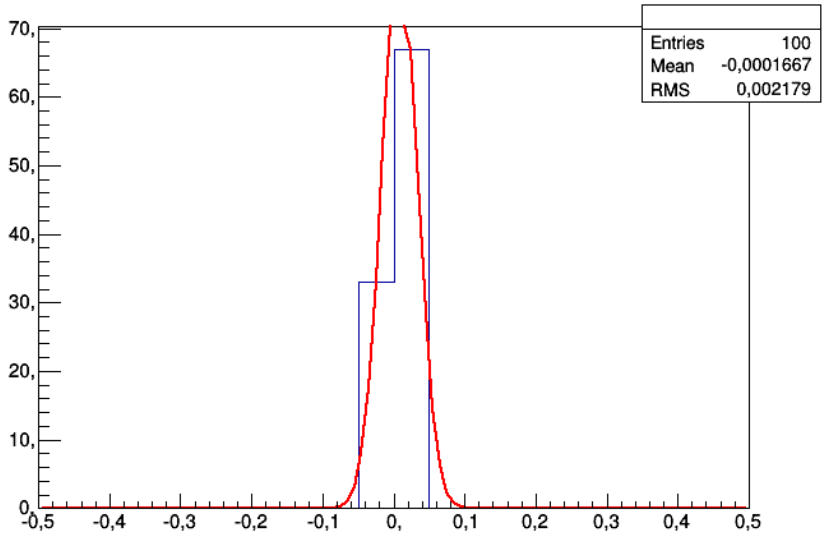

FIGURE 4.3: Distribution of the difference between detected Y coordinate of the master image before and after moving the arm in each iteration. $\Delta Y \approx 2\ um$.

That is why the painted corner contains various shades of grey, which in some points are darker, than the table underneath the sample (background). All these result into the picture one can see in the Figure **??**. This kind of pictures has random distribution of dark areas on it. That is why the pattern recognition algorithm has such error while comparing two pictures (master template and acquired image) with random distribution of black areas. Moreover, this kind of tests lasts around one hour, which is long enough for the sun to change the ambient light in the laboratory. Even though all reasonably possible measures were done to prevent such effect, the acquired image is very sensitive

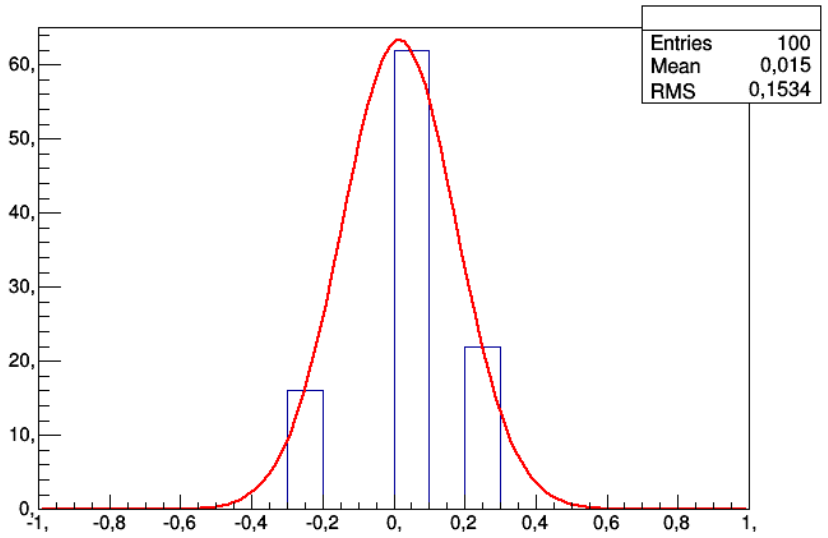

FIGURE 4.4: Distribution of the difference between detected angle orientation of the master image relatively to the acquired image before and after moving the arm in each iteration. $\Delta theta \approx 0.15$ *degree*.

for light. For example, the effect of the sun light can results in the threshold variation for 20 units (the color depth is 256) even with covered window in the laboratory.

FIGURE 4.5: The view of the corner after applying the Threshold.

4.1.2 Pattern recognition on the marker of the dummy sensor

The same test, but with dummy silicon sensor and real marker on it, was done. Before the test marker was aligned as much close to zero degrees as possible. At the Figure **??** one can see that the edge of the marker after applying Threshold is almost

perfect (+/- one pixel). This fact itself is already a proof that Threshold step of pattern recognition is feasible.

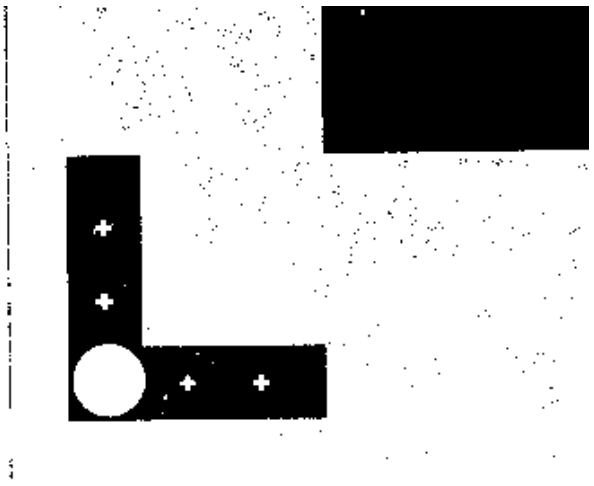

FIGURE 4.6: Sensor marker after applying Threshold.

The Distribution of X, Y coordinates and theta shows better results than with painted corner, which was expected. For X and Y it is less than a micron, which is already at the limit of camera resolution. $\Delta theta$ is one order of magnitude better than with painted corner — $\approx 0.02 degree$.

A screenshot of the application during the test is shown on the Figure **??**. On the pattern recognition curve one can see that at 0 degree there is an unexpected short upward shot. This peak is not a fluctuation and it is not consist of only one point in the plot. As the scale increases, more points form this peak appear. It was not observed in the previous test just because the theta step was one order of magnitude bigger.

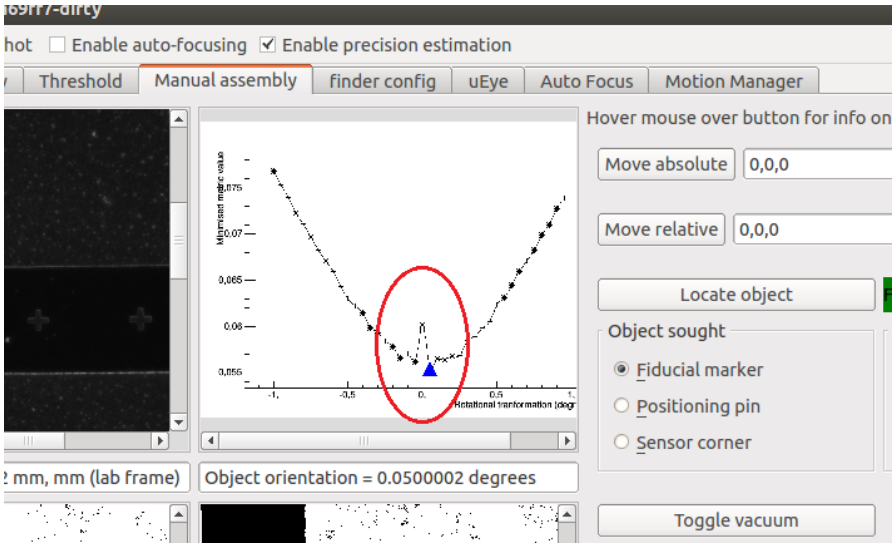

FIGURE 4.7: Screenshot of application during precision estimation test with dummy silicon sensor and marker on it.

4.2 Pickup precision tests

Next set of tests were oriented to explore the effect of pickup and setdown on the precision of the system. In other words, will this process move the sample or not in XY plane. Due to the risk of breaking fragile silicon samples, we used glass ones.

4.2.1 Pickup precision tests without assembly platform

The first pick-up/down test was done without assembly platform. The step-by-step outline of this test is listed below:

1. Move to the image acquiring position.
2. Acquire image and run pattern recognition.
3. Move to pre-pickup position.
4. Move to pickup position.
5. Turn negative vacuum on.
6. Move up.
7. Move down.
8. Release vacuum.
9. Move down.
10. Move to pre-pickup position.
11. Move to the image acquiring position.
12. Acquire image and run pattern recognition.
13. Save data of the current iteration and go to the next one.

One can notice the step of moving first to pre-pickup position before going to the pickup position and visa versa. This step is essential. The motion stage provides equal speed in all three axes, so when it receives a command to move to some position it starts to move with equal speed in each of three directions towards the destination simultaneously. As soon as destination in one axis is reached, it obviously stops moving in this axis while moving in other axes is going on. Therefore there is an unlike situation when the robotic arm reaches the sample in Z-axis while X and Y axes would still keep moving, which may cause damage or even break a sample. To prevent such situation we added the step of moving to pre-pickup position.

The results of the test showed movement of the sample, which could be noticed with a naked eye. In order to minimize this movement we decided to make a touch test – the same as pickup and setdown, but without vacuum. Such kind of test can show the contribution of the touch to the sample movement in the pick-up and -down test. The results if this test was quite similar to the previous test, which means that the touch itself contributes the most to the sample movements in the pick-up and -down test. The interesting fact of this movement is its trend. On Figure **??** one can see X and Y movement trends with respect to iteration number in the touch test. This plots show that the movement is not random and it is more or less constant both in value and direction.

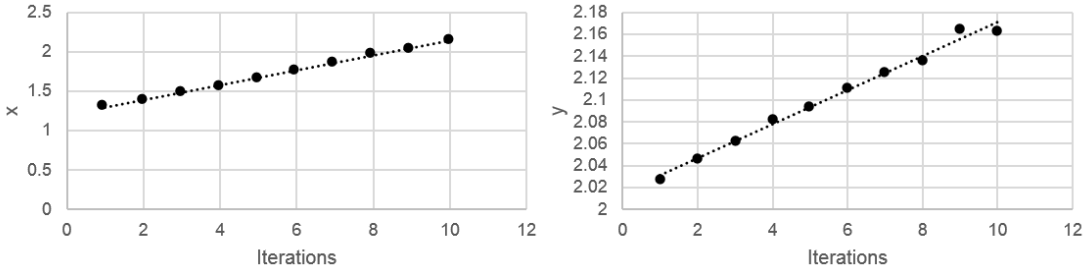

FIGURE 4.8: X and Y movement trends with respect to iteration number in the touch test.

The most probable reason for such effect is that pickup tool surface is not parallel to the table, where samples lay waiting for being picked-up. This fact perfectly corresponds to the constant movement of a sample to the one direction. Unfortunately, it is very complicated to align this surface parallel enough to make sample movement negligible. Another way to avoid this movement is to use an assembly platform, which can hold samples with vacuum underneath so they would be fixed.

4.2.2 Pickup precision tests with assembly platform

The step-by-step outline of the pickup and setdown test with the assembly platform is very similar to the one without it. The only difference is that samples are always fixed by the underneath vacuum and are only released, when the vacuum from pick-up tool is provided, so it can lift the sample upwards.

1. Move to the image acquiring position.
2. Acquire image and run pattern recognition.
3. Move to pre-pickup position.
4. Move to pickup position.
5. Turn pick-up tool negative vacuum on.
6. Turn assembly platform negative vacuum off.
7. Move up.
8. Move down.
9. Turn assembly platform negative vacuum on.
10. Turn pickup tool negative vacuum off.
11. Move down.
12. Move to pre-pickup position.
13. Move to the image acquiring position.
14. Acquire image and run pattern recognition.
15. Save data of the current iteration and go to the next one.

Distribution of the difference between detected X coordinate of the master image before and after moving the arm in each iteration is shown on the Figure **??**.

As one can see from the Figure **??**, $\Delta X \approx 4\ um$, which is very close to the limit of 1 um of the pattern recognition itself without touching the sample. Moreover, the X and

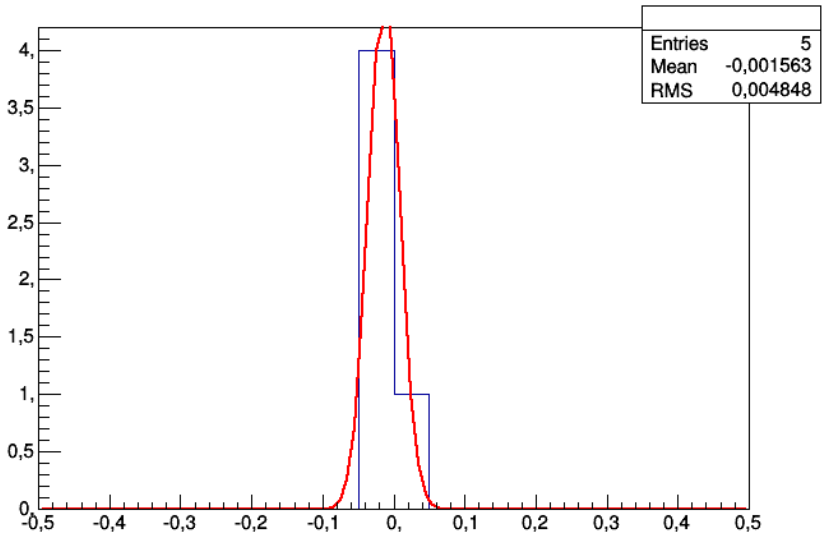

FIGURE 4.9: Distribution of the difference between detected X coordinate of the master image before and after moving the arm in each iteration. $\Delta X \approx 4\ um$.

Y coordinates show no trend with respect to the iteration number. Taking into account these results, it is possible to say that using an assembly platform can fix samples enough tight, so it is possible to neglect their movements.

4.3 First assembled prototype

After all tests mentioned above it was possible to assemble the very first prototype of the module. To test the precision of the assembling module for the first time it was decided to use simplified assembly algorithm, which would provide maximum precision only on one corner. Schematic view of the first module prototype is shown on the Figure ??.

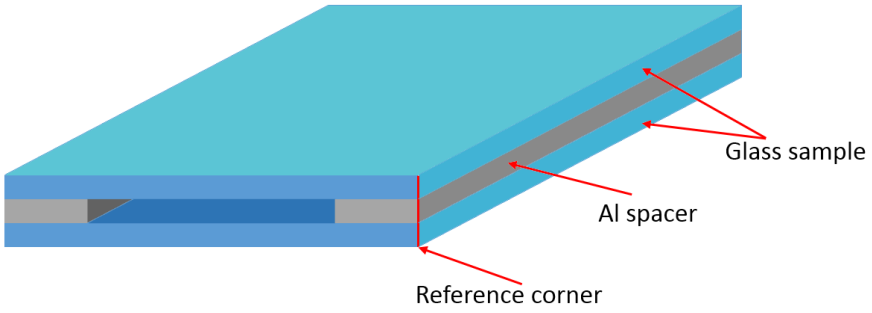

FIGURE 4.10: Schematic view of the first module prototype.

The step-by-step outline of the assembly process is listed below:

1. *Place and align top sample parallel to X-axis of motion stage and lift it up with pick-up tool.* Place top sample on the assembly platform and fix it with vacuum. Find with the pattern recognition its orientation in space. Rotate the platform to align top sample parallel to the edge of the pick up tool. As soon as theta recognition do not provide enough accuracy, the following method was used. Move camera for 5 cm along the sample edge and measure the shift of the sample edge. Knowing two cathetus of it is possible to find the angle of the right triangle, which equals to the angle between sample edge and x-axis of motion stage. Rotate the assembly platform for this angle. To sum-up, first – rough top sample orientation estimation with direct pattern recognition, second – precise calculated sample orientation with Pythagorean theorem. Now the top sample is aligned enough parallel to the X-axis of the motion stage. Next, save position of the reference corner with a pattern recognition. Finally, pick-up top sample with pick-up tool and leave it there.
2. *Place Al spacers.* Put Al spacers to the inserts, fix them with bottom vacuum and align them parallel to the X-axis of the motion stage. Absolutely the same way as with top sample. Save the coordination of the reference corner with pattern recognition. Comparing coordinates of the reference corners of Al spacer and top sample, move the robotic arm in XY plane of the motion stage so they would match.
3. *Glue Al spacers to the top sample.* Place the glue on Al spacers and move robotic arm with attached top sample down. It is important to move robotic arm down for exactly correct distance. This topic was discussed in details in the Module Assembly Chapter. Wait glue to be cured and lift glued structure (Al spacers and top sample) from assembly platform.
4. *Place and align bottom sample parallel to X-axis of motion stage.* Place bottom sample on the assembly platform and fix it with bottom vacuum. Make the parallel alignment to the X-axis with the same method as with top sample. Find XY coordinates of the reference corner with the pattern recognition. Compare them to XY coordinates of spacers and top sample. Move the robotic arm with attached structure glued before to match reference corners of all parts of the module.
5. *Final gluing.* Put the glue on the bottom sample and move robotic arm down for correct distance to glue entire prototype. After glue is cured the assembly process is finished.

The glued prototype is shown on Figure **??**.

We used simplified algorithm for the first prototype assembly aligning only one corner. The quality of assembled prototype is very promising. It meet all necessary requirements. On the Figure **??** one can see the photo of the reference corner from the microscope. Pictures form other directions look very similar having approximately the same accuracy.

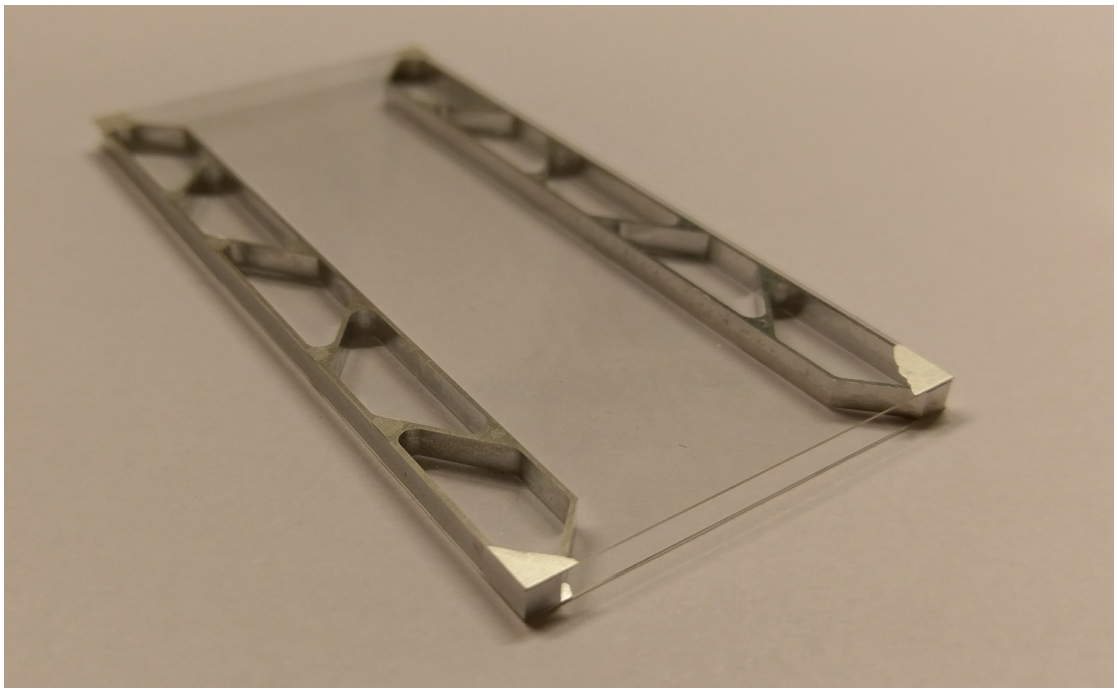

FIGURE 4.11: Photo of the first assembled prototype.

This assembled prototype proved the feasibility of the automated assembly and showed lots of issues to be solved in future. Even though it has only silver pained corner, not precise lithography, this prototype showed very good results in terms of assembling accuracy.

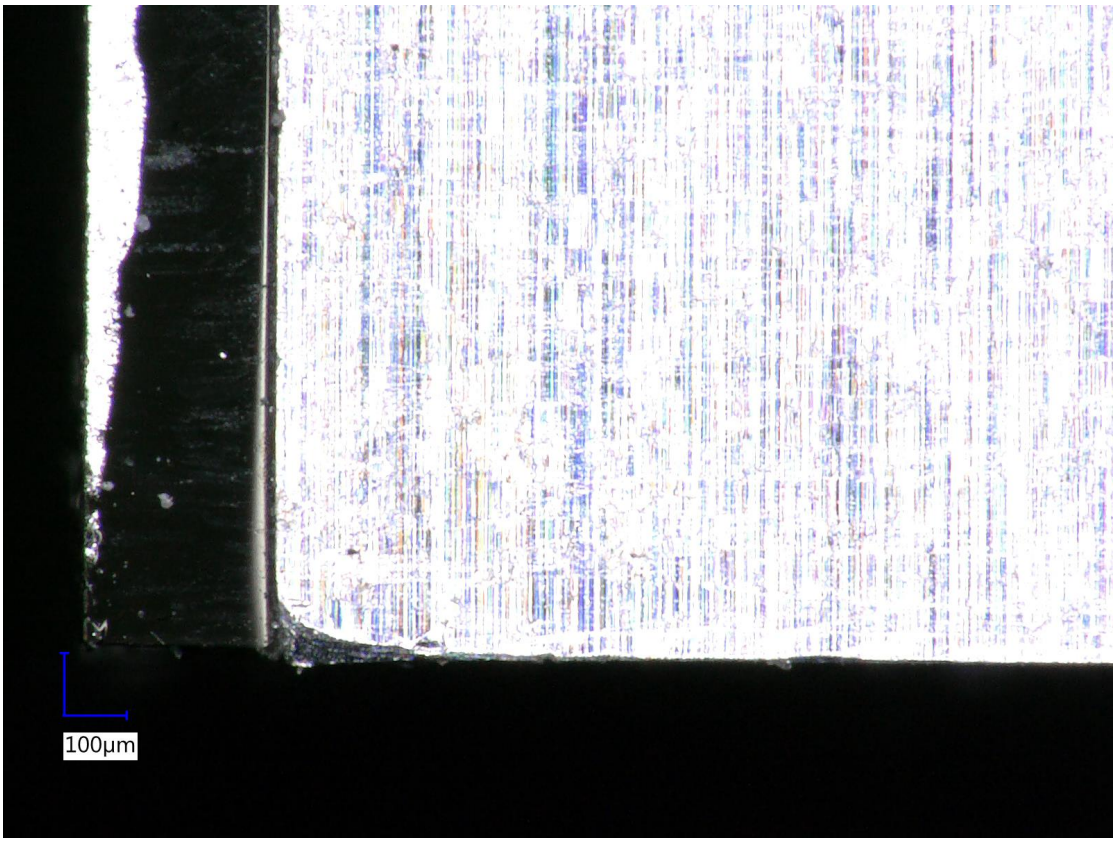

FIGURE 4.12: Microscope photo of the reference corner of the first prototype. Misalignment of the components is around 20 um.

Chapter 5

Conclusion

During my work on the project of Automation of CMS Phase II Tracker module assembly the following results were reached:

- Proved the possibility of an automated module assembly with the required accuracy.
- The first prototype is assembled demonstrating the potential of the system.
- The PSAuto software is modified.
- The assembly platform was designed, manufactured, implemented to the system and successfully tested.
- The fast adhesive technique is investigated and successfully implemented.
- Lots of tests were done showing abilities of the system and its accuracy.

Appendix A

Appendix

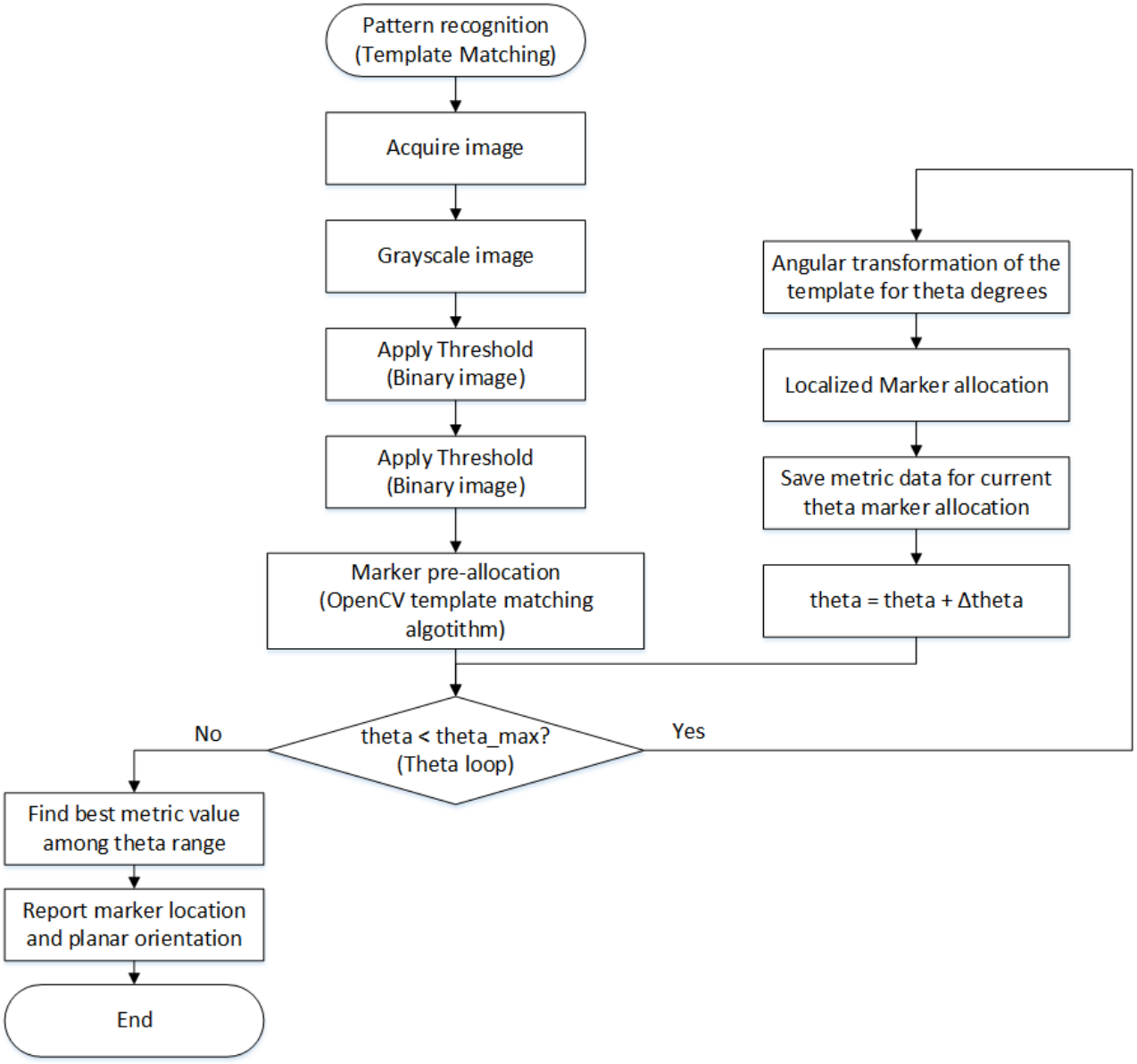

FIGURE A.1: Pattern recognition Flow Chart.