THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

3D Rigid Object Scanner by Kinect

Master Degree Thesis

ISMAIL BOZKURT

ISTANBUL, 2015

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

GRADUATE SCHOOL of NATURAL and APPLIED SCIENCES
COMPUTER ENGINEERING

3D RIGID OBJECT SCANNER by KINECT

Master Degree Thesis

ISMAIL BOZKURT

Thesis Advisor: Ph.D TARKAN AYDIN

ISTANBUL, 2015

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
COMPUTER ENGINEERING

Name of the thesis: 3D Rigid Object Scanner by Kinect
Name/Last Name of the Student: Ismail Bozkurt
Date of the Defense of Thesis:

The thesis has been approved by the Graduate School of Natural and Applied Sciences.

Signature

Assoc. Prof. Nafız ARICA
Graduate School Director

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Arts.

Signature

Asst. Prof. Dr. Tarkan AYDIN
Program Coordinator

This is to certify that we have read this thesis and we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Arts.

Examining Comittee Members	Signature
Thesis Supervisor Ph.D Tarkan AYDIN	---------------------------------
Thesis Co-supervisor Ph.D Övgü ÖZTÜRK ERGÜN	---------------------------------
Member Ph.D Ömer KORÇAK	---------------------------------

ACKNOWLEDGEMENTS

“Theory guides. Experiment decides”

An old saying in science, seen attributed to many different persons.

First thanks go to my supervisors Kemal Egemen Özden and Tarkan Aydin, for all patient lessons and theoretical advises in a perfect scientist-assistant and friendly teacher-student atmosphere. Moreover, I am thankful for all the time they patiently spent on this project.

Lastly my deepest thoughts go to my family, especially to my mother Zeynep.

Istanbul, 2015 İsmail Bozkurt

ABSTRACT

3D RIGID OBJECT SCANNER by KINECT

Ismail Bozkurt

Computer Engineering

Thesis Advisor: Ph.D Tarkan Aydin

December 2015, 59 pages

3D Object scanning is the result of scanning the environment by a camera, and gaining the scanned environment as a map. Because of the ability of scanner cameras (such as Kinect) can lead various Computer Vision applications. For Instance Robotic Applications, due to the fact that GPS is not available for indoor environments rigid object scanning brings a solution for indoor robotic applications. In other words, mapping the indoor environment and navigating. Also outdoor applications can need the low distance environment detection for some reasons such as vehicle following distance for unmanned vehicles. Same thing can apply for unmanned air vehicles. In addition the ability of 3d scanners inspire many applications like free controller games, augmented reality applications/games etc.

In this project, building a model map from 2D image feature detection to registration of 2 set point cloud is shown by using the following method, algorithms and hardware: SURF feature detection, feature matching, RANSAC, Rigid Registration and Kinect. Lastly, local feature matching process is enhanced by a measure called Depthscale.

Keywords: 3D Registration, Kinect, Local Image Features

ÖZET

KINECT ile 3 boyutlu obje tarayıcısı

Ismail Bozkurt

Bilgisayar Muhendisligi

Tez Danışmanı: Yrd. Doç. Tarkan Aydin

Aralık 2015, 59 sayfa

3 boyutlu nesne tarama olayı, ortamı bir kamera aracılığıyla tarama ve bu taranmış veriyi bir takım yöntemlerle bir bir 3 boyutlu nesne elde etmekten oluşur. Kinect gibi tarayıcı kameraları birçok çeşitli Computer Vision uygulamarına öncülük ederler. GPS'in kullanılamadığı kapalı ortamlarda, 3 boyutlu nesne tarayıcıları, Robotik uygulamalara çözüm getirebilirler. Daha detaylı olarak, ortamın bir haritasını çıkarıp bu ortama göre yön bulma konusunda yardımcı olur. Hatta açık ortamlarda yakın mesafe görüşü gerektiren uygulamalarda da tarayıcılar kullanılabilir. Örnek vermek gerekirse araç takip mesafesi uygulamaları, ayrıca insansız hava araçları da bu teknolojiyi kullanabilir. Bunlara ek olarak kontrol cihazsız oyunlar, artırılmış gerçeklik uygulamarlı bunlara örnek verilebilir.

Bu projede 2 boyutlu resimlerden 3 boyutlu bir modelin nasıl çıkarılacağı gösterilmiştir. Projede algoritma ve metod olarak SURF feature tespit etme ve eşleştirme, RANSAC ve Katı cisim eşleştirme, donanım olarak da Kinect kamera kullanılmıştır. Son olarak local feature match surecinin iyileştirilmesi icin deneyler yapilmıştır.

Anahtar Kelimeler: 3 boyutlu birlestirme, Kinect, Local Image Features.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DS: Depthscale

ICP: Iterative Closest Point

PCA: Principal Component Analysis

RANSAC: RANdom SAmple Consensus

ROC Curve: Receiver operating characteristic Curve

SDK: Software Development Kit

SIFT: Scale Invariant Feature Transform

SURF: Speeded Up Robust Feature

SVD: Singular Value Decomposition

1 INTRODUCTION

3D object scanning is a challenging application. So far, there are different rigid application scanner applications and systems on the market. By Kinect's release the computer vision market had a new scope for developing applications. The main reason for that is Kinect has quite advantages. Firstly Kinect has an affordable price. Secondly and more importantly it provides depth information for every single frame, which makes the run project faster and easier. On the other hand, as a disadvantage the depth data that is provided by Kinect can contain noisy data. Therefore elimination of noisy depth is unavoidable. By combining these ups and downs a 3D rigid object scanner application became possible.

Point cloud registration is not a new topic of computer vision. There other algorithms and approaches like standalone PCA and ICP to achieve it.

PCA [14] is often used in classification and compression techniques to project data on a new orthonormal basis in the direction of the largest variance. A rough transformation matrix can be obtained by using covariance matrices of consecutive point clouds.

Standalone ICP is very simple solution compared to my solution. (Check section 2.7). However ICP does not perform well against noisy data and it tries to register point clouds in an iterative way, which might be waste of resources. There are two variants of ICP, point to point and point to surface. Point to point ICP [14] is the simple one. It uses the paired matched points to estimate transformation between set of points. As you see, since it requires fixed pair of matches, it is hard to implement and has no room for noise. Point to surface ICP [14] assumes point clouds are locally linear, instead of decreasing Euclidean distance it tries to minimize scalar projection of this distance.

In this report you will see how to build a map of the scanned environment by the Kinect. To be more precise, we firstly use the Kinect's outputs (rgb and depth frames) to detect and match some common features on 2D consecutive frames. After the matches found we will try to eliminate noisy matches in 2D. Secondly we try to find 3D corresponding points of these matches and estimate the transformation matrix between them. In 3D we again eliminate noisy matches (since the Kinect depth data is noisy). To conclude, we will try to register the consecutive point clouds. Lastly we will try to improve matching process in 2D by introducing a new measure called Depthscale.

1.1 GOALS

Main goal is building a map of the 3D objects by using Kinect RGB and depth data. Firstly take consecutive depth and grey scale image from Kinect. Then by SURF algorithm detecting matched descriptor points. Then elimination of noisy matched descriptors by RANSAC. Secondly creating the point cloud of the frames and finding matched descriptor points. Lastly, by using matched descriptor points discovering the transformation matrix for the second frame. These steps are followed to build 3D map.

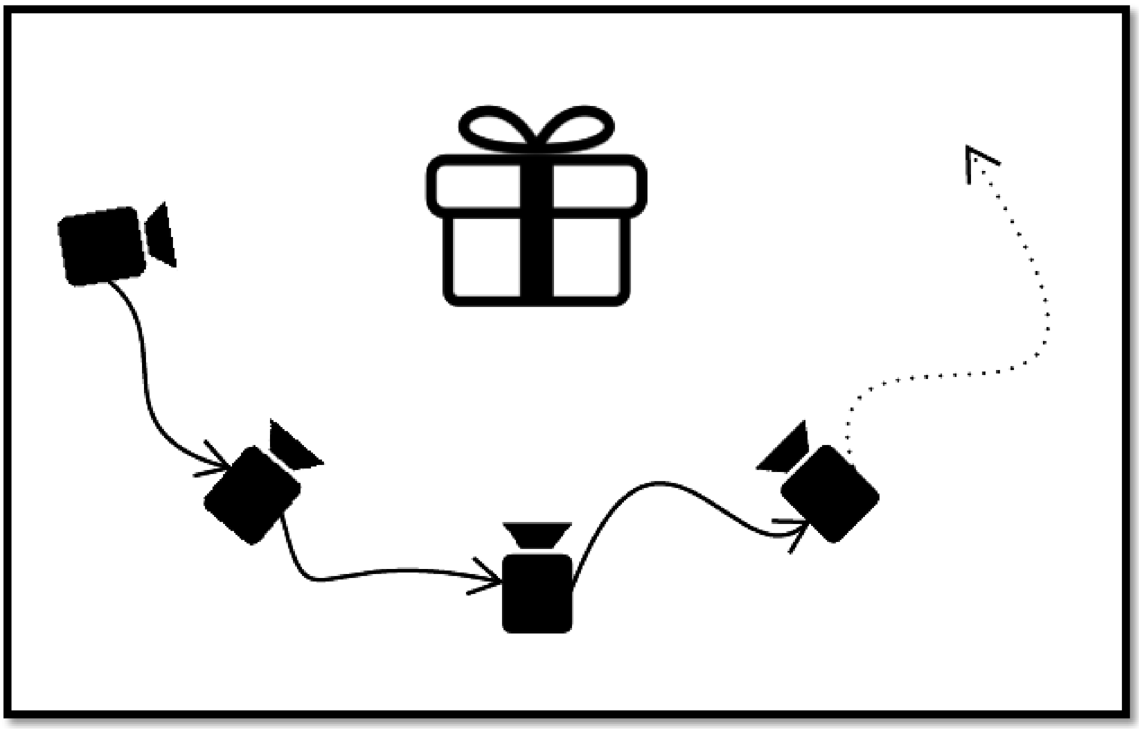
Finally, noise reduction. Because of the new Kinect depth output data, there are many possible new ways to approach well to new computer vision solutions and algorithms. So in this report several new error reduction proposes are explained.

By building a 3D rigid object scanner, different type of systems can be built and carry technology forward. Some of these technologies are unmanned vehicles and aircrafts, house cleaning robots, augmented reality applications and so on.

1.1.1 Setup

The setup consists of a computer and a Microsoft Kinect. One of the biggest challenge was to estimate camera transformation between Kinect frames. In other words, there are no pre-defined camera path or fixed camera coordinates.

Figure 1.1: Application setup



As demonstrated in figure 1.1, the application tries to estimate camera movement and merge/align the Kinect frames to accomplish registered point clouds. As mentioned above, we do not use any estimation or specified camera path during the scan process, though we try not to exceed the 30 degree angle between shots.

1.2 OUTLINE

The rest of the document is structured as follows:

Section 2 presents the background. Which hardware, algorithms and approaches are used in this project is explained.

Section 3 presents the 2D space pipeline of the scanner, which are feature matching between consecutive frames and how to use 2d data in 3d consecutive depth frames.

Section 4 presents the 3D space pipeline, that how to create a map by calculating camera movement and rotation. In other words how to calculate 3D transformation between consecutive point clouds. It consists of 3d point cloud alignment/registration via several algorithms.

Section 5 presents the Depthscale measure. How it works, benefits and comparison with Lowe's measure

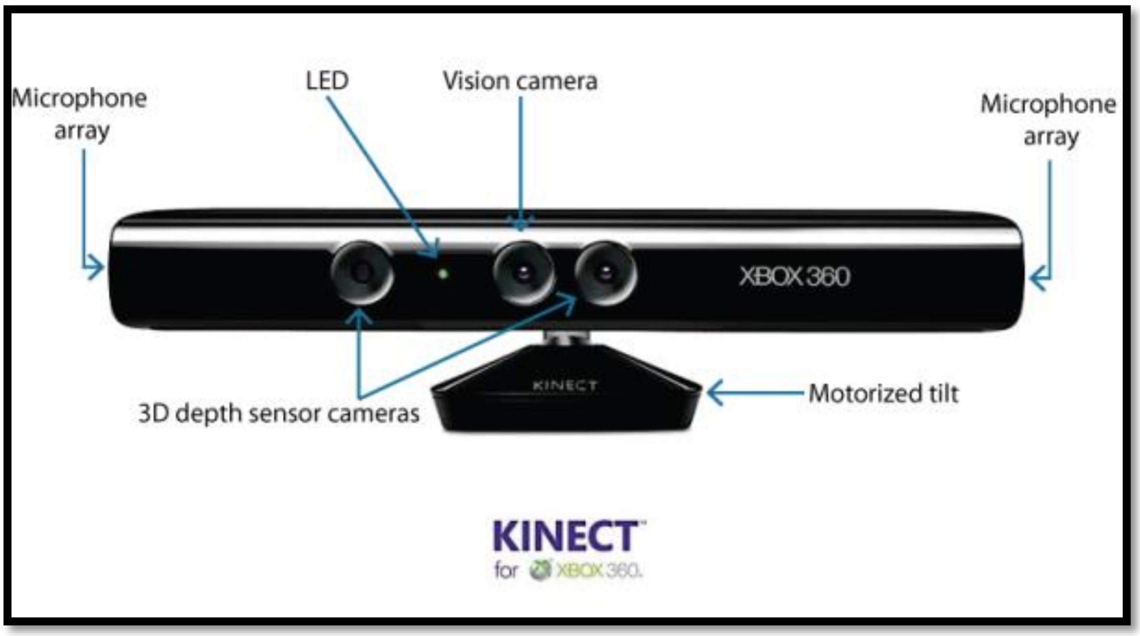
Section 6 presents the conclusion, future works and performance and quality improvement ways.

2 BACKGROUND

2.1 MICROSOFT KINECT by PRIMESENSE

Kinect is the only external hardware that used in the project. Kinect is developed by PrimeSense and its main purpose is XBOX 360 game controller for controller free game experience. As shown the figure 2.1 below Kinect poses an RGB vision camera, 3D depth sensor cameras, microphone array and motorized tilt.

Figure 2.1: Kinect components



Source: https://dev.windows.com/en-us/kinect, September 2015

Features of Kinect are listed as follows:

1) Depth Sensors are able to detect distance between 0.4 to 4 meters with 640x480 or 320x240 resolution at 30 Hz frequency. The best depth data results are obtained between 1.5 and 3 meters. Also depth data is kept in 12-bit depth.
2) RGB Sensor is a regular camera which is able to 640x480 or 320x240 resolution at 30 Hz frequency. Moreover it gives 24-bit color data.

Kinect for Windows SDK which is released by Microsoft on February 2012, is used to develop the project. Although Microsoft released a new Kinect model, however it is not used in this project.

2.2 FEATURES

There are different local features with different characteristics. In this project the camera has no limitations like navigating on a pre-defined road or something like that. That means camera can move anywhere during scanning.

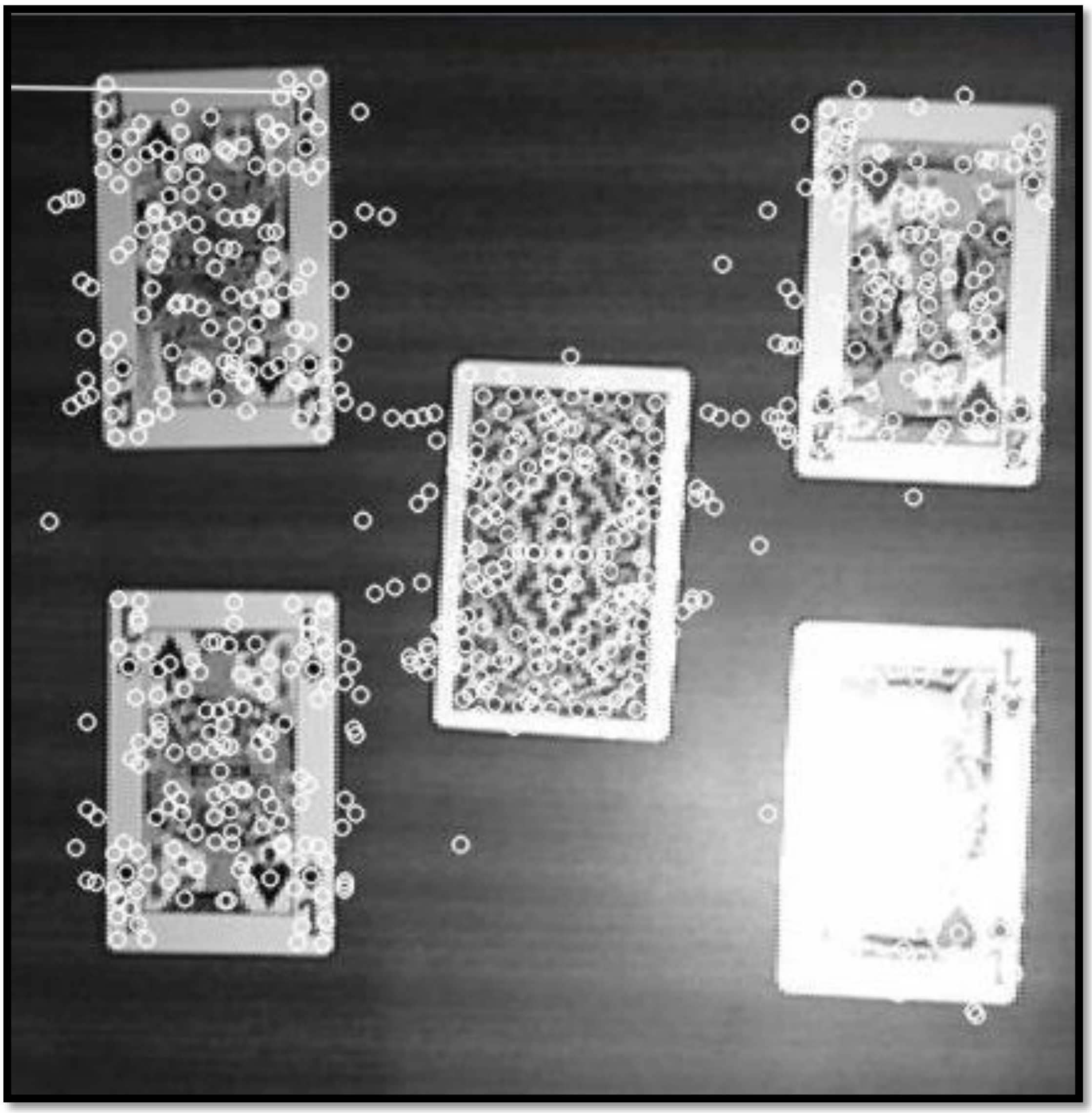
Also the local features is used to detect certain fingerprints on the taken picture will be used to detect camera position in 3D space. To achieve that, the local features must be invariant to scale and rotation. When it comes to scale and rotation invariant. There are a few features that can be applied. First one of them is SIFT feature [13]. SIFT is scale and rotation invariant. Another one is called SURF [1]. SURF and SIFT are shares the same principles, but they get differ in detail.

On this project SURF feature searched, experimented for the detecting extracting and matching descriptor points of consecutive frames.

2.2.1 SURF Feature

SURF algorithm was presented by Herbert Bay in 2006. Firstly SURF algorithm is inspired by SIFT features. More importantly SURF feature works several times faster than SIFT algorithm. Also SURF algorithm is widely used in Computer vision and robotic applications.
The main idea is finding descriptor points on 2D image (See figure 2.2)

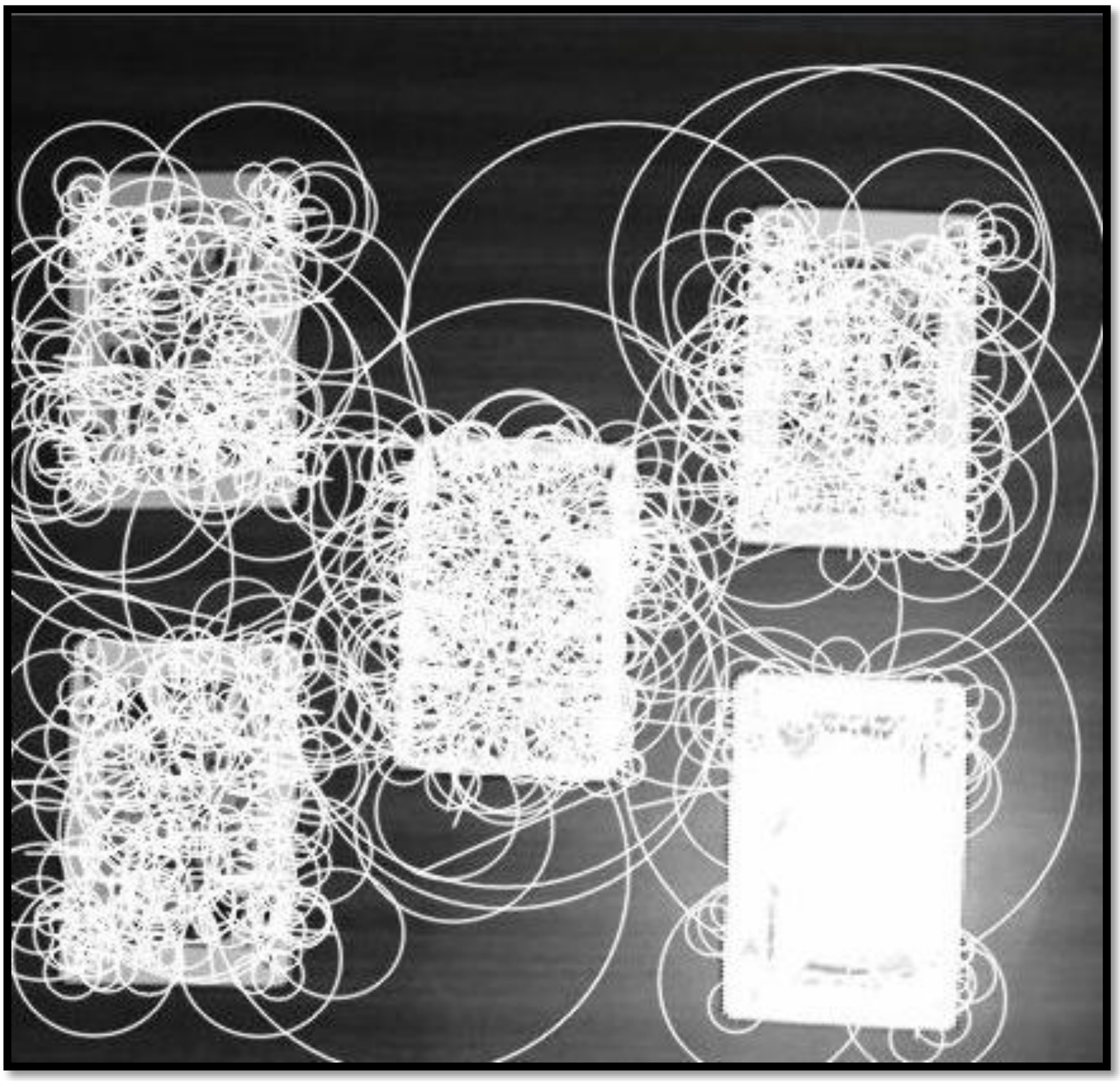
Figure 2.2: The descriptor points of the frame in white circles



2.2.1.1 SURF descriptors

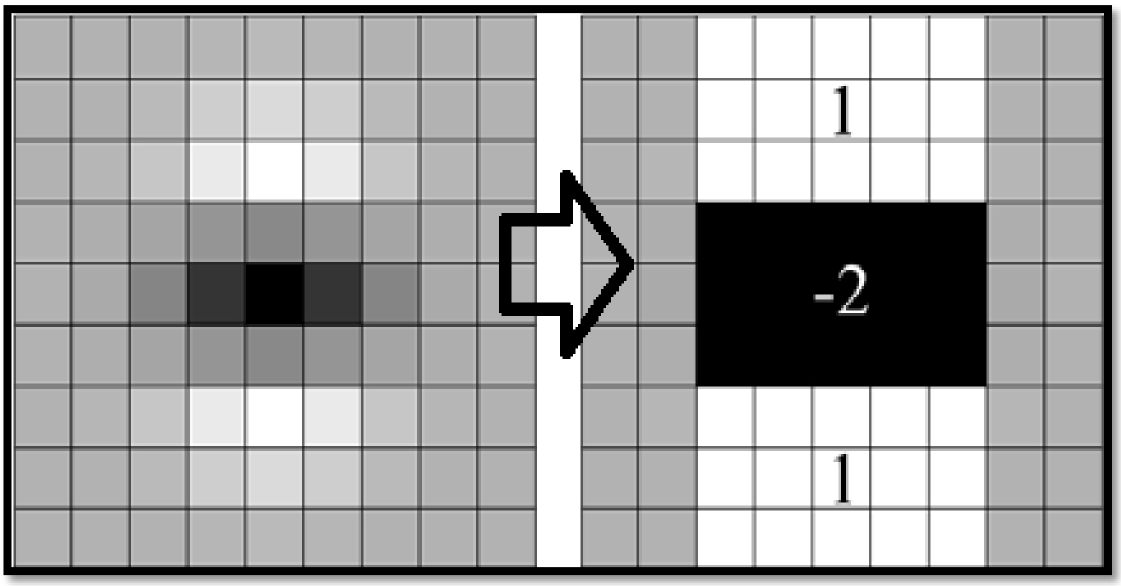
SURF features include the pixel array (like finger print for each descriptor), size, angle and coordinates of each descriptor points. Therefore rotation, scaling do not create any handicap for feature matching operations. In figure 2.3 shows the angle and size information of figure 2.2.

Figure 2.3: Descriptor points with size and angle information



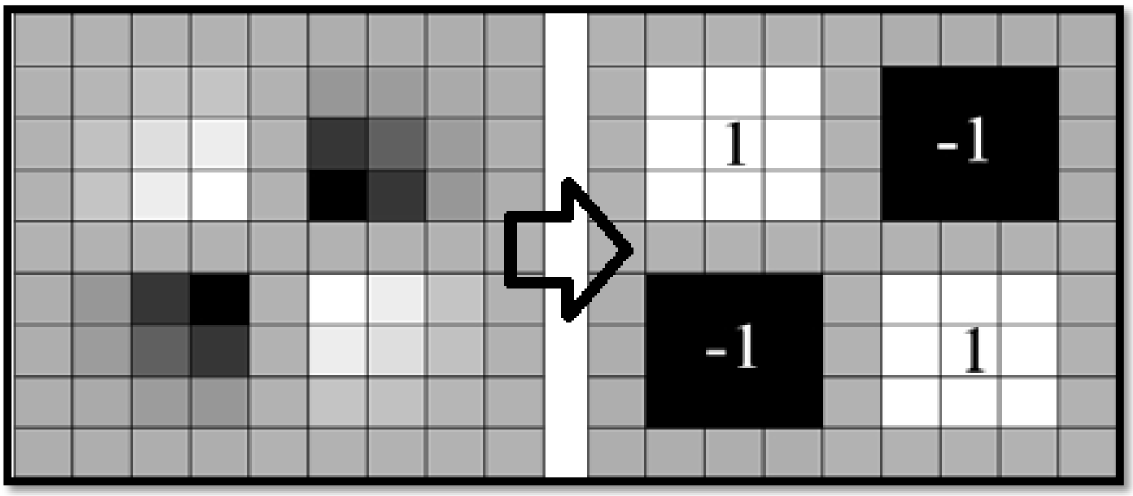
In order to determine characteristics of each descriptor points, SURF descriptors depends on approximation of Gaussian second order derivation (See figure 2.4 and 2.5), and 2D approximated 2D Haar wavelet responses (See figure 2.6, 2.7) [1].

Figure 2.4: Approximation of Gaussian second order derivation in y-direction (Grey boxes are equal to 0)



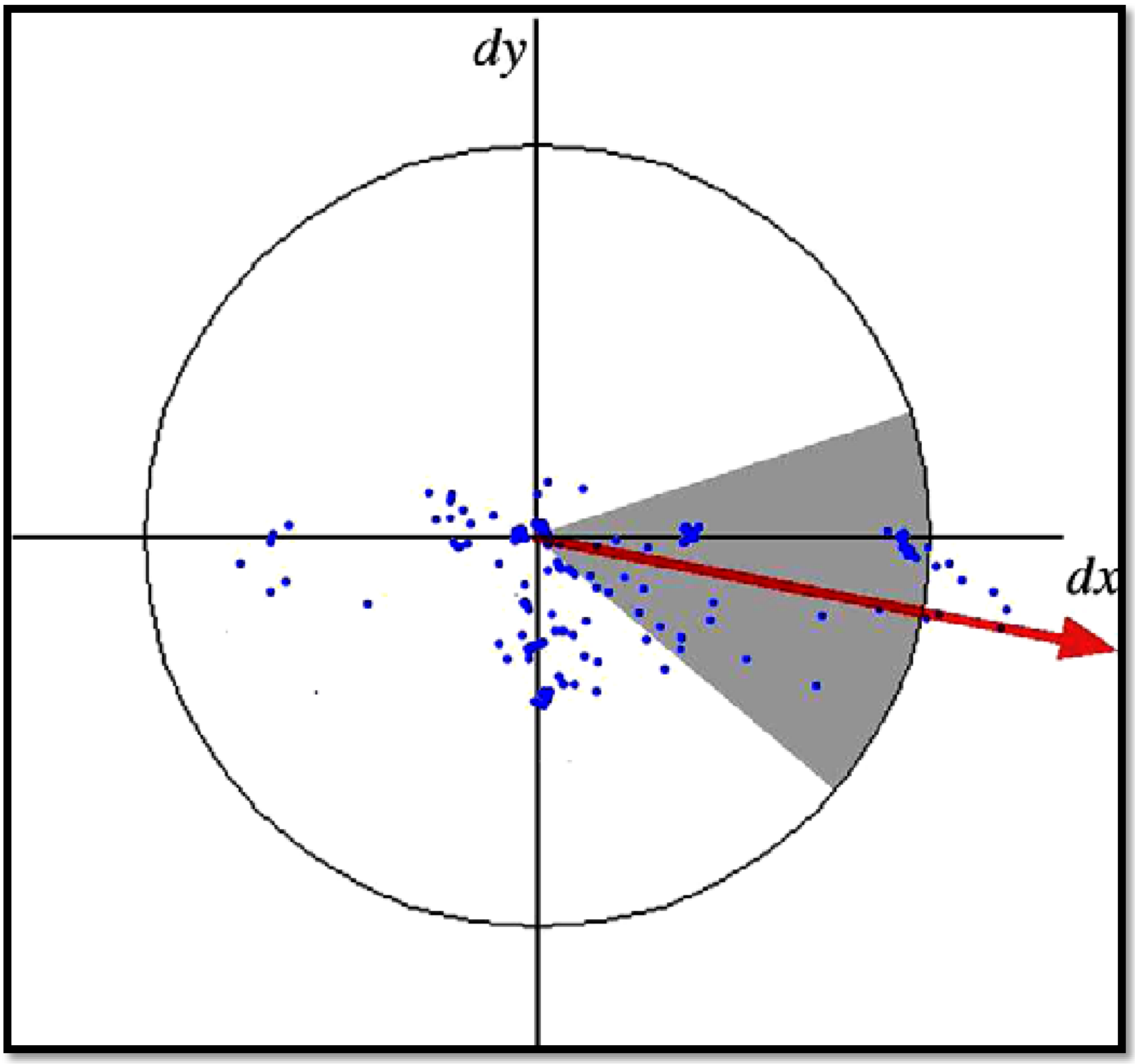
Source: http://www.vision.ee.ethz.ch/ , October 2015

Figure 2.5: Approximation of Gaussian second order derivation in xy-direction



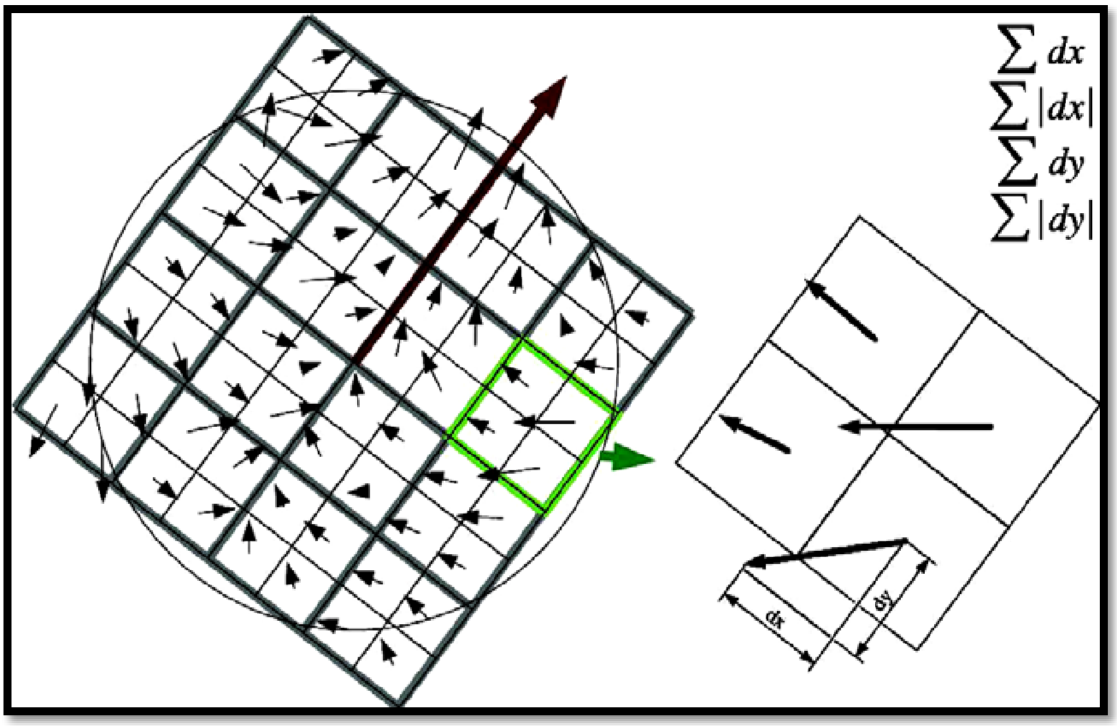
Source: http://www.vision.ee.ethz.ch/ , October 2015

Figure 2.6: the dominant orientation of Gaussian weighted Haar Wavelet responses are found by summing all results within the sliding orientation size (for that example sliding orientation size is Π/3)



Source: http://www.vision.ee.ethz.ch/ , October 2015

Figure 2.7: Overall view of local orientation vectors,. To build the descriptor, summing all of the orientation vectors shape the descriptor point characteristics.

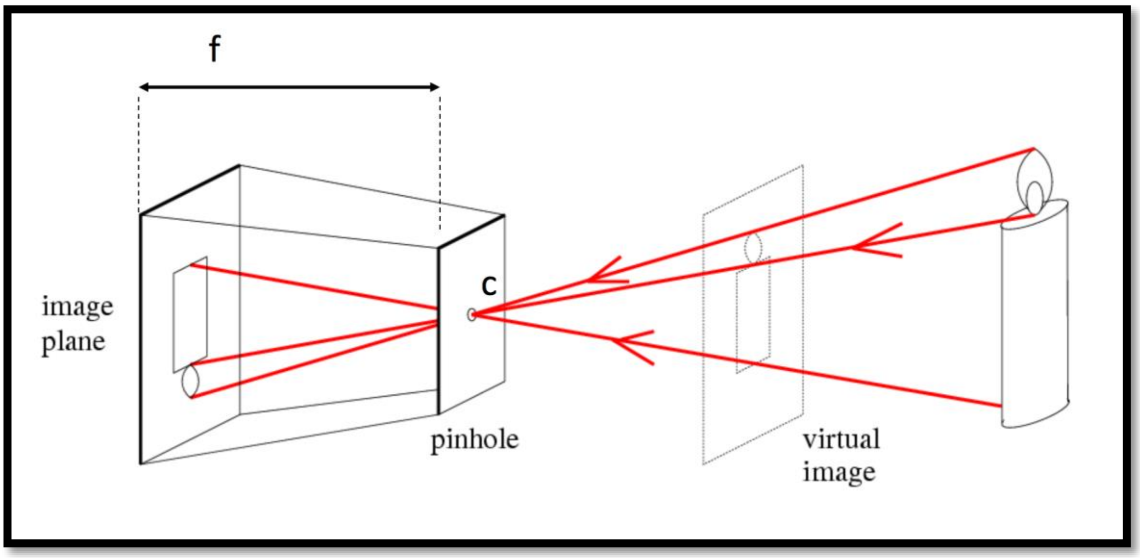


Source: http://www.vision.ee.ethz.ch/ , October 2015

2.3 CAMERA INTRINSICS

One of the requirements of 3d rigid object scanner is to generate 3d point clouds via cameras. At this point Kinect provides a huge advantage to other traditional RGB cameras, which is it provides a depth fame along with RGB frame. A Depth frame is like RGB frame, with one difference. It holds the distance between 3D points and the camera on each corresponding x and y positions, instead of red green and blue channels. To estimate point cloud dimension in real world we will apply pinhole camera model to RGB and depth sensors on Kinect. Thus we can compare real world meters to use in our calculations.

Figure 2.8: Pinhole camera model



Source: https://en.wikipedia.org/wiki/Pinhole_camera , November 2015

f is focal length and c is the centre of the camera.

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & t_u \\ 0 & f & t_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad \textbf{(2.1)}$$

X, Y and Z are the coordinates in 3D u and v are the corresponding point of 3D point on the RGB frame. f is the focal length and t is the centre of the camera.

Our pin hole camera application consist of three steps

1. Find the surf features position 3D space.
2. Then calibrated these points by using depth sensor calibration matrix. In other words align depth and RGB sensor on Kinect.
3. Then reverse equation 2.1 and align the corresponding point of the RGB frame Surf feature location and depth frame depth value. Thus we can use depth sensor information in our further computations.

2.3.1 Kinect Camera Calibration

As displayed Figure 2.1 Kinect Depth and RGB cameras are separated. Therefore RGB cameras coordinates do not align with the depth cameras. To get this problem over, depth camera calibration parameters which are found out by Nicolas Burrus [3], are used (see figure 2.9). Additionally parameters provide accurate results.

Figure 2.9: Rotation, Translation parameters of Kinect depth camera calibration which are used find RGB frame descriptor point coordinates on the 3D point cloud coordinates.

$$\begin{aligned}
&\text{fx\_rgb} = 5.2921508098293293e+02; \quad \text{fx\_depth} = 5.9421434211923247e+02\\
&\text{fy\_rgb} = 5.2556393630057437e+02; \quad \text{fy\_depth} = 5.9104053696870778e+02\\
&\text{cx\_rgb} = 3.2894272028759258e+02; \quad \text{cx\_depth} = 3.3930780975300314e+02\\
&\text{cy\_rgb} = 2.6748068171871557e+02; \quad \text{cy\_depth} = 2.4273913761751615e+02
\end{aligned}$$

$$R = \begin{bmatrix} 9.9984628826577793e-01 & 1.2635359098409581e-03 & -1.7487233004436643e-02 \\ -1.4779096108364480e-03 & 9.9992385683542895e-01 & -1.2251380107679535e-02 \\ 1.7470421412464927e-02 & 1.2275341476520762e-02 & 9.9977202419716948e-01 \end{bmatrix}$$

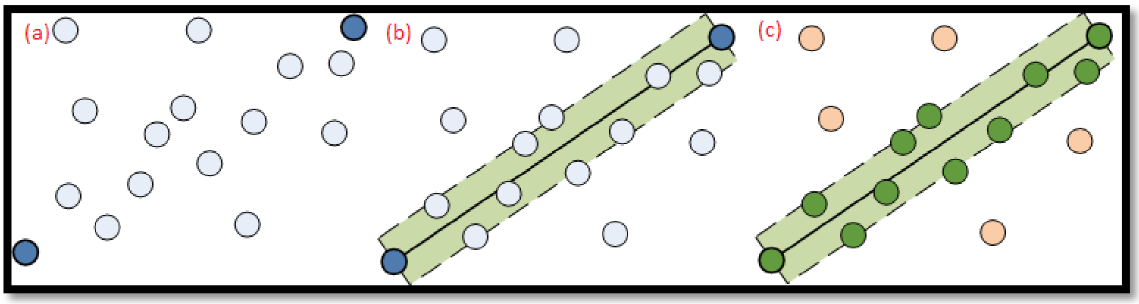
$$T = \begin{bmatrix} 1.9985242312092553e-02 & -7.4423738761617583e-04 & -1.0916736334336222e-02 \end{bmatrix}$$

2.4 RANSAC

RANSAC is abbreviation for RANdom SAmple Consensus [2]. The main idea of RANSAC algorithm is elimination the noisy data from a set of data. In other words determines the outliers and inliers by given or found threshold value that depends on the case.

A single step of RANSAC is shown in figures 2.10. It randomly chooses k item, then determines the inliers and outliers by threshold value.

Figure 2.10: Visual presentation of a single RANSAC iteration



Source: Wikipedia

RANSAC is actually a concept of the separating a set of data by outliers and inliers. The generated RANSAC algorithm that is used In this project is presented as below:

Algorithm 1 RANSACS

```
Input:

      MPLeft, MPRight – matched descriptor point arrays[n]

      Iteration – number of iteration

      Threshold – Threshold value that determines inliers and outlier

Variables:

      i – loop index

      p1,p2,p3 –random three matched 3D Points

      counter – candidate case(number of max supporter points)

      transformationM – Candidate transformation matrix

      A – set of three candidate Points which are belong to MPLeft

      B – set of three candidate Points which are belong to MPRight

OutPut:

      CounterBest – Best case/Max number of supporter 3D points of candidate
transformation matrix

      TransformationMBest – Best candidate transformation matrix


counterBest := -inf

for i=0 to Iteration

      p1 := randomInteger() % MPLeft.Length()

      p2 := randomInteger() % MPLeft.Length()

      p3 := randomInteger() % MPLeft.Length()

      A:= {MPLeft[p1], MPLeft[p2], MPLeft[p3]}

      B:= {MPRight[p1], MPRight[p2], MPRight[p3]}

      transformationM := Get-TransformationMatrix(A,B);


      counter := SUM(Euclidean-Distance(MPLeft , transformationM * MPRight)
< Threshold)

      if(counter > counterBest)
```

```
            TransformationMBest := transformationM
            counterBest := counter
        end if
end for
return TransformationMBest, CounterBest
end
```

Look section 2.6 for Get-TransformationMatrix

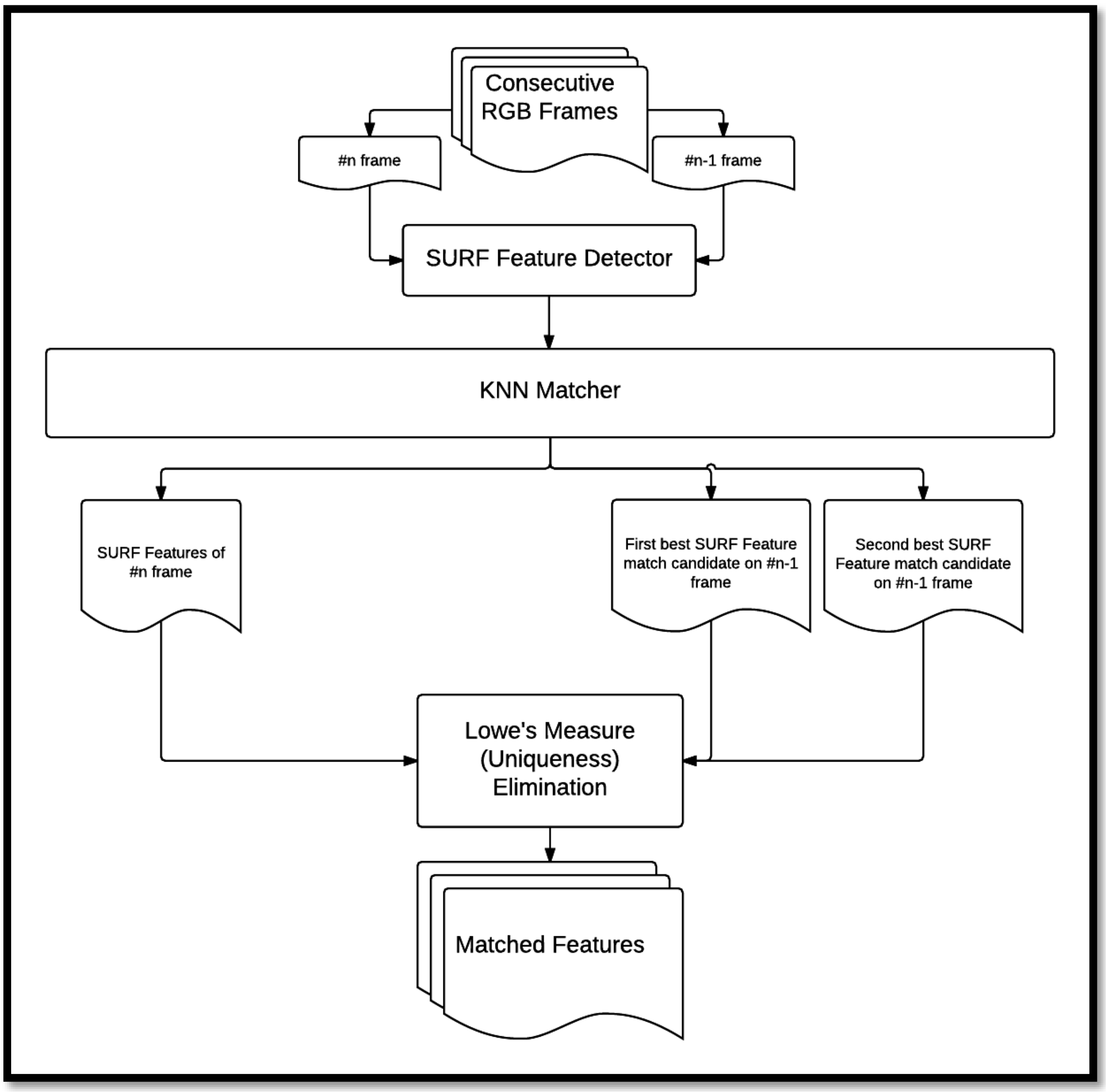
2.5 LOWE's MEASURE

Like the Depthscale measure (see section 5), Lowe's measure [5] is an elimination method to determine if the match is correct match.

Lowe's measure takes place after knn (where k is equal to 2). Knn provides takes a set of train feature (in our case, they are SURF Features) and set of test feature. Then it matches every train feature with two closest test feature, one is closest the other is the second closest. After point Lowe's measure ides is to determine if the match is a health match. Because these matches are local feature there might be a second test feature similar to the first test feature. To be sure it check the first and second features similarities by a threshold. (In this project mostly threshold is used as 0.8.)

Since Lowe's measure is an elimination process by looking the candidates' similarities. The work pipeline until the Lowe's Measure is demonstrated in figure 2.11.

Figure 2.11: KNN match and incorrect match detection by Lowe's measure


Consecutive RGB Frames
#n frame
#n-1 frame
SURF Feature Detector
KNN Matcher
SURF Features of #n frame
First best SURF Feature match candidate on #n-1 frame
Second best SURF Feature match candidate on #n-1 frame
Lowe's Measure (Uniqueness) Elimination
Matched Features

As displayed in figure 2.11 the main idea for this method is after the knn match that returns two candidate feature for each observed feature, checks the candidate features if these two candidate features are too similar, then Lowe's measure tags that match as a incorrect match, in other words outlier. The matched features will be used afterward to generate 3D transformation matrix between consecutive point clouds.

2.6 RIGID TRANSFORMATION

In the rigid transformation progress we will use an algorithm, which was developed by Arun KS in 1987 [4], to determine the relation between two different 3D coordinate system. To accomplish that matched descriptor points and their 3D space presentations are needed. The relation between two set of matched descriptor points is provided by Euclidean transformation matrix which is consist of rotation **R** matrix and translation **t** vector.

$$d_i = \mathbf{R}\, m_i + \mathbf{T} + \mathbf{V}_i \tag{2.2}$$

In the equation 2.2, $\{d_i\}$ $\{m_i\}$ presents the matched descriptor points. Also **R** for rotation matrix, **T** for translation vector and $\mathbf{V_i}$ for noisy vector.

The progress of finding the transformation matrix is listed as follows:

$$\overline{d} = \frac{1}{N}\sum_{i=0}^{N} d_i \tag{2.3}$$

N stands for count of the cluster.

$$\overline{m} = \frac{1}{N}\sum_{i=0}^{N} m_i \tag{2.4}$$

$$d_{c_i} = d_i - \overline{d} \tag{2.5}$$

$$m_{c_i} = m_i - \overline{m} \tag{2.6}$$

$$\mathrm{H} = \sum_{i=0}^{N} m_{c_i} d_{c_i}{}^{\mathrm{T}} \tag{2.7}$$

$$\mathrm{H} = U \Lambda V^T \tag{2.8}$$

Singular Value Decomposition equation 2.8.

$$R = VU^{\mathrm{T}} \tag{2.9}$$

$$T = \overline{d} - R\overline{m} \tag{2.10}$$

1) Following equations 2.3, 2.4, 2.5, 2.6, 2.7, and 2.8 is applied respectively.
2) After finding the U and V^T, Rotation matrix can be found by equation 2.9.
3) Lastly by rotation the centroid point of descriptor points by the following equation translation vector can be gained by equation 2.10.

$$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.11}$$

To summarize, as shown in the equation 2.11 placing the Rotation 3x3 matrix and translation matrix 3x1 in a matrix 4x4, the final transformation matrix that provide rigid transformation, get its final form.

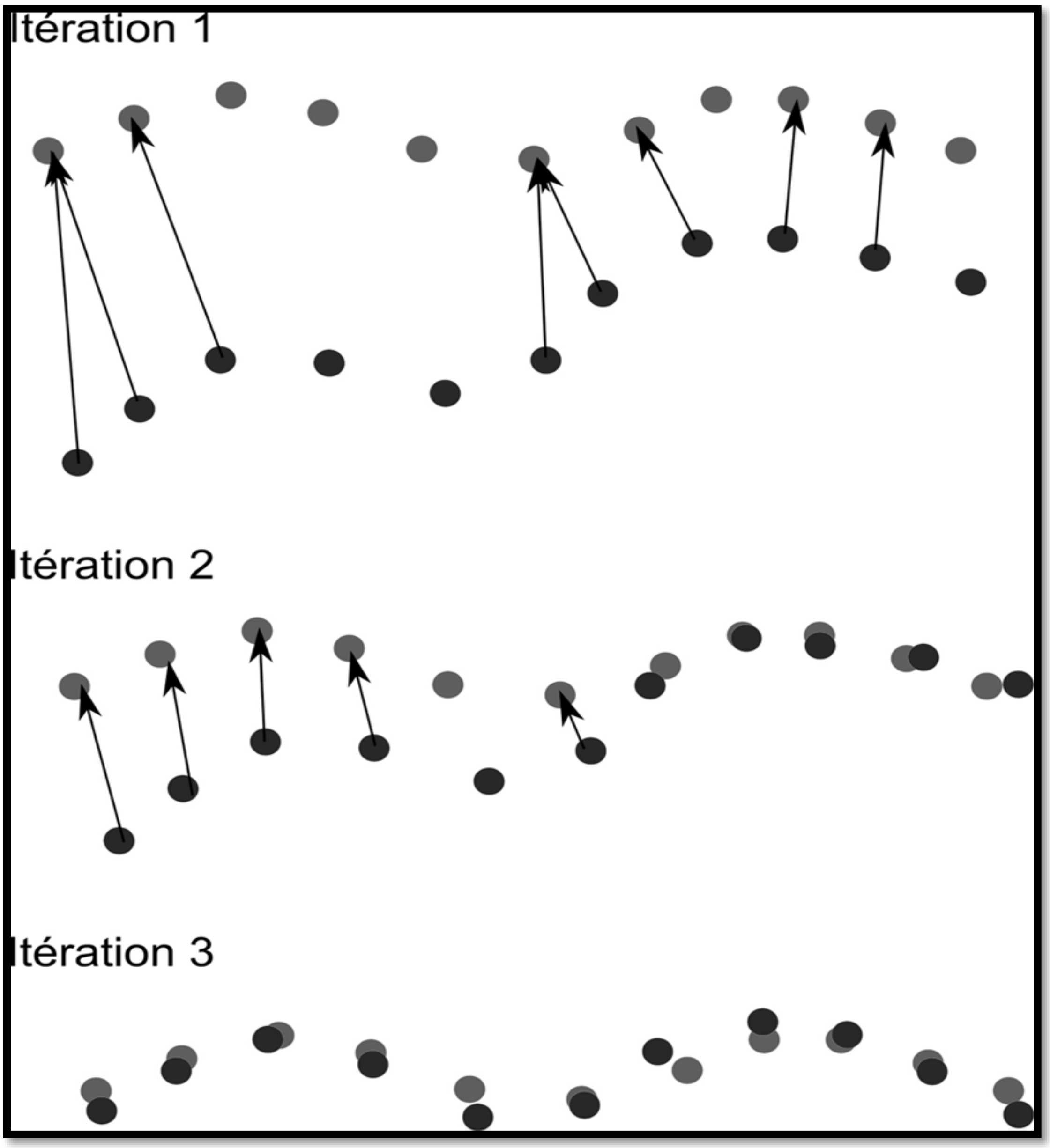
2.7 ITERATIVE CLOSEST POINT

Iterative Closest Point [9] is an algorithm that is used to align consecutive point clouds. There are various approaches to accomplish Iterative Closest Point. Main purpose of the algorithm is, it tries to reduce the Euclidean distance between two closest point cloud sets. It does that in an iterative way.

The algorithm takes two set of point clouds (I call them train and test) and iteration count as inputs and returns a transformation matrix between them. The algorithm is accomplished by in the following order:

1. For each point in the train point cloud, find the closest point in the test point cloud.
2. Estimate the combination of rotation and translation using a mean squared error cost function that will best align each train point to its match found in the previous step.
3. Transform the train points using the obtained transformation.
4. Iterate (re-associate the points, and repeat previous steps until iteration count ends).

Figure 2.12: An illustration of ICP algorithm iterations.



Source: http://code.opencv.org/projects/opencv/wiki/2012 , October 2015

As given in illustrated in figure 2.12, the ICP works well. However it has some disadvantages. First of all it can be very expensive to iterate entire point cloud estimate transformation. Also more importantly the way ICP algorithm causes misaligned results if the initial alignments of the point clouds are not near enough. This means two consecutive point cloud should be close each other. Otherwise it may end up with misaligned point clouds. So it is crucial to be sure before applying ICP.

3 FEATURE MATCHING

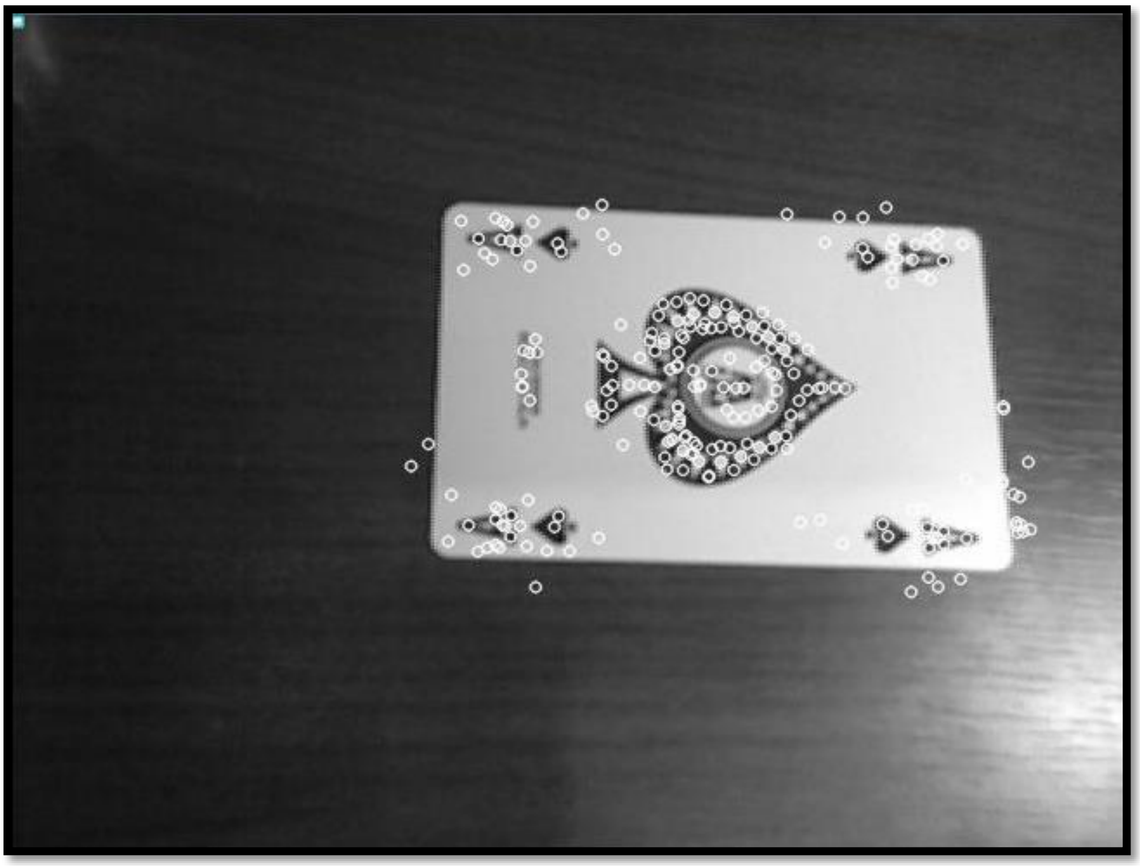
In this part of the project the purpose is finding and matching descriptor points of two consecutive frames. Also eliminating the noisy matches are included. In order to able to achieve that on that project, OpenCV C++ libraries are used [6]. OpenCV C++ libraries provides SURF feature for CUDA supported GPUs and CPUs.

3.1 FEATURE EXTRACTION

The main goal in this step extracting the descriptor points of both consecutive frames. To achieve that SURF algorithm used. SIFT algorithm could be used. However SURF algorithm works couple times faster than SIFT as mentioned.

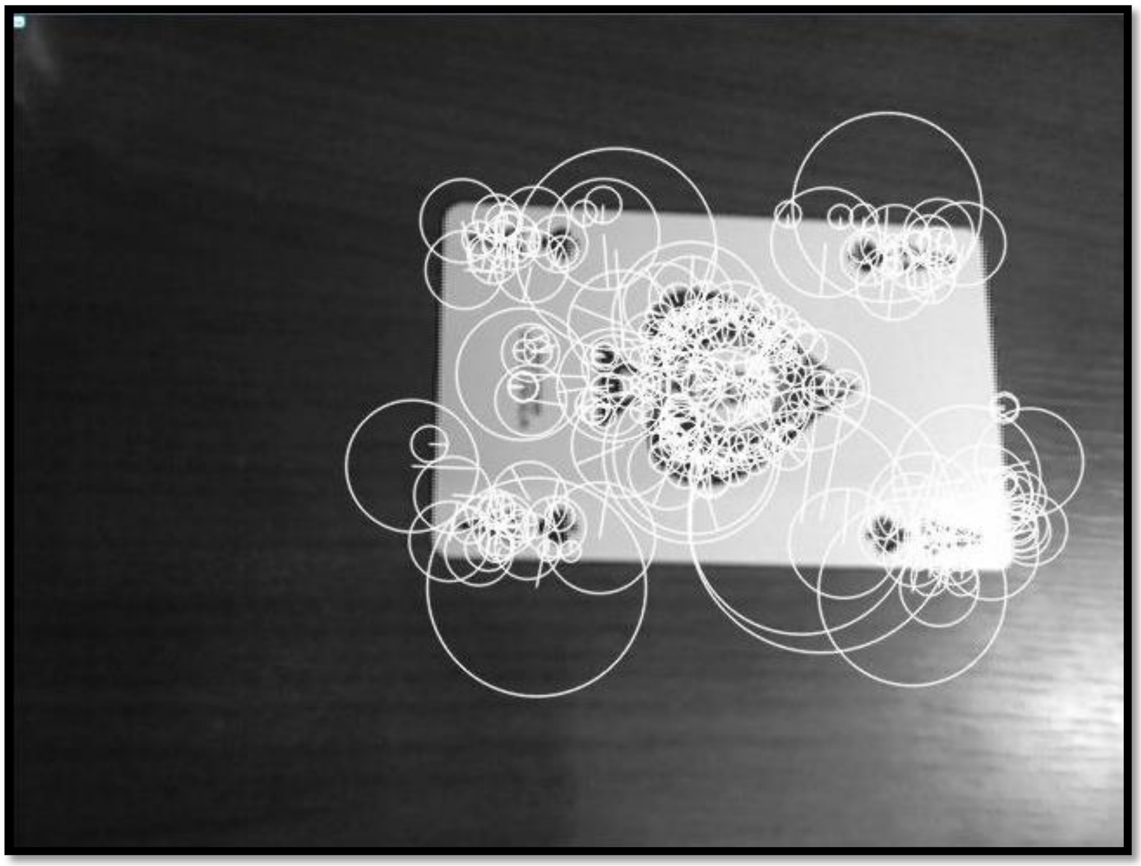
SIFT and SURF algorithm provided by OpenCV.

Figure 3.1: Feature Extraction Result of an image



In the figure 3.1 displays locations of the feature these are extracted from the image. (See Figure 3.2 for more detailed descriptors with size and angle)

Figure 3.2: Descriptor points with angle and size information



3.2 FEATURE MATCHING

Relation of two consecutive images frame is reached by KNN match. KNN match returns two nearest neighbour descriptor match for single descriptor point. In figure 3.3 most of descriptor points have more than one match. In order to eliminate this error, algorithm performs an elimination criterion. The criterion contains the similarity ratio by 0.8 of two matches for a single descriptor points, also Depthscale elimination/measure is used eliminate ineligible matches. If the descriptor matches are not eligible then it does not count the descriptors as a match. Otherwise they are counted as matched pairs. In figure 3.4 results of the elimination is displayed.

Moreover, Depthscale Measure is used to eliminate the incorrect matches. Thus by using Lowe’s measure and Depthscale inlier ratio is increased, also that decreases the performance needs for the RANSAC and the rest of the application methods.

Figure 3.3: Matched Descriptor Points without Lowe’s measure

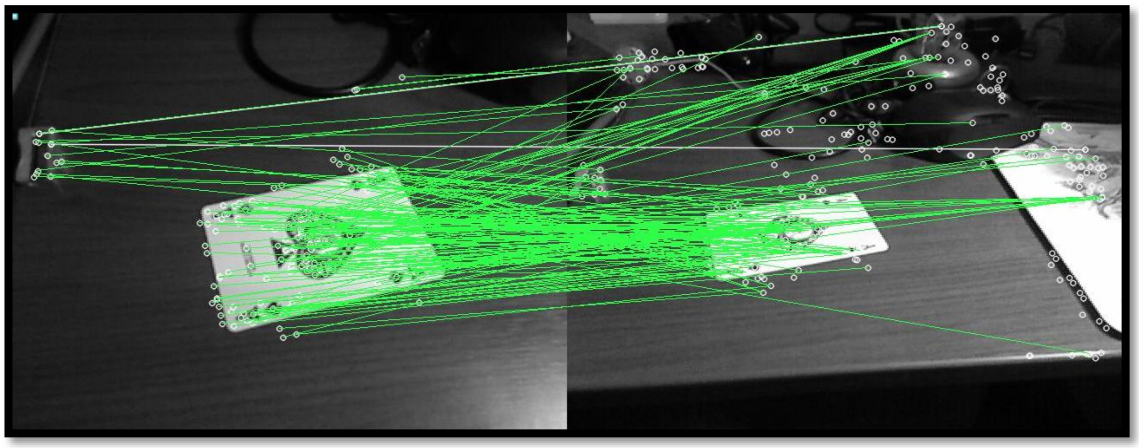
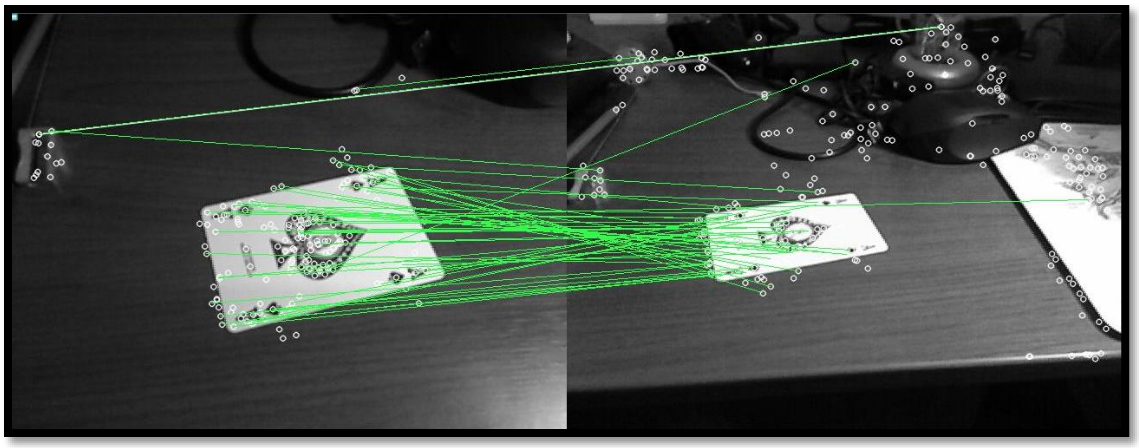


Figure 3.4: After the KNN match Lowe’s measure and Depthscale elimination

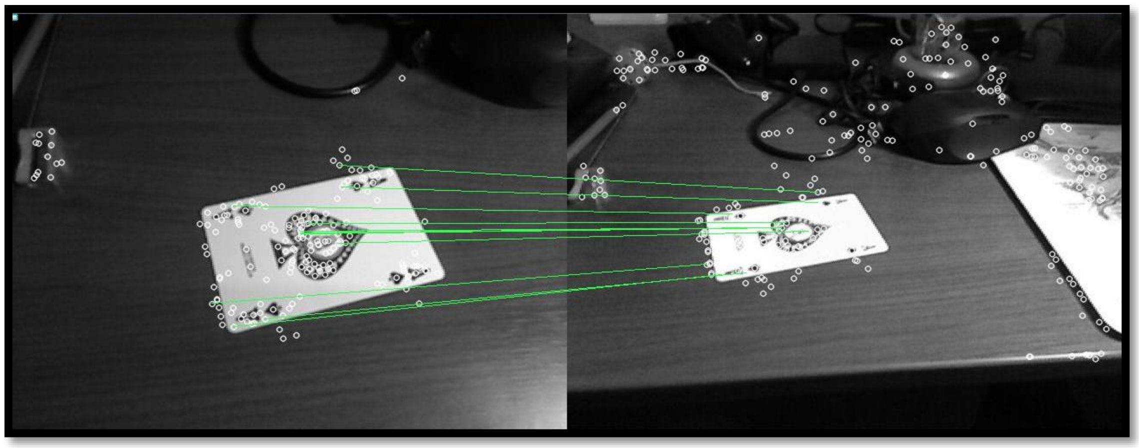


3.3 RANSAC

As mentioned in section 2 RANSAC is widely used method in order to detect inlier and outliers. Only after that method the most accurate matches can be found. After the RANSAC these matches are used to find out the transformation matrix between consecutive point clouds.

Due to the fact that after the Lowe's measure there is a high probability results contain the incorrect matches as shown in figure 3.4. In order to eliminate these outlier matches RANSAC algorithm is applied. The result of the matches after the RANSAC process is demonstrated in figure 3.5.

Figure 3.5: The match result after the RANSAC algorithm applied

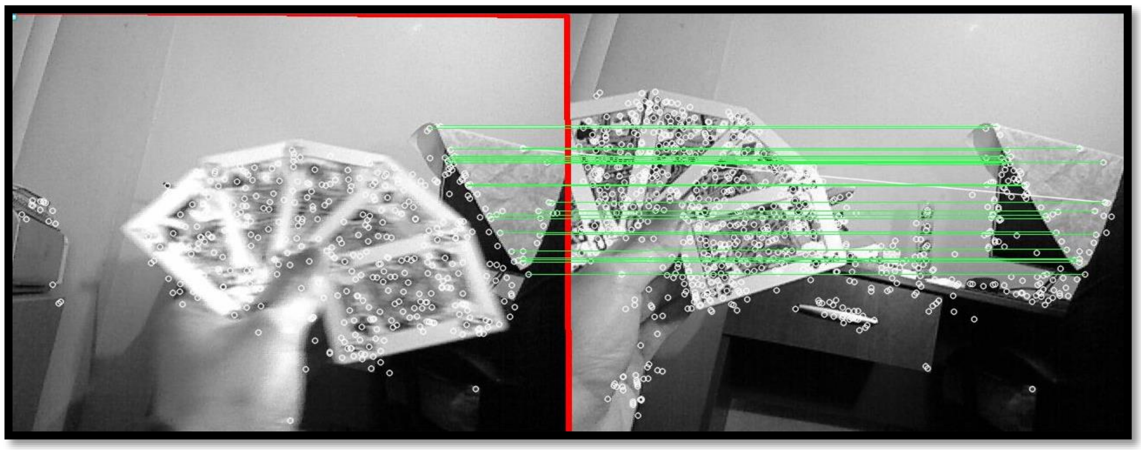


In this project RANSAC runs for 3D locations of features. Therefore some inliers in 2D space might be counted as outlier in RANSAC. Because Kinect Depth sensor is unreliable. Therefore some inliers in 2D space can be outliers in 3D space due to incorrect depth information. Check section 4.2 for more about RANSAC.

3.4 ANALYSIS

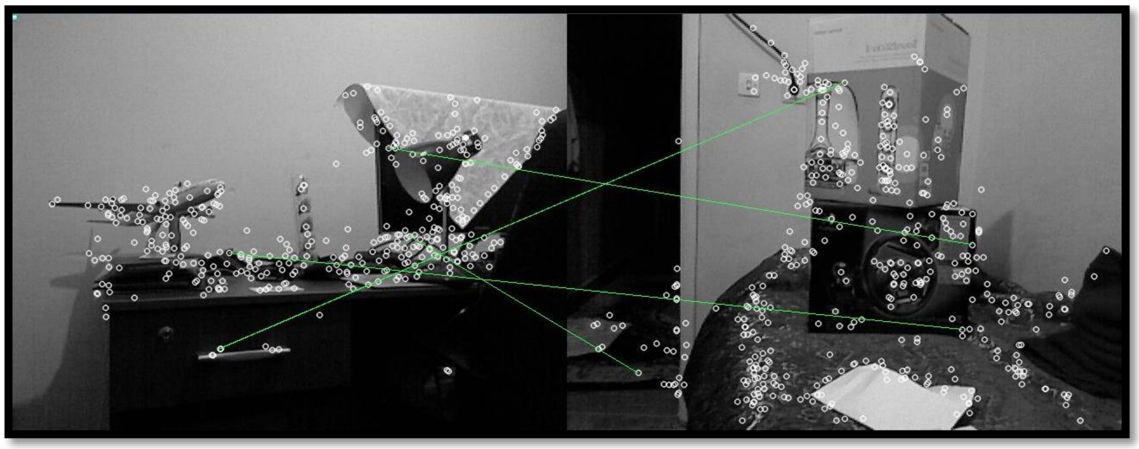
Feature matching progress runs as expected if the necessary working environment is provided for the application. Starting with the environmental effects. Firstly environment should be bright. If the environment is shady then grey scale images can be consisting of black features these are not different than each other. Another thing is objects which are used to scan should be rigid (see figure 3.6), in other words it should not be moved during the test. An additional rule for camera, it should not vibrated or shook during the experiment.

Figure 3.6: Shook and/or moved object observation



Lastly the consecutive frames should be common frame fragments. If totally two different frame is used then the matched result can be return null, or even incorrect matches as displayed in figure 3.7.

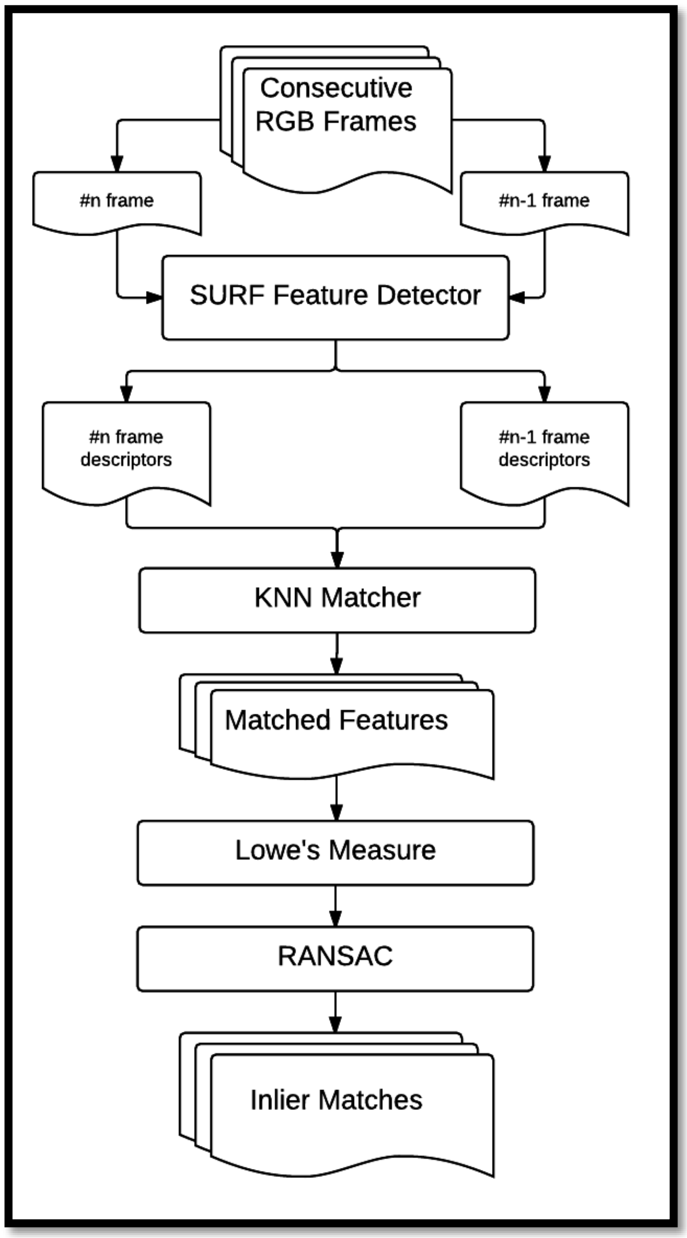
Figure 3.7: Incorrect match of unrelated frames



3.5 SUMMARY

That section starts from inputs (two consecutive frames) to outputs (matched descriptor points in 2D coordinates). As stated in figure 3.8 by using respectively SURF Feature, KNN match and finally RANSAC algorithms, matched descriptor points are found. The whole process pipeline is shown in figure 3.8.

Figure 3.8: Process Pipeline of Feature Matching Section



4 BUILDING THE MAP

This section consists of 3D algorithms, approaches applied. Main goal in this section is building a map by using these matched descriptor points.

So after Kinect depth camera calibration each depth pixel pushed to the real 3D space (see Section 2.4). Thus matched descriptor points in 3D can be get by matched descriptor points in 2D that is acquired in previous section.

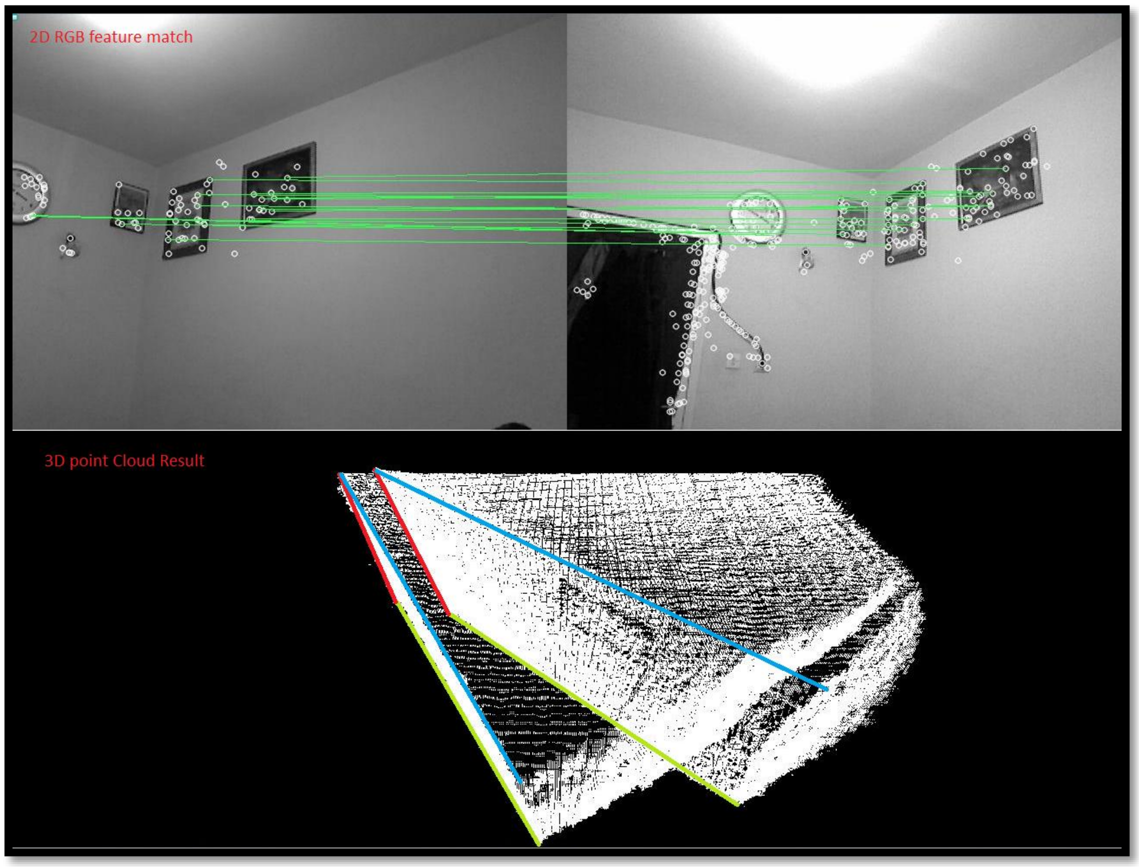
Respectively the following steps are applied:

1) 3D Rigid transformation/registration
2) RANSAC outlier elimination
3) As final touch, Iterative Closest Point (ICP)

4.1 3D RIGID TRANSFORMATION

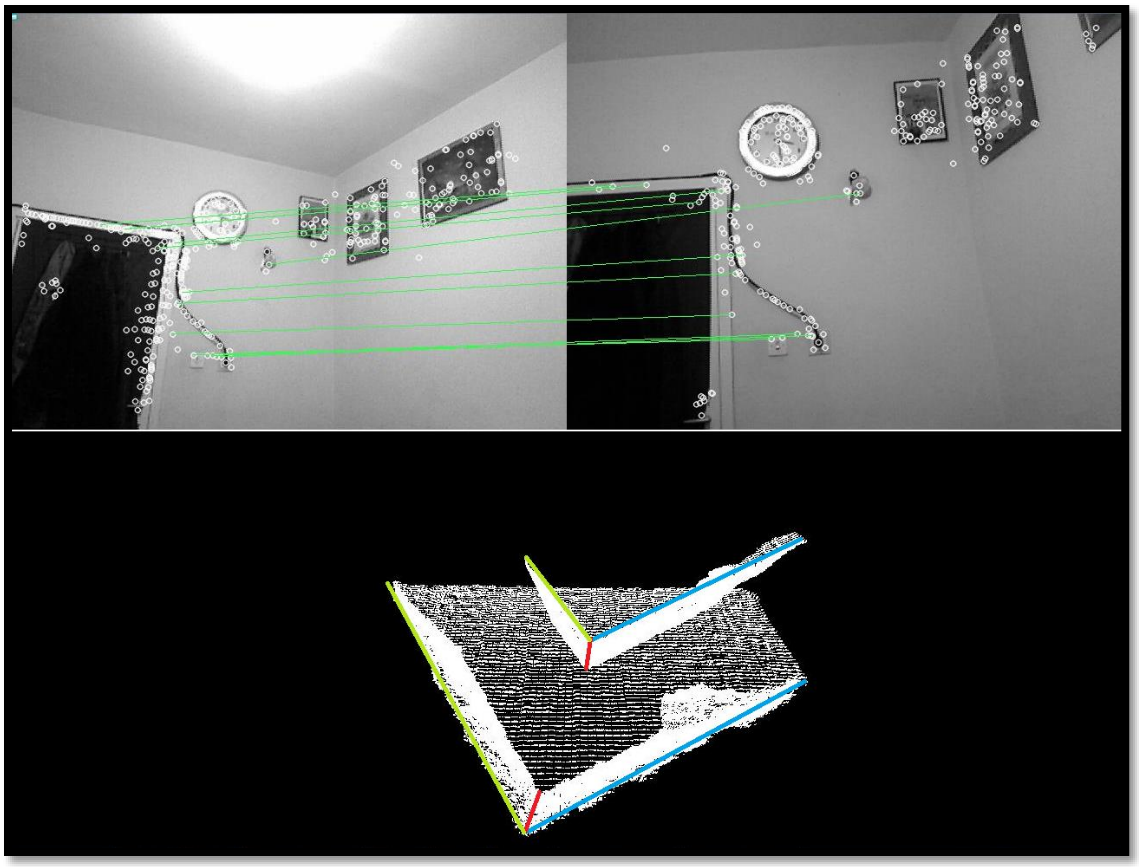
Starting with the rotation matrix, after founding the 3D matched descriptor points, rigid transformation algorithm [4] is applied (see section 2.5) to find a Rotation matrix and translation vector between two consecutive point clouds. Firstly figure 4.1 shows no transformation by rotation matrix.

Figure 4.1: Aligned two consecutive results by the rotation matrix



After rotation matrix applied result is displayed result is shown in figure 4.2

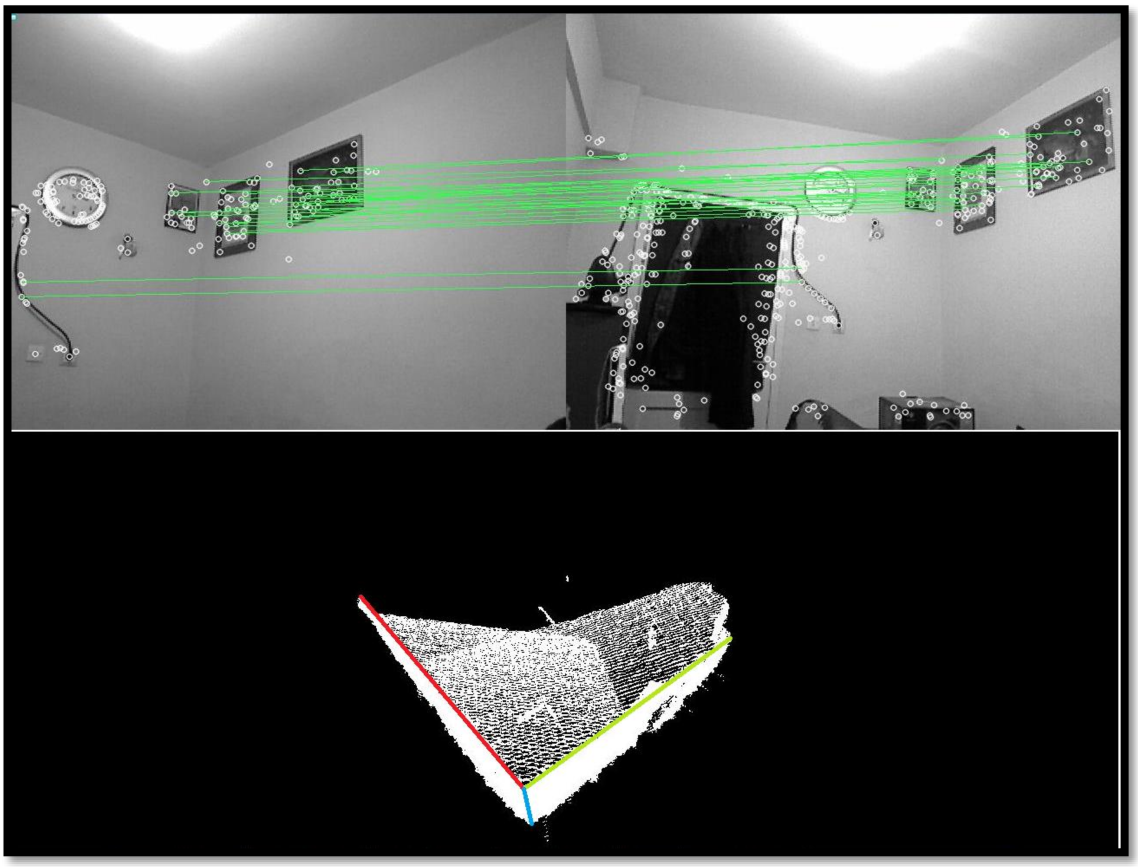
Figure 4.2: transformed by Rotation matrix result



As shown in the figure 4.2 the result, the point clouds are parallel, that means the algorithm give us good accuracy by only 8 matched descriptor points.

Final step is translation, after translation is added to transformation matrix consecutive point clouds match to each other. See figure 4.3

Figure 4.3: Final result of matched descriptor points



Some more results shown in figure 4.4, 4.5 and 4.6

Figure 4.4: Another experiment results 1 (red and green lines are the border of the point cloud frames, and pink, blue and yellow lines belong to 3 different figure)

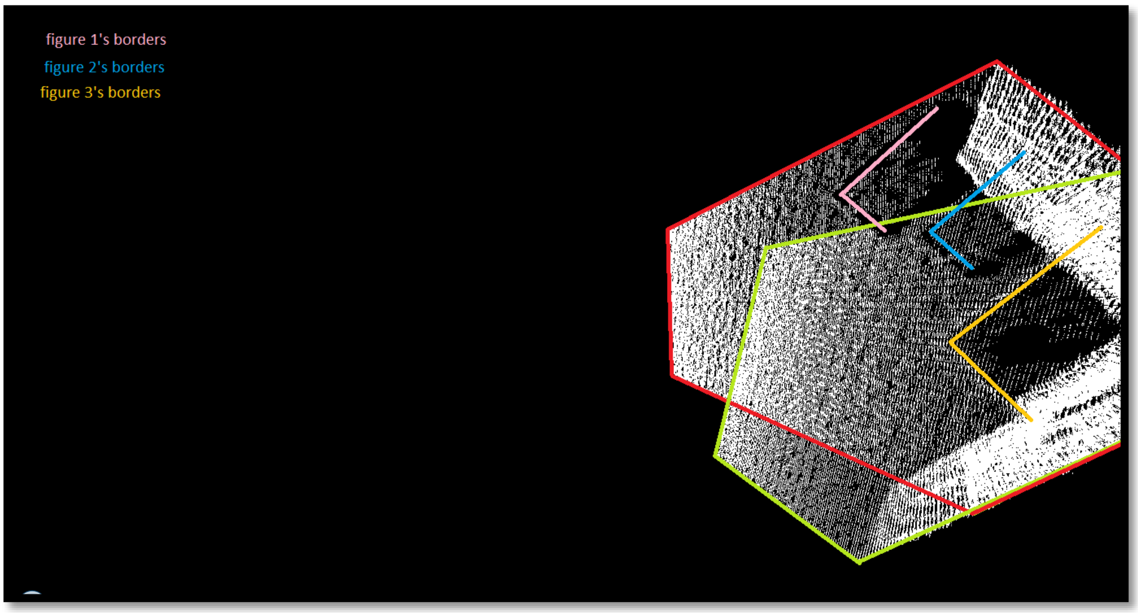


Figure 4.5: Another experiment results 2 Rotated view (60o rotated on Y axis) of the figure 4.4

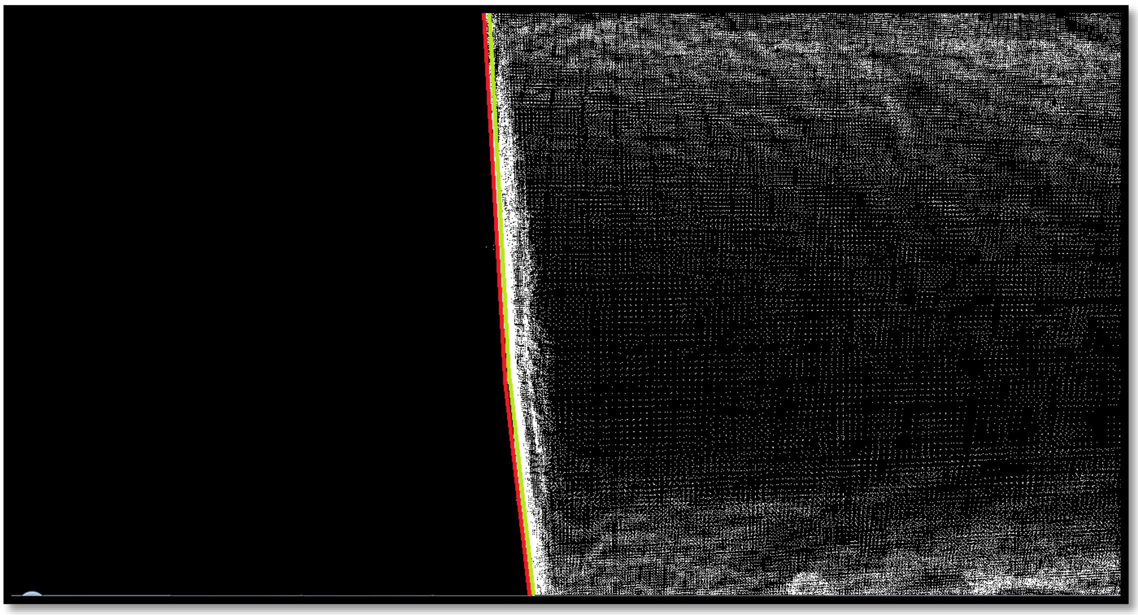
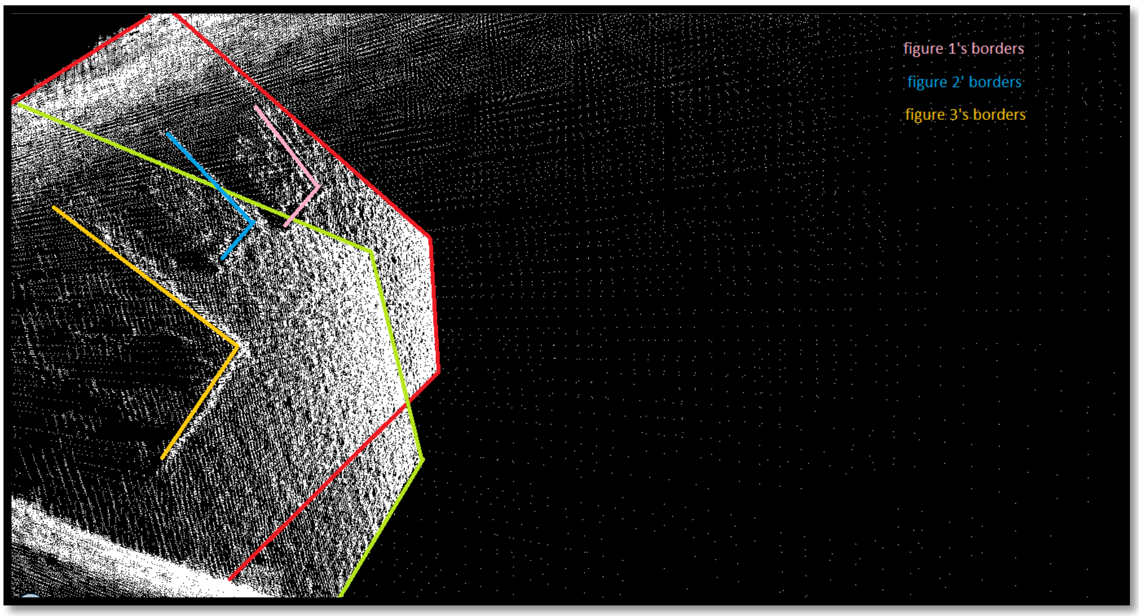


Figure 4.6: Another experiment results 3 Rotated view (60o rotated on Y axis) of the figure 4.5



4.2 RANSAC

Kinect as described in section 2 gives depth frame along with RGB frame. With this new output it opened new ways to find some applications, like sensing the 3D environment or improving existing ones. However Kinect is not perfect, the depth frame Kinect provides actually a super sampled frame of a lower resolution frame. So it contains error before arriving to client. Plus there are more errors which come from 2D feature match process.

In section 3.3 after RANSAC elimination the inlier SURF features are demonstrated. We will use RANSAC to eliminate Kinect depth frame and SURF Feature match errors. To achieve that RANSAC is applied by using 3D matched points and 3D pint clouds registration algorithm which described in section 4.

RANSAC applied by following order

1. Get training and test matched features, iteration count and threshold distance in 3D space as inputs
2. Generate transformation matrix by choosing 3 random matched feature. As explained in section 4.
3. Apply transformation to test pairs of matched.
4. Calculate the distance between matches, and count how many of them within the threshold.
5. Hold the max value and the random indices
6. Repeat step 2, 3, 4 and 5 until iteration count exceeds.
7. Tag max value and the random 3 matches and the supporter matches
8. Return tagged matches as inliers.

4.3 EXPERIMENTS

After all code built up some sort of different scenes are used to find the result on different characteristic environments like sitting person on a chair, living room, and some box stacks, as shown in figure 4.7, 4.8 and 4.9.

Figure 4.7: Human body observing (5 different images are merged)

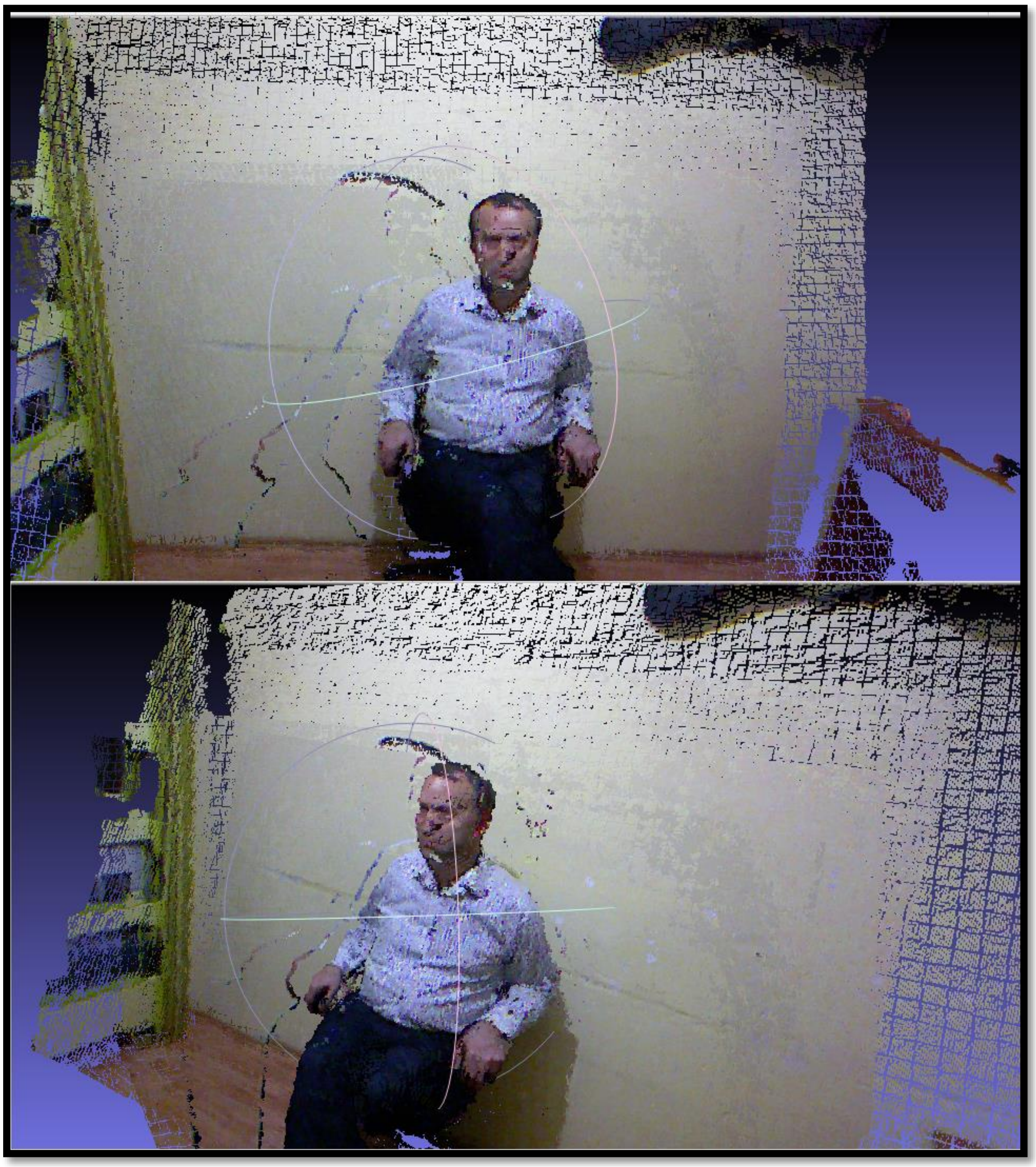


Figure 4.8: different human body observing (5 images)

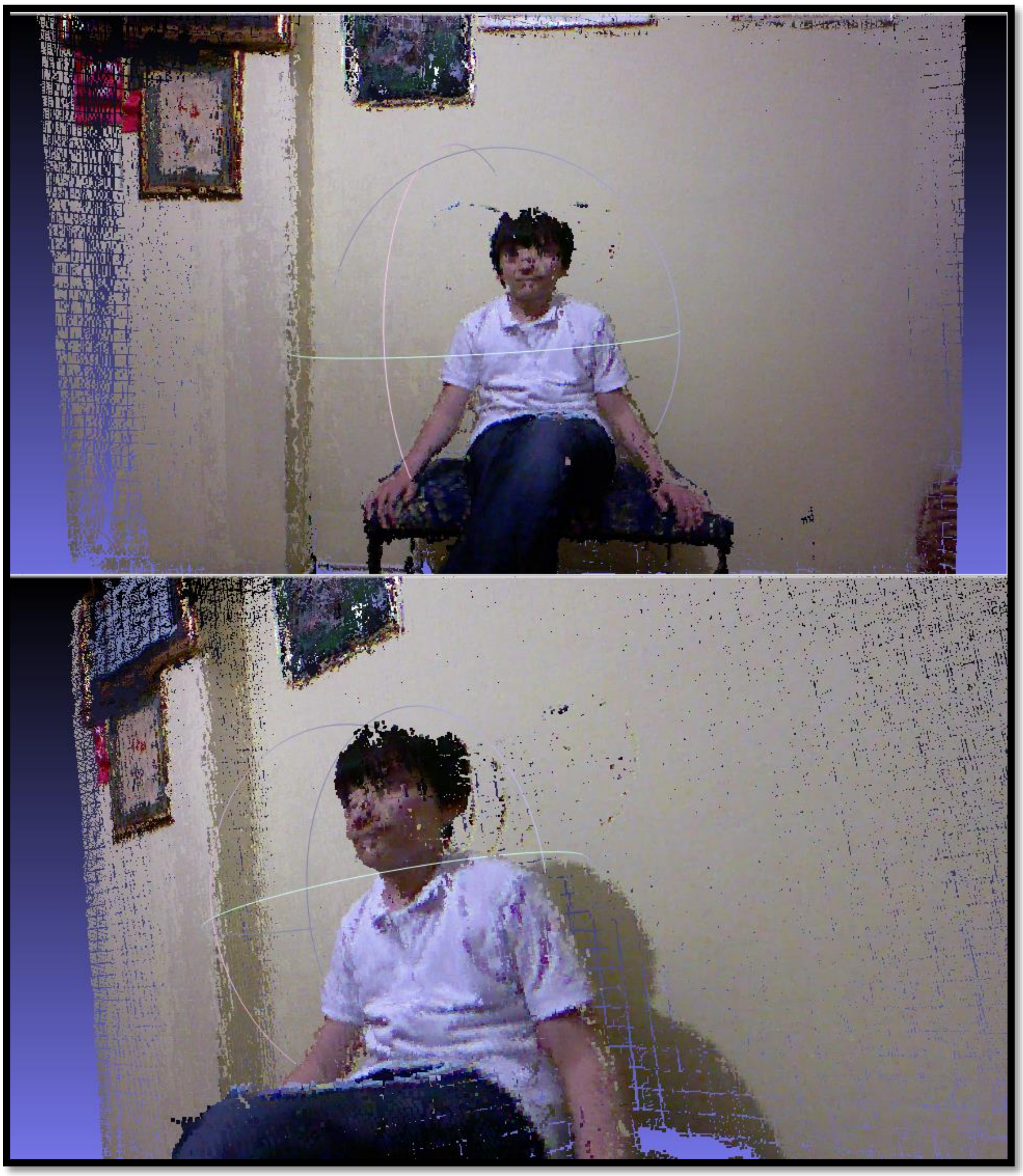
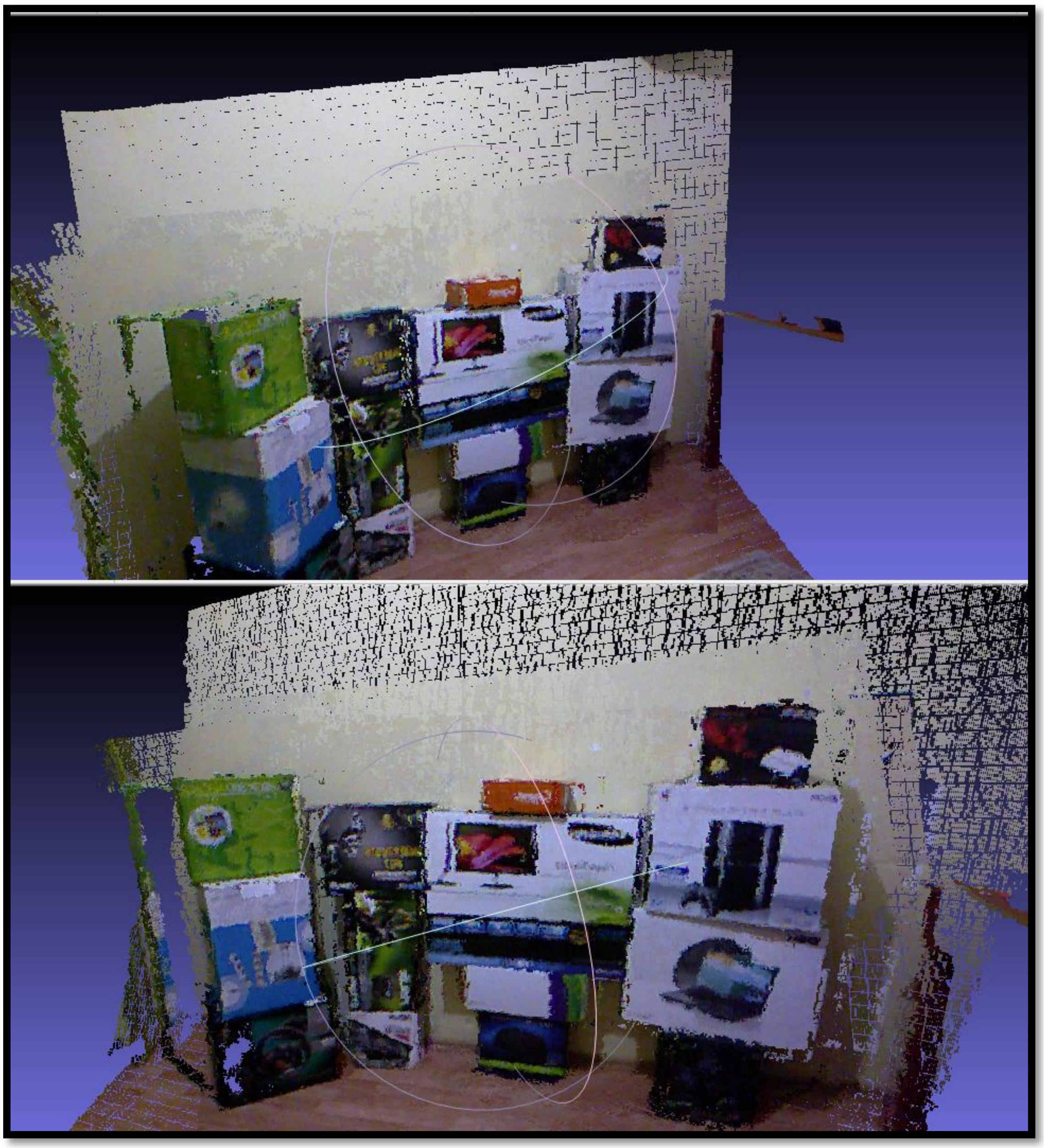


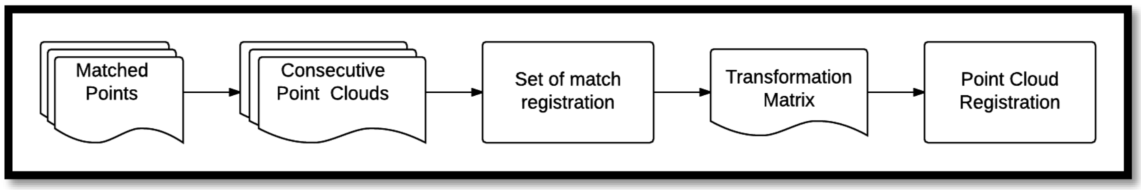
Figure 4.9: Box stack (5 images)



4.3.1 Conclusion

As stated in figure 4.10, the main goal is building a map from matched 2D descriptor points.

Figure 4.10: Process pipeline of rigid transformation



Process of Matching of two point clouds is developed by using "Arun KS, Huang TS, Blostein SD (1987) Least-squares fitting of two 3-D point sets" method [7]. After transformation matrix found, transforming each point of the last point cloud that comes from the last frame, two consecutive point cloud matches.

In spite of the noisy of Kinect depth sensor, inexact matches can occur. But that error is inversely proportional with number of matched descriptor points. In other words, the error minimizes, while number of matched descriptor points increase.

4.4 IMPROVING THE POINT CLOUD REGISTRATION

As every person interested computer science know, in the computer science history, there are too many of algorithms to solve daily problems or achieve greater complex structures. However they are not perfect. So as you read there are a lot of scientists try to improve or discover/create alternative to these algorithms to improve their structure. Another way to improving an existing algorithm is combining it’s alternative to surpass it’s weaknesses. More specifically, if two alternative algorithms/approaches cannot surpass each other, then another approach might be combination of these two approaches. In other words, if there are two ways of achieving a goal, and these two approaches is lack of performing a task that the other can perform, then marry these to surpass their disadvantages and put forward their advantages. In this section another point cloud registration algorithm is married with another called ICP.

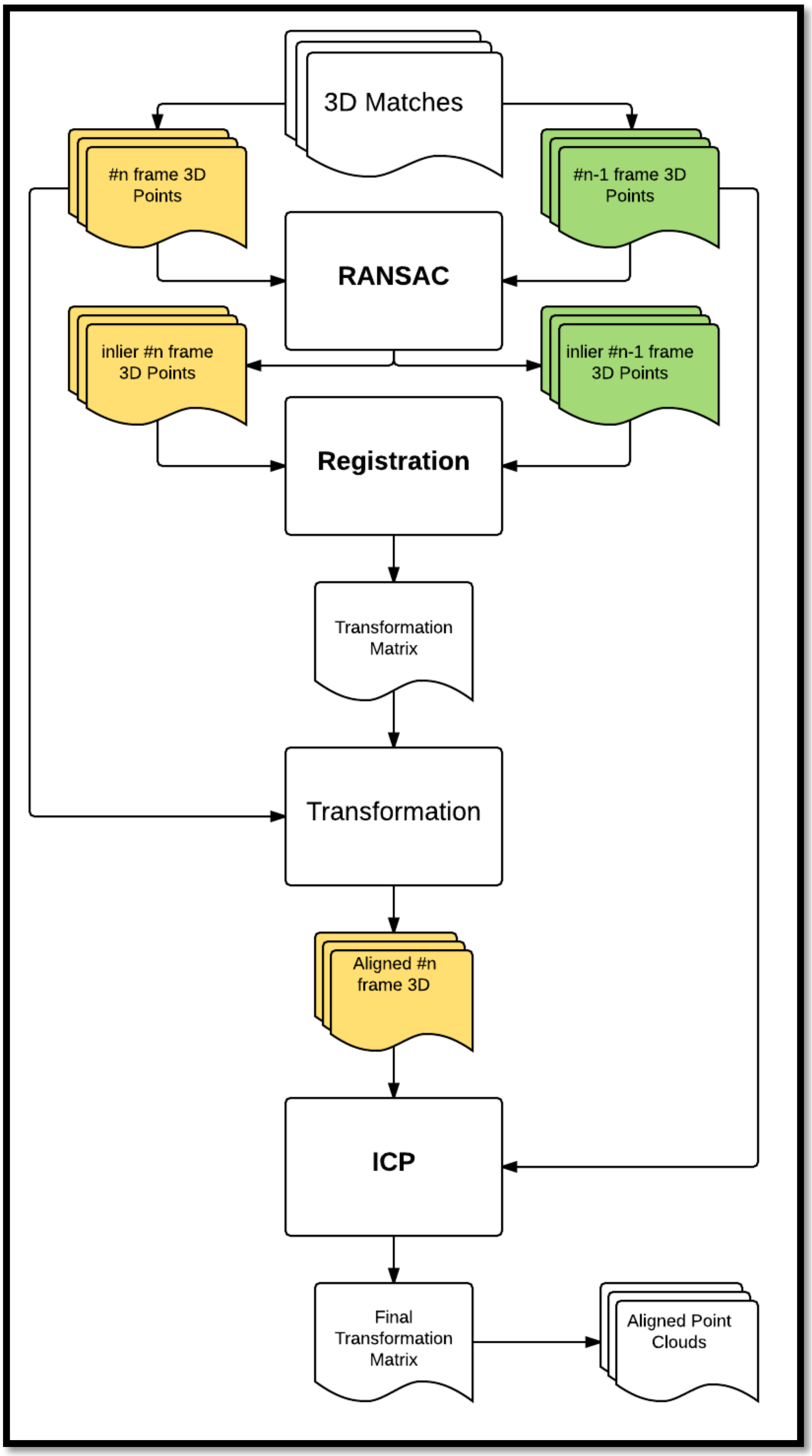
Point cloud registration pipeline is quiet complex. However this much complexity is not enough to create a good point cloud registration. So far, RANSAC was the core of the registration process. If RANSAC fails to detect true positives, then the whole pipeline fails. Thankfully, during our tests, RANSAC was able to eliminate true negatives effectively. But due to fact Kinect depth data noise and SURF Feature match pipeline flaws (check section 3), registration pipeline may not give good results.

Iterative Closest Point algorithm (see section 2.7) has some disadvantages like it is not a smart way to register point clouds. Because it just check every point in train point cloud to detect nearest point in test point cloud, and try to estimate transformation. If the nearest point is not the actual point in train point cloud, in other words, these points are actually not close each other, it leads to misalignment. So the train and test point cloud must be really close. However, if the ICP was applied standalone instead of the RANSAC would not be used to determine the matched points. So ICP is much simpler. But it lacks sense of true or false alignment. Because, it has no structure to measure alignment error. To summarize, that ICP is only good when the consecutive point clouds are really close and nearly aligned, otherwise the algorithm may not work as expected.

So far, we were able to register two consecutive frames. But registration may not good enough all time. That means mismatched point clouds and noisy point cloud. So standalone RANSAC is not every time and ICP is a very bad choice if the train and test point clouds are not close enough. Therefore as the final touch, the Iterative Closest Point algorithm is used to minimize the point cloud registration error. Thus the ICP would work expected and minimize the small errors. So the RANSAC elimination is essential for ICP.

The ICP integration demonstrated in figure 4.11, it takes the consecutive point clouds as input and returns the optimum transformation matrix between them as output. It is important to note that the consecutive test point cloud (tagged as #n frame) must be transformed by the transformation matrix which is obtained after 3D Rigid Transformation (check section 4.1) and RANSAC algorithms pipeline (check section 4.2). This is the part where the RANSAC elimination and ICP combined.

Figure 4.11: Iterative Point Cloud integration



4.4.1 Experiments

To see if the proposed solution for improving the point cloud registration works as expected the sitting person test data is used. This scene is more complex than the rest of the data sets. The other scenes are actually aligned nicely even without the ICP. But the sitting person data scene is the only scene that the RANSAC elimination didn't work and resulted in misaligned point cloud. In figure 4.12 the misaligned scene is shown. The picture is taken from above of the sitting person. So the white lines are the wall surface. So there are actual two lines which means alignment is faulty, even it can be detected by human eye. So the wall surfaces must be aligned and demonstrated as one.

Figure 4.12: Two consecutive point cloud registration before ICP

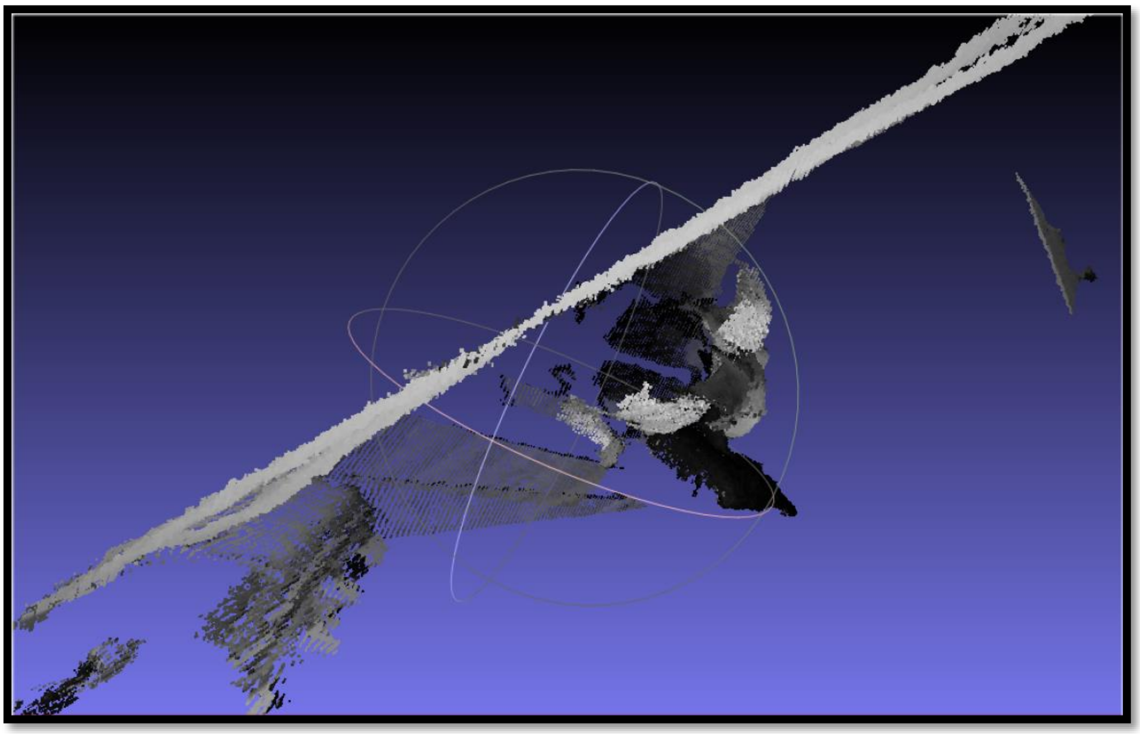
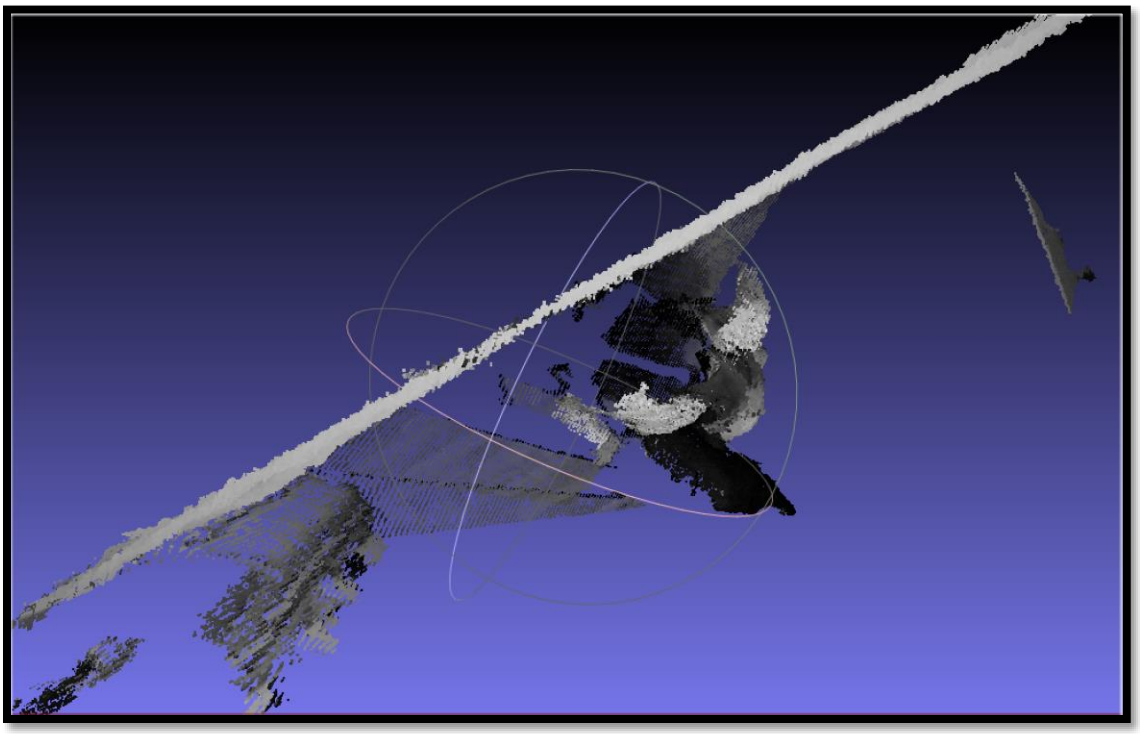


Figure 4.13: Two consecutive point cloud registration after ICP



The figure 4.13 displays alignment result after ICP applied. This time the white wall surfaces is aligned and demonstrated as one.

4.4.2 Conclusion

The ICP is combined with the 3D rigid transformation and RANSAC elimination pipeline. Thus the better aligned point clouds are obtained. Difference between figure 4.12 and 4.13 pictures that white wall demonstrates a better alignment after ICP. Before ICP it is easy to detect that the consecutive wall point clouds didn't aligned good enough.

To achieve the ICP integration the Point Cloud Library [11] C++ is used.

4.5 SUMMARY

In this section integration of all 3D space algorithm and approaches are explained. The main goal is to obtain point cloud registration in 3D space by using 2D matched features as inputs. The process pipeline is:

1. Get 3D matches through 2D matched SURF Features
2. Eliminate all outliers/true negatives by using RANSAC
    a. Generate 3D transformation matrix by using 3D Registration algorithm (section 4.1) to detect outliers
3. Generate 3D transformation matrix by using 3D Registration algorithm. This time use the inliers an input.
4. Transform the train (#n frame) point cloud by transformation matrix.
5. Apply ICP to minimize generated3D transformation matrix error and improve registration.
    a. Generate second transformation matrix to minimize first transformation matrix error.
6. Transform the train point cloud again with output transformation matrix of ICP.

5 DEPTHSCALE

Depthscale is a different approach to matching the local features. The main idea is detecting the features real world sizes and using this information as a criterion in the feature matching process. For Depthscale equation two parameter is required. These are feature's size, which can be obtained from SIFT and SURF features and Euclidean distance between camera and feature's centre position on point cloud.

$$Depthscale = DistanceToCamera * FeatureSize \tag{5.1}$$

By multiplying these two parameter as given in equation 5.1, Depthscale of the feature can be obtained. Depthscale feature is used to learn, if Depthscale values of a pair points are similar enough or not.

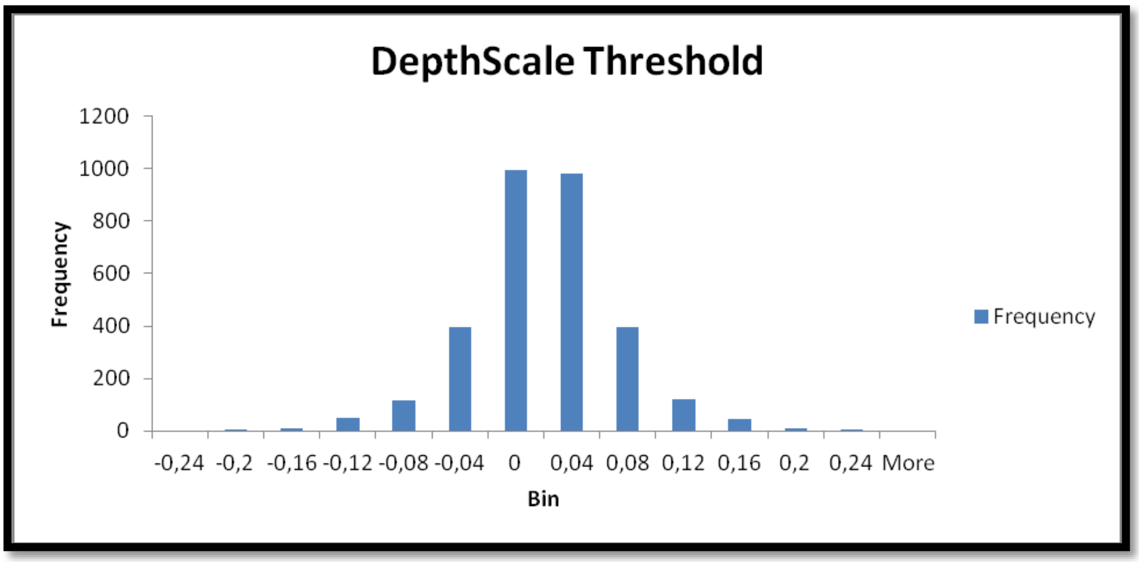
However it is important to note that, due to the fact that, every camera has different internal properties like focal length. Therefore the Depthscale measures of different cameras will result in false measurement.

5.1 FINDING DEPTHSCALE BOUNDARY VALUE

Noise needs to be taken into account in any robust vision system. As given in equation. 5.1, feature size in a fixed image scale, is multiplied by the depth of the feature. Hence we must take into account the magnitude of the Depthscale while comparing differences. Considering high depths and big scales will give higher errors due to multiplication, we propose normalizing the Depthscale difference with average Depthscale of the compared values in a feature match would give stable results. Below formulation describes the normalized Depthscale distance (NDSD) between two features of a single match:

$$NDSD = \frac{|DS1 - DS2|}{DS1 - DS2} \tag{5.2}$$

Figure 5.1: Depthscale threshold bound test



Equation 5.2 shows NDSD distribution of the inliers taken from an experiment (without taking the absolute value). Different test sets result in similar histograms. As can be seen in the figure 5.1, NDSD=0.25 seems to be a good threshold to eliminate outliers.

5.2 DEPTHSCALE METHOD in 3D REGISTRATION of RGB-D SENSOR OUTPUTS

In 2014, I and Kemal Egemen Ozden published a paper [10] about our Depthscale experiments to VISAPP 2014 conference. Similar to Lowe's measure we propose a new method to decrease the false matches in a match-set which is simple to compute. Assuming the 3D scans are taken from the same sensor, we can ignore the effects of RGB cameras internal calibration matrix.

Equation 5.1 represents the scale of the feature's support area in 3D world where feature's depth info comes from the 3D sensor and scale is the size of the support area of the feature on the intensity image. The equation directly results from inverse application of perspective projection. The features that belong to same 3D point can be considered to have same Depthscale. This measure can be applicable for many feature detectors, which returns a invariant support area. In contrast to long feature descriptors (e.g. SIFT is a 128 byte vector), Depthscale can be represented with a simple integer, which makes it more efficient to compare. Also since scale information it encapsulates is not related to intensity based feature descriptor, it gives extra information that can be exploited during matching. However due to its simplicity it would not be as descriptive as intensity features.

5.2.1 Combining Depthscale and Lowe’s Measure

In its simplest form Depthscale measure can be a simple efficient way to increase quality of the match-set. However it would be desirable to combine Lowe’s measure and Depthscale measure to increase matching performance. Here we will focus on simplest way of fusion, leaving the more advanced approaches to another work. Since there are already tools to find knn set for intensity features, we introduce the Depthscale measure to this pipeline. After finding the knn neighbourhood in intensity feature space, we apply Depthscale elimination for closest neighbours with a certain threshold. After that Lowe’s measure is applied. In a sense, we filter the match-set first with Depthscale measure, than Lowe’s measure, resulting in an “AND” operation.

5.2.2 Experiments

As a novel approach we cast the problem of removing outliers from a match-set as “binary classification” problem, taken from machine learning field. Indeed what Lowe’s measure does is, given a candidate match, checking the second nearest neighbour to mark it as inlier and outlier. In order to analyse that way, ground truth information is needed. Typically this is done manually in machine learning problems. However since we can have hundreds of matches given a pair of scan, we approximated this procedure by using all the candidate matches in an excessive RANSAC loop and detected inliers/outliers. This procedure might include one or two false positives or true negatives in the resultant ground truth but such amounts would have minimal effect.
We showed the effects of different thresholds for Lowe’s measure, pure Depthscale method and combined approach as a Receiver Operating Characteristics (ROC) curve. ROC curve is classical mechanism to show the performance (true positive vs. false positive) of a binary classifier for different threshold parameters. In our case inliers and outliers are labeled as positive and negative respectively. However we must note that there is no training happening here. The thresholds determine the classifier directly.

We tried to take experiment data in environments with different characteristics from 4 different test sets. Each test set has 5 (but only 2 are used in the graphs) scans and they are taken with a Kinect device from different angles and distances.

The ROC curves in Fig. 5.2, 5.3, 5.4 and 5.5 show that pure Depthscale measure gives better true positive ratio for high false positive ratios. Which means if we would like to easily eliminate many outliers while keeping almost all inliers, pure Depthscale will be give better results. For example a workstation based modelling application would choose that operating mode due to requirement of many inliers for better accuracy. However for lower false positive ratios Lowe' approach gives higher positive ratios. This is more suitable for applications which cannot tolerate high outliers in the data not to spend time in many RANSAC iterations. A robotics localization routine may opt for this operating mode. The combined approach converges to the best individual approach for different ends of the ROC curve, sometimes even beating them. However for left ends of the curve, it gives sporadic results for certain thresholds and converges to the worse approach occasionally. A deeper look shows that inferior results are caused by unrealistically tight thresholds for Depthscale such as 0.05. As a note, ROC figures are created by sampling various thresholds (for Lowe's measure between 0,65 and 1,0 and for Depthscale 0,05 and 0,6). Note that no ICP like refinement or surface reconstruction techniques are utilized here.

Figure 5.2: ROC curve of “Posters on a wall” experiment.

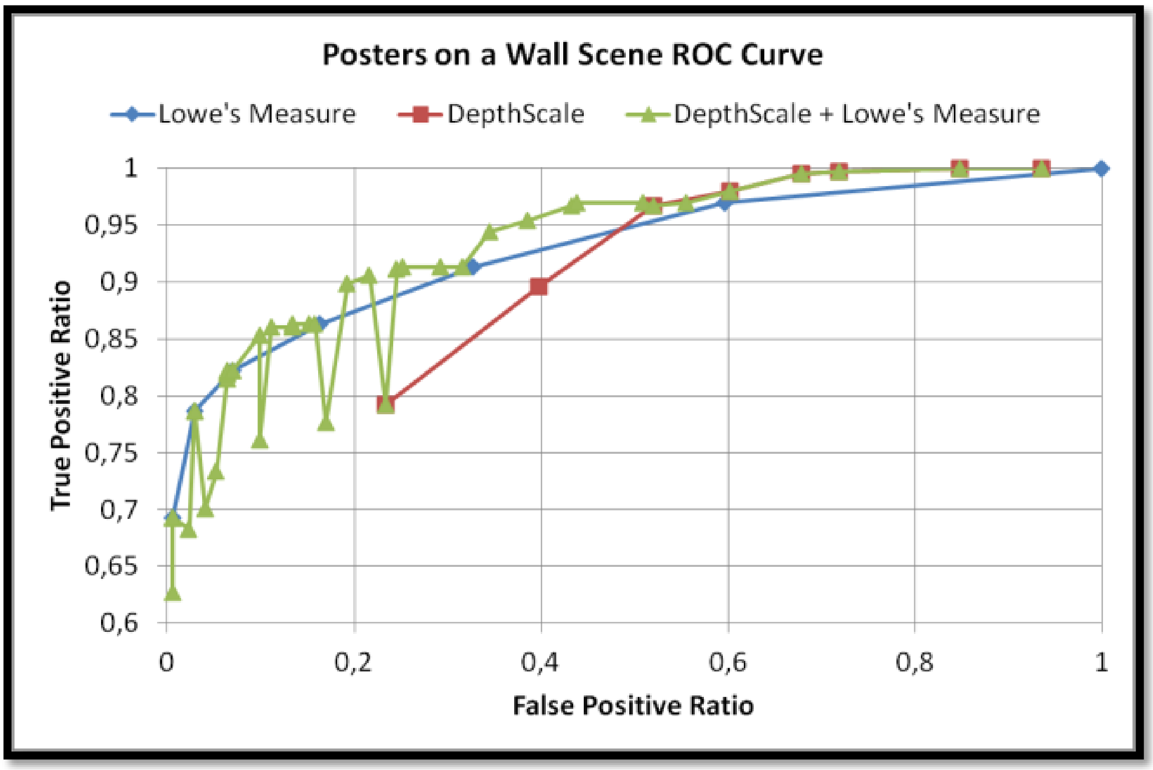


Figure 5.3: ROC curve of “Box Stack” experiment.

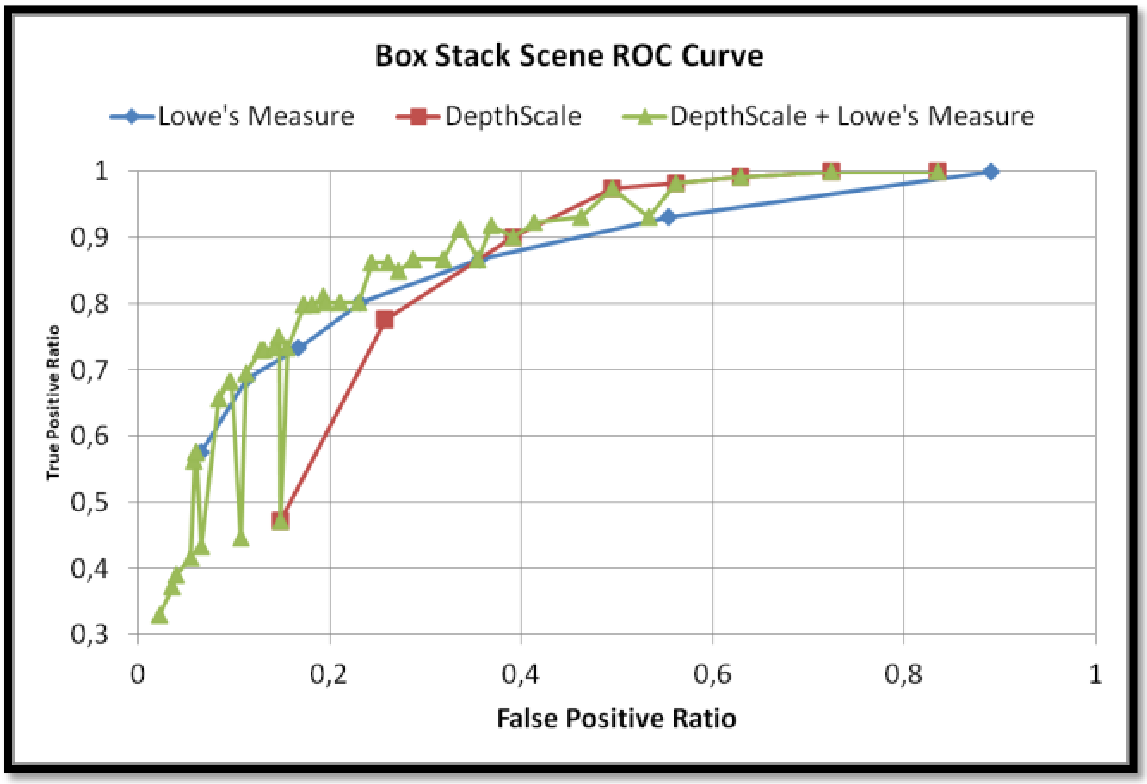


Figure 5.4: ROC curve of "Sitting Person" experiment.

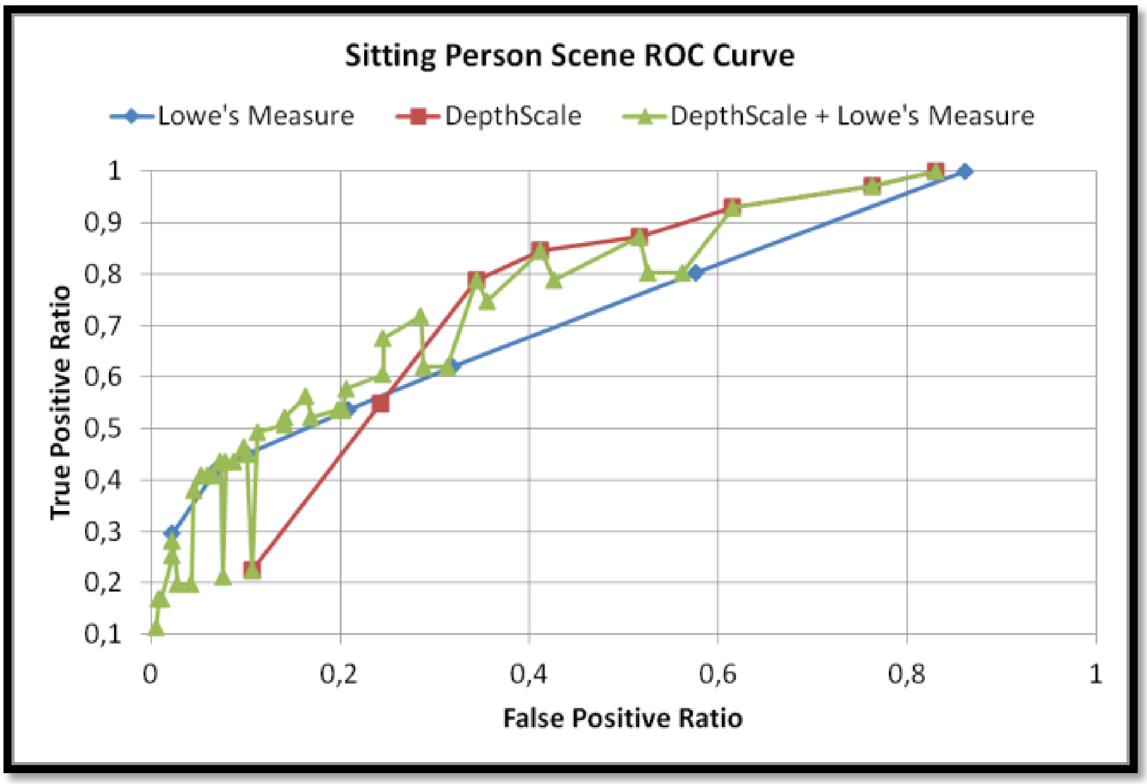
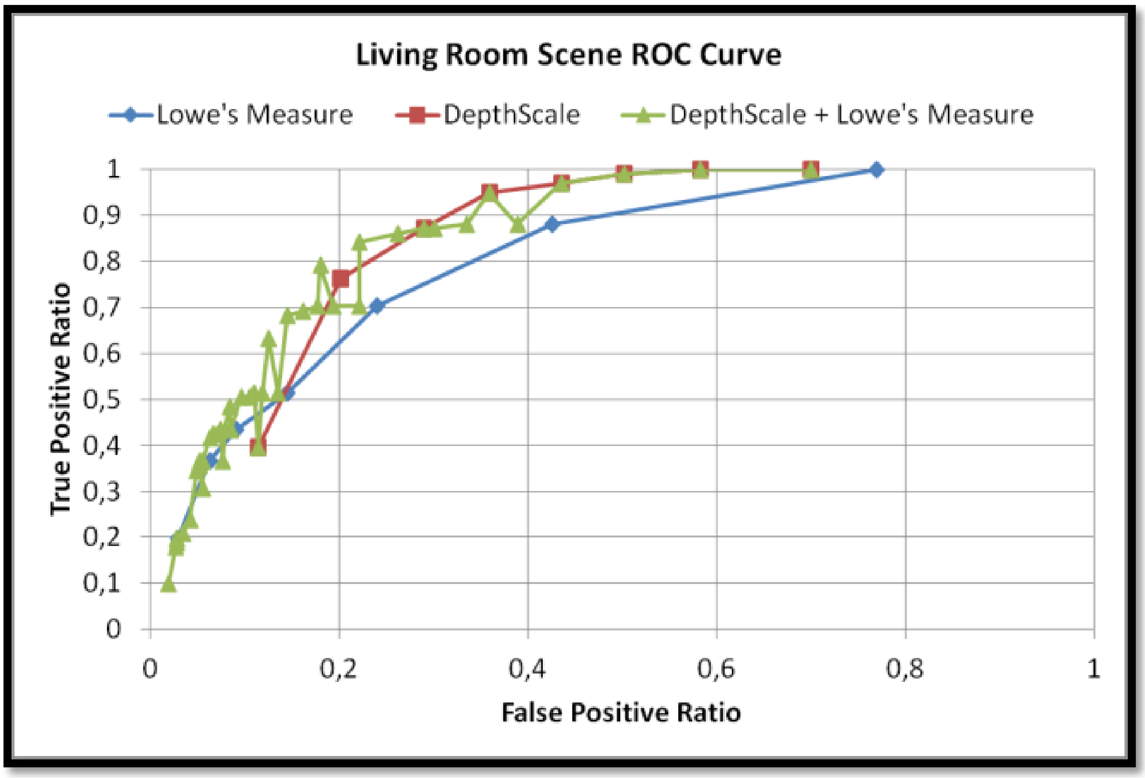


Figure 5.5: ROC curve of "Living Room" experiment.



5.2.3 Conclusion

We introduced a simple local image feature based 3D registration system for Kinect like sensor. It is designed to be built on available open source systems. We also introduced a new measure called Depthscale measure to increase the matching performance by exploiting the fact that depth measurements are available for the detected features. Experiments are presented to show the usefulness of this new measure. Eliminating outliers from a match set is cast as a classification problem and hence analysis is done through familiar ROC curves.

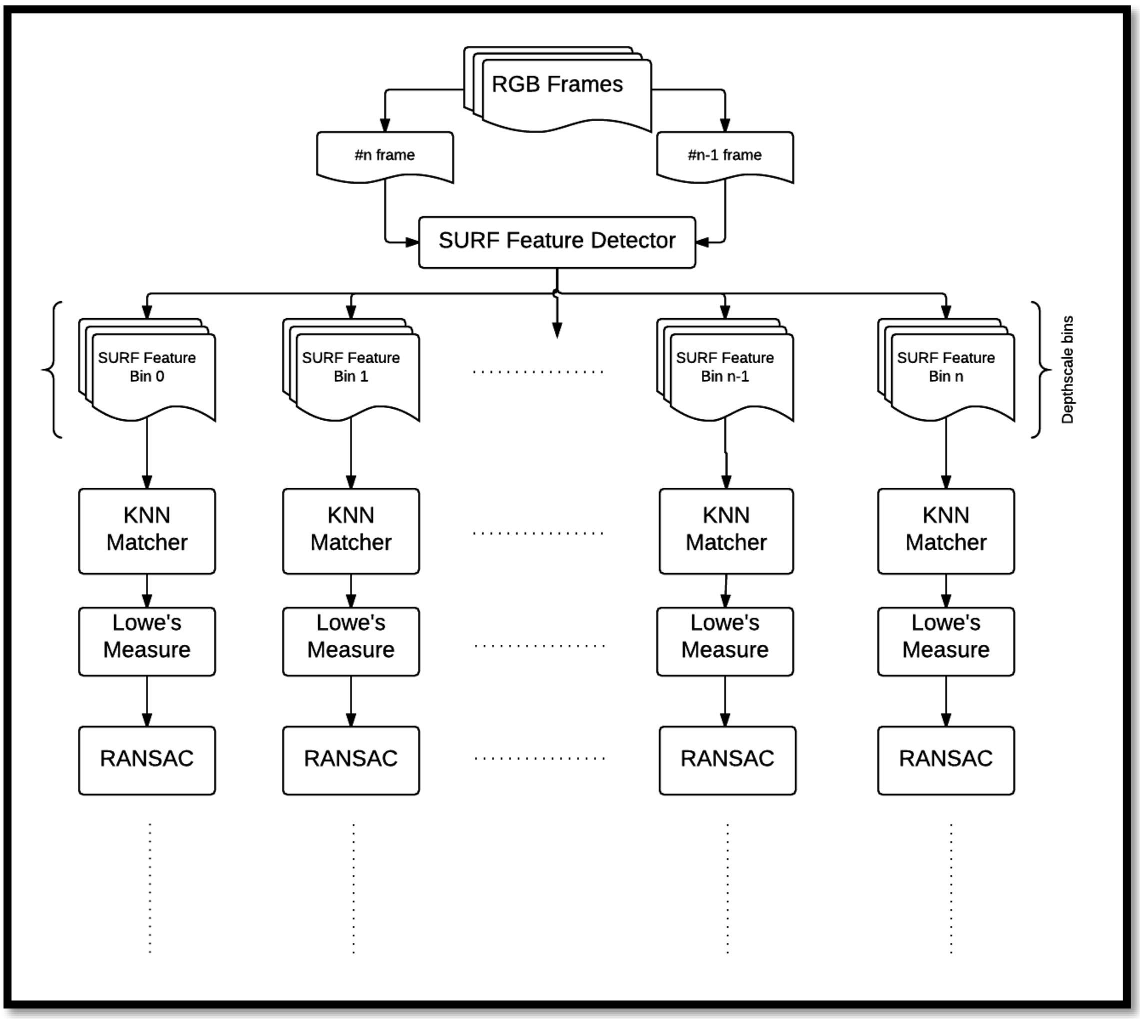
5.3 DEPTHSCALE BIN DIVISION APPROACH

Another approach (beside in section 5.2) was dividing the SURF Features into Depthscale bins. SURF Feature matching and Lowe's measure have no sense of real world feature sizes. So from that perspective, we thought that if the real world sizes of the features is used to determine small clusters of SURF Features it may be able to eliminate some false negatives and take them back as true positives.

5.3.1 Dividing train and test features into bins

The idea started with how to SURF improve SURF Feature matching process. When get into the details of the SURF Feature Matching, we encounter that SURF Feature matching focuses on 2D space not 3D. However with Kinect we have the actual 3D information. So we gather the all the information like depth data, to improve SURF Feature matching. At this point, we thought using the Depthscale not after but before the Lowe's measure and knn matching.

Figure 5.6: Depthscale bin division chart.



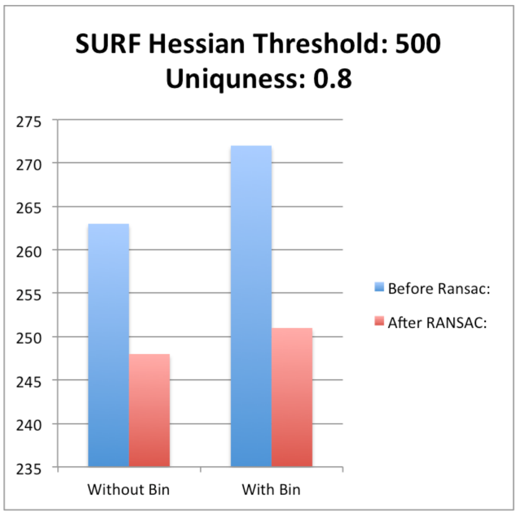
As displayed in the figure bin division approach is different than previous approach (see section 5.2). Depthscale measure is used the divide the SURF Features in into small clusters. After that division the life cycle of the small clusters are separated, until Transformation matrix generating step.

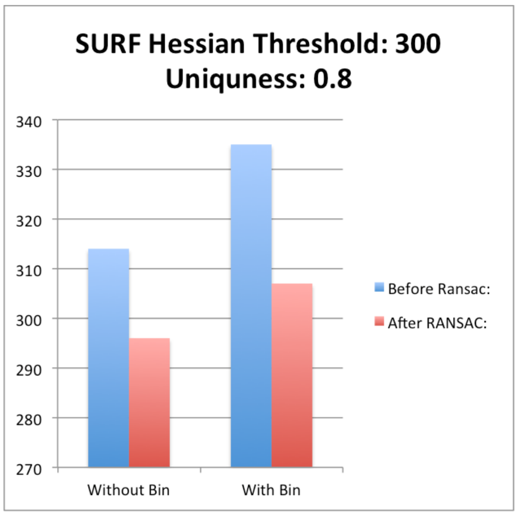
5.3.2 Experiments

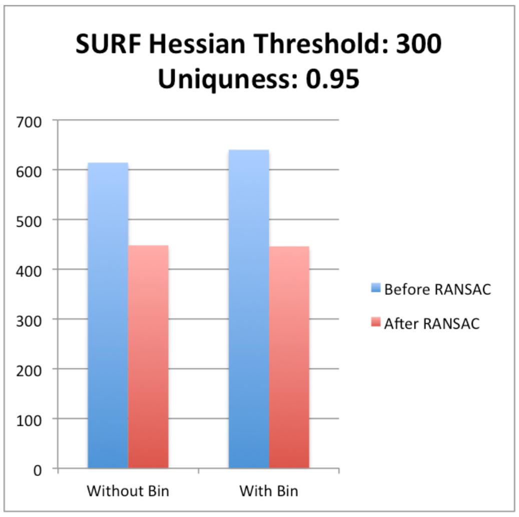
To test bin division approach the data sets within the section 5.2 are used.

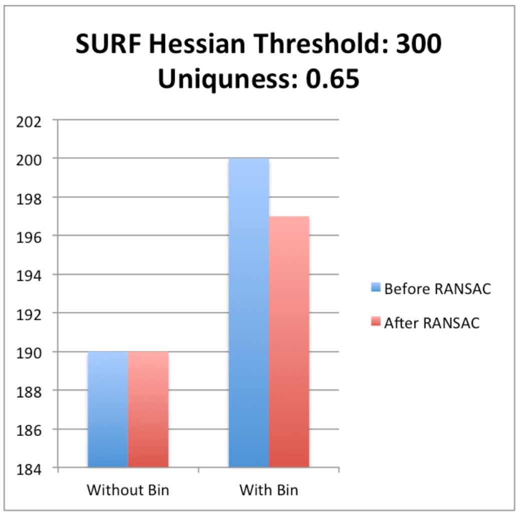
For this case we used the some combination of Lowe's measure and SURF hessian thresholds. But the NDSD is same as before 0.25.

Table 5.1: Boxes Scene 1 and 2 frame match inliers count charts for different threshold values



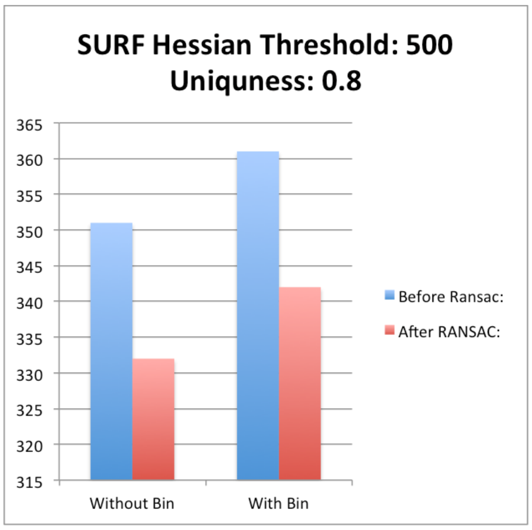


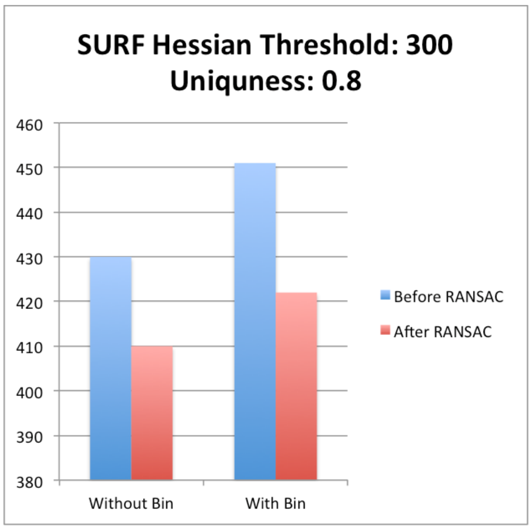


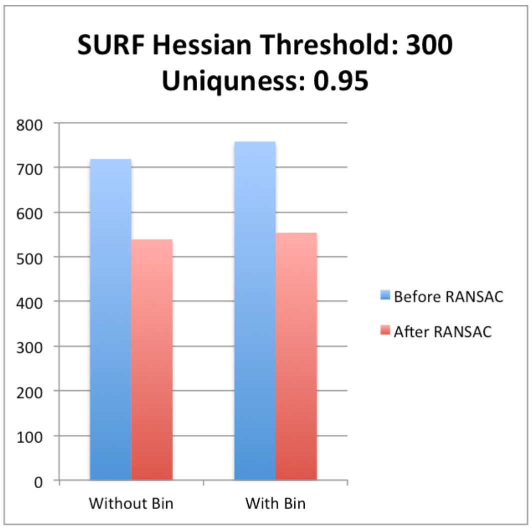


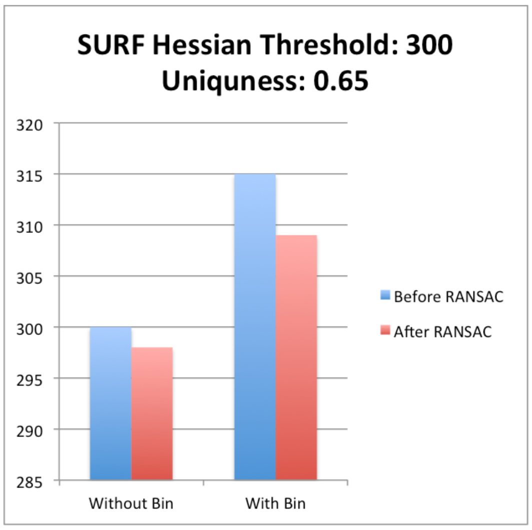
Boxes scene consists of a group of boxes and some text and images on them, which are very useful when it comes to detect SURF features on them. To see if the bin division performs as expected, different Lowe’s measure threshold are applied. Bin divison approach applied to boxes scene first and second frames. As given table 5.1 the bin division approach gives better results for all different thresholds before RANSAC. Moreover, when Lowe’s measure threshold gets lower the bin division approach performs better. After RANSAC the all test results promise better results, expect when Lowe’s measure threshold is to 0.95.

Table 5.2: Boxes Scene 2 and 3 frame match inliers count charts for different threshold values





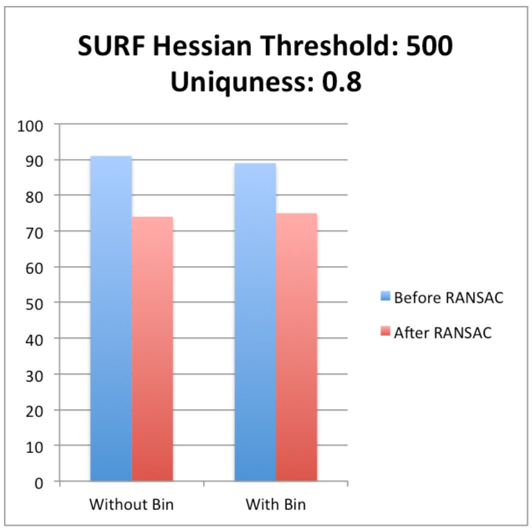


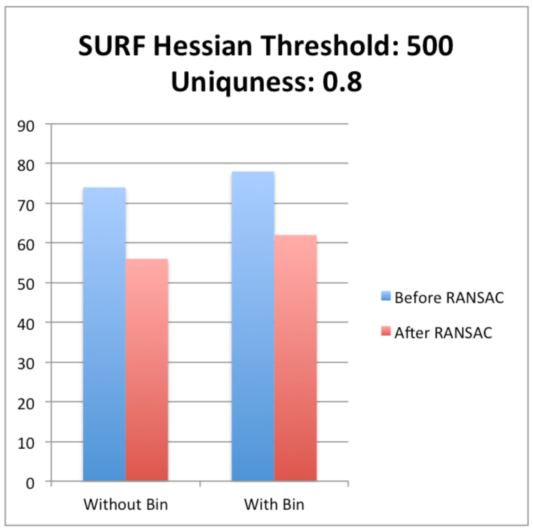


For the second the third frame matching, again before RANSAC, the bin division gives more inliers. Additionally and more importantly, after RANSAC, true positives count increases even where the Lowe's measure threshold is 0.95.

As displayed in table 6.1 and 6.2, boxes scene seems suitable scene for bin division approach. Most cases bin division provides better inlier count. But how about different scenes, which do not contain planar surface objects, like a human body.

Table 5.3: Sitting person scene frame 1 and 2 - 2 and 3 frame match tests. Inlier counts after and before RANSAC.





As given in table second figure, the 2nd and 3rd frame match gives some inlier number increase by using the bin division approach. On the other hand, for the 1st and 2nd frame match test, inlier count stays same. But bin division approach helped to decrease false positive number before RANSAC.

5.3.3 Conclusion

Bin division approach is a new way feature matching, it's main purpose is to decrease false negative and increase true positive. In the test case scenarios as given above, bin division does that. However it does not provide the expected impact on the end user when it comes to performance, because Bin division add extra computational cost to processor, At this point, bin division approach is recommended the applications which have no performance limitation and more importantly recall ratio is more important than the performance.

5.4 SUMMARY

To conclude Depthscale measure is an innovative elimination method for feature matching. It is simple multiplication two float number and to compare two Depthscale only subtracting two of them is sufficient. Despite the fact the other measures are requires much bigger data computations. More surprisingly it gives better results in 3D scene images when combined with Lowe's measure. As shown in figures above it provides a better true positive ratio.

As for Depthscale bin division approach seems legit, but it has additional impact on the performance when combined with Lowe's measure. Additionally, it may not demonstrate the expected results all time. So it is recommend for the applications, which are more focused on Recall than the computational time performance.

6 CONCULUSION AND FUTURE WORKS

With this work, building a 3d map of scanned environment by a RGB-D camera is explained by the following steps:

1) Feature Extraction in 2D
2) Feature Matching by using KNN match algorithm
3) Incorrect Match elimination
   a) Eliminate two similar too similar matches which are found by KNN match
   b) RANSAC elimination; detecting last inlier matches
4) Converting the depth frame and matched descriptor 2D points into real world coordinates
5) Finding the rigid transformation matrix between two point clouds and
6) Transforming the points that align consecutive point clouds.
7) And finally applying the ICP to improve transformation.

6.1 QUALITY IMPROVEMENTS

1) Due to the fact that, noisy depth data from Kinect depth sensor results in inaccurate point cloud. Therefore finding more accurate transformation matrix is unavoidable. Moreover this accurate transformation matrix can be achieved by RANSAC [2] elimination for 3D set of matched points.

2) Depthscale measure and Depthscale bin division improves the quality of inlier matches. Moreover integration with Lowe’s measure is maximizing the true positive ratio of descriptor matches.

6.2 PERFORMANCE IMPROVEMENTS

In order to decrease the performance needs, the Depthscale Method decreases the outlier matches, thus the application deals with the less the outliers, in other words decreasing the outlier matches means RANSAC and the other methods do not use the unnecessary matches, which increases the application performance.

To summarize, in this project, an Intel i3 2.4 Ghz two core cpu is used. The application spends 2-4 sec for per frame transformation. However because of the graphical calculation performance of GPUs, can provide same results with in less time. Due to the fact that OpenCV libraries [6] supports CUDA technology, also SURF Feature extraction process designed for both CPU and CUDA supported hardwares. Moreover the repeated computational parts of the process like calibrating the RGB and depth frames, transformation matrix of 3D points appears the obvious candidates for GPGPU approach. In short applying GPGPU approach to this project will result in less computational time, and increase responsiveness.

6.3 ADDITIONAL TASKS

To complete the 3D rigid scanner there is an additional step, which is surface reconstruction. With surface reconstruction the aligned/registered point clouds can be displayed in a better way thus human eye can observe the outputs easily. Another benefit of surface reconstruction is that the 3D meshes (output of the Surface Reconstruction) can be used in 3D games and 3D printer. Nowadays printing arms for the people who lost their limbs is very popular, after applying the Surface Reconstruction people can print their arms or legs by scanning the opposite limbs. Thus the prosthetic limbs can look more natural.

REFERENCES

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.*, 110:346–359, June 2008.

[2]Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Commun. ACM, 24(6):381–395, 1981.

[3] Nicloas Burrus. Kinect Calibration Parameters. http://nicolas.burrus.name/index.php/Research/KinectCalibration (last access: december 2012)

[4] D.W. Eggert, A. Lorusso, R.B. Fisher. Estimating 3-D rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications* 9: 272–290, 1997

[5] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. 2004.

[6] OpenCV libraries for C++ http://opencv.org/ . (Last access: November 2015)

[7] Arun KS, Huang TS, Blostein SD Least-squares fitting of two 3-D point sets. IEEE Trans Pattern Anal Machine Intell 9:698–700 1987

[8] Virgile Högman. Building a 3D map from RGB-D sensors. Master's Thesis in Computer Science. Computer Vision and Active Perception Laboratory Royal Institute of Technology (KTH)

[9] Zhengyou ZHANG. Iterative Point Matching for Registration of Free-form Curves, 1992.

[10] Ismail Bozkurt, Kemal Egemen Ozden. Depth-scale method in 3D registration of RGB-D sensor outputs, 2014. Bahcesehir University. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7294846 20

[11] Point Cloud Library. http://pointclouds.org . (Last access November 2015)

[12] Eigen C++ Library http://eigen.tuxfamily.org/index.php?title=Main_Page . (Last Access November 2015)

[13] Lowe, D.: Object recognition from local scale-invariant features. In: ICCV. (1999)

[14] Ben Bellekens, Vincent Spruyt, Rafael Berkvens, and Maarten Weyn. A Survey of Rigid 3D Pointcloud Registration Algorithms, CoSys-Lab, Faculty of Applied Engineering University of Antwerp, Belgium.

[15] https://dev.windows.com/en-us/kinect , September 2015

[16] http://www.vision.ee.ethz.ch/ , October 2015

[17] https://en.wikipedia.org/wiki/Pinhole_camera , November 2015

[18] http://code.opencv.org/projects/opencv/wiki/2012 , October 2015

CURRICULUM VITAE

Name Surname : İsmail Bozkurt

Address : Altmaltız Sok. Erdemli Apt. D:2 No:3 Halıcıoğlu Mah. 34445 Beyoğlu / İstanbul Turkey

Birth Place and Date : Ankara Altindag / 1989

Languages : Turkish, English

High School : Science Major - Guner Akin

Bachelors Degree: Computer Engineering Bahcesehir University

Mobile : +90(506) 285 16 88

E-mail: ismail.bozkurt@stu.bahcesehir.edu.tr, ismailbozk@gmail.com

Computer Skills :

Programming Languages: C++, Objective C, Swift, C#, Java.

SDKs: iOS, Android.

Computer Vision Libraries: OpenCV, Eigen, Point Cloud Library.

GPGPU and Graphics Programming: iOS Metal, Nvidia CUDA.

Professional Experience :

I have worn several hats in my career, like researching, software engineering and even a little bit marketing. As a result, I think, I have ability to overcome different challenges. So far, I built software libraries and published several mobile applications.

Monitise	May 2014 - Present
Wallit	September 2013 - April 2014