Optimization of the Single Point Active Alignment Method (SPAAM) with a Random Forest for accurate Visual Registration

Sravya Kanuganti
sravyak@iastate.edu

Follow this and additional works at: https://lib.dr.iastate.edu/creativecomponents

Part of the Computer Sciences Commons

Recommended Citation

https://lib.dr.iastate.edu/creativecomponents/322

This Creative Component is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Creative Components by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Optimization of the Single Point Active Alignment Method (SPAAM) with a Random Forest for accurate Visual Registration

by

Sravya Kanuganti

A Creative Component submitted to the graduate faculty in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Dr. Rafael Radkowski, Major Professor
Dr. Jin Tian, Major Professor
Dr. Samik Basu

Department of Computer Science
Iowa State University
Ames, Iowa
2019

Copyright © Sravya Kanuganti, 2019. All rights reserved.
ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratitude to Dr. Rafael Radkowski for his guidance, patience and support throughout the course of the research and writing this report. I would like to thank my committee members Dr. Jin Tian and Dr. Samik Basu for their help and support. I would also like to thank Danfoss Power Solutions for the equipment used in the research.
ABSTRACT

The research addresses the visual calibration of head-mounted displays such as the HoloLens. The HoloLens is an optical see-through viewing device that allows a user to experience the real world populated with virtual objects. These virtual objects need to be correctly aligned with physical objects in the environment to experience a visually appropriate scene. However, several factors, such as an outside-in tracking system, tracking errors, the user's eye position, and others degrade the alignment between the virtual and physical object. A popular calibration method to correct this misalignment is the so-called Single Point Active Alignment Method (SPAAEM) [1]. It allows one to improve the alignment by measuring and correcting the alignment error. Nonetheless, one encounters alignment errors since, SPAAM assumes a constant error between the physical object, the display, and the user's eye. Modern low-cost tracking systems such as based on RGB-D cameras (e.g., Kinect) come with dynamic errors. Consequently, SPAAM cannot yield the required accuracy; theoretically, dynamic errors require a dynamic calibration. The objective of this research is to study the improvement a dynamic error calibration can yield regarding alignment and registration accuracy. To improve the visual experience for a user, a random forest method will be adopted for this purpose. The hypothesis is that the random forest can dynamically select the best SPAAM calibration matrix with respect to the relative position of the user and a physical object. Experimental results demonstrate improvement by a factor of four; thus, indicate that random forest is an appropriate method to mitigate object misalignment due to dynamic tracking errors.
CHAPTER 1. INTRODUCTION

Augmented reality (AR) is a human-computer-interaction technology that allows a user to experience the real world populated with virtual objects. Therefore, an augmented reality system superimposes computer-generated graphics models on real-world objects such that a graphical model is accurately aligned with the real object. AR requires viewing devices that allow the user to experience the enhanced reality. The HoloLens is such a device. However, the current version of HoloLens does not provide any means to track individual physical objects in real space. Therefore, we use an outside-in tracking solution, based on the Kinect camera and point clouds extracted from Kinect images. Our solution can detect and track the object of interest in the point cloud. The 3D tracking information is used to superimpose a real part with virtual information, which are further visualized via a HoloLens display. However, the HoloLens and the Kinect use different coordinate system which need to be aligned. This difference requires a mathematical transformation to render the models so that a user experiences them from the right perspective when seeing them through the HoloLens. Here, several issues need to be addresses such as tracking errors, calibration tolerances, the user's eye position in front of the display, and others, which all together degrade the alignment precision between the virtual and real object.

The so-called Single Point Active Alignment Method (SPAAM) method is the state-of-the-art method to correct the alignment error. It calibrates the display by measuring the deviation between real and virtual object. Its process requires a user to calibrate the display by manually aligning the position of certain points in the real world with virtual object points. SPAAM yields a calibration matrix which improves the alignment. Although SPAAM is popular in the field, it was
designed to improve a static offset. Outside-in tracking such as with the Kinect introduce a set of new challenges which SPAAM does not address.

The objective of my research is to study the improvement a dynamic error calibration can yield in terms of alignment accuracy and registration. We execute this by computing calibration matrices using SPAAM for multiple positions of the user with respect to the physical object. A random forest is trained during system set up, with the user position as an input and a SPAAM calibration matrices as the output. During application runtime, when the user moves, his/her real time position are passed to the random forest, and it returns the appropriate calibration matrix for the relative position of the user.

The hypothesis is that the random forest can dynamically select the best SPAAM calibration matrix with respect to the relative position of the user and a physical object. To determine the accuracy gained, we compare the outcome yielded with the random forest to a conventional SPAAM approach. The approach is an experimental on utilizing landmarks. A dataset incorporating HoloLens to real object relations will be obtained. Several points on the real object will be designated as landmarks; their exact position is known with respect to object coordinates. Virtual points will be rendered at the exact same position using no calibration, SPAAM, and the random forest. Ideally these positions match the landmark positions; the mentioned error prevent this. The improvement each method yields will be measured as a root-mean-square-error and compared. The results will allow me to accept or reject the hypotheses.
CHAPTER 2. RELATED WORK

Our research addresses the optimization of augmented reality object registration using and advanced method of SPAAM. Since the SPAAM approach has been published, researchers already contributed several iterations of SPAAM like stereo SPAAM for video-displays [4], and SPAAM utilizing a depth camera on HMDs[6]. Also, researcher already investigated the impact of user position during the calibration procedure [7], and several others. Kenneth et al. [7] examined that the SPAAM calibration when done at an arm length shows significance difference compared to SPAAM performed with room scale alignments. Additionally, there is no significance impact on the calibration due to the user pose (sitting and standing). The authors also performed SPAAM calibrations in a controlled environment where they mimicked the user with a rigidly mounted camera; this condition removes the effect of noise from uncontrollable postural sway [7]. In both the environments, the distance between user and the calibration tool affects the outcome significantly.

SPAAM involves user interaction during the calibration procedure, when calibrating the head-mounted displays. Thus, a robust automatic calibration procedure suited for nontechnical users has been developed coined as the Interaction Free Display Calibration (INDICA) method [5]. The authors reported that this calibration system produces a calibrated projection matrix for both eyes automatically without the need for any user interaction. The system also updates the calibration on-line, which allows the system to accommodate unpredictable errors, such as movement of the display on the user’s head during use.
CHAPTER 3. RESEARCH SETUP

The following section describes the setup used for this effort. The next section introduces the Single Point Active Alignment Method (SPAAM). Section 3.2 describes the test setup. SPAAM requires on to determine 2D-3D point-pairs. The 3D points are idealized model points, and the 2D points are image points. The following Section 3.3 and 3.4 explain how these point-pairs were obtained. Section 3.5 describes the random forest approach to dynamically determine the right correction matrix.

3.1 Single Point Active Alignment Method

In order to experience an effective augmented reality environment, any computer-generated virtual model should be accurately aligned with its physical object. This requires display calibration during system set up. SPAAM is a calibration method that facilitates this calibration. Its procedure yields a calibration matrix which ensures that the virtual and physical model appear aligned in the display.

The SPAAM set-up process works with 2D-3D point pairs to obtain a correction matrix $T_{spaam}$. Mathematically, it maps the known 3D points to their corresponding 2D points in the image such that

$$P_I = T_{spaam}P_M$$

where $P_I [u_I, v_I]$ is the accurate image point of $P_M [x_M, y_M, z_M, 1]$ (a point on the real model w.r.t camera tracked using magnetic tracker). $T_{spaam}$ is 3 x 4 correction matrix corresponding to the
overall camera transformation instead of estimating intrinsic and extrinsic parameters separately.

Here, we use SPAAM to correct only the extrinsic parameters, where we calculate a correction matrix \( C \) that maps the known 3D points on the virtual model to the corresponding 3D points on the real object.

\[
P_R = C P_M
\]

where \( P_R \) is the accurate point on the real model where \( P_M \) the corresponding point on virtual model should lie. We ensure that both are in the same coordinate system. \( P_R \) is calculated using Marker Tracking. \( C \) is a 4 x 4 matrix that corrects the pose/rigid transformation of the model (extrinsic parameters). \( C \) is of the form \([R|t]\) where \( R \) is the correction for the rotation and \( t \) is correction for the translation.

\[
C = \begin{bmatrix}
c_{11} & c_{21} & c_{31} & c_{41} \\
c_{12} & c_{22} & c_{32} & c_{42} \\
c_{13} & c_{23} & c_{33} & c_{43} \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

This can be rearranged in terms of unknown parameter vector \( x = [c_{ij}]^T \) (\( x \) is all the entries of \( C \) put into a column vector) to be estimated into a linear equation to be solved

\[
Bx = A
\]

where the coefficient matrix \( B = \)
The matrix $B$ has $3n$ rows, three for each datapoint and 12 columns. Solving this equation gives us the correction matrix $C$. There need to be at least 6 different points along all the axes to achieve correct calibration. Here, we use 8 different points covering all the directions.

### 3.2 Environment Setup

Figure 1 describes the setup I use for this thesis. It is a part of a pump housing sitting on an assembly cart. The objective is to superimpose a 3D model correctly aligned to this cart. Figure 2 describes the working of this setup. It shows the physical object is tracked using Kinect camera and transfers the pose of the object to the HoloLens. The HoloLens places a fixed coordinate system (World Anchor) at the location of the Kinect and renders the virtual model with respect to the anchor. Figure 3 shows the world anchor placed near the Kinect.

![Figure 1. Physical object with the corners marked](image)
Figure 2. Working of the tracking system

Figure 3. Position of the world anchor and the Kinect
3.3 Determination of the positions on the Virtual Model

SPAAM requires one to determine the current pose of a rendering. Since it is unpractical to use a complex 3D model for this purpose, one reduces the 3D model to a set of idealized points. These points are idealized points, which give the current position of any augmented model. The position of each point depends on the current object locations, thus, on the tracking data incorporating all tracking errors. Note that the points used in this research align with the corners of the cart as described in Section 3.2.

We currently use a feature-based tracking method to track physical assets such as the piston motor housing. It employs a point cloud from an RGB-D camera. Figure 4 shows the a registration of a 3D model with the physical object using Kinect data for tracking. Figure 5 shows the coordinate frames involved in tracking the 3D pose. All measurements are performed in camera coordinate space. The pose is given as rotation $Q_K = [x_K, -y_K, -z_K, w_K]$, and as a translation $T_K$, a 3x1 translation vector $T_K = [-x_K, y_K, z_K]$. Please see [2] for any details about the tracking algorithm and tool.

With the pose at hand, we describe the expected position of the 3D model utilizing a set of points that are mathematically aligned with the corner points of the test setup. We obtain the position of each point in camera coordinate frame by:

$$P_K = Q_K P_M + T_K$$

with $P_M$, the position of the point in object local coordinate system and $P_K$ is its position in the Kinect camera frame. Note that the current tracking algorithm tracks objects using right-hand
coordinate system, whereas HoloLens 3D renderer uses a left-hand coordinate system. So, we changed the quaternion and translation to left-hand coordinate system above.

Figure 4. Object tracking using Kinect
3.4 Determination of the positions on the Real Object

Where the measurement described in Section 3.3 represents the current erroneous position of the model, the second set of points describes the expected location of each point. Note that these points still represent an idealized 3D model. For this effort, we use marker tracking to measure the position of each point with respect to the HoloLens.

Figure 6 illustrates the involved objects, the involved coordinate frames, and the coordinate frame transformations. We measure the location of each point using a pattern marker in marker coordinate frame. Since we use the HoloLens as a display device, we render all 3D models with respect to the HoloLens display coordinate system. Thus, each object point needs to be represented in this HoloLens frame. Since SPAAM work only in one coordinate frame, and the
object pose is tracked in the Kinect camera reference frame, we need to transfer each measurement into the Kinect camera reference frame.

Figure 6. The involved coordinate systems and coordinate systems transformations

Figure 7 shows the point measurement process using a marker; the photo was captured with the HoloLens’s RGB camera. The process is straightforward. One places the tip of the marker at the required corner point of the object and can take a photo from this marker with the HoloLens’ camera. A pattern detection algorithm can detect the marker and determine the pose via a Perspective-N-Point (PNP) algorithm. This process is repeated for all essential points on the object of interest.
Next, with all points at hand, each point is transferred into a common camera coordinate system. The Kinect reference frame in our case. To determine this transformation, we need to track the HoloLens with respect to the Kinect camera. However, HoloLens doesn’t have a fixed coordinate system/origin. A practical solution for this problem is a so-called HoloLens world anchor, a fix coordinate reference frame in the HoloLens’s internal map. We align this world anchor manually with the position of Kinect camera. Note that the camera coordinate system and the HoloLens coordinate system will not perfectly align, which causes an additional pose to shift our SPAAM approach should resolve.

Mathematically, we can transform each measured marker point into the Kinect reference frame using

$$P_R = Q_A P_{Mark} + T_A$$

with $P_R$, the position of a reference point on the real object, $Q_A$ and $T_A$ represent the orientation and translation of the HoloLens w.r.t anchor, and $P_{Mark}$, the measured marker position.
3.5 Random Forest for Visual Registration

Although SPAAM is used commonly for display calibration, it is designed to correct a static offset since the original developers considered high-accurate tracking devices. Whereas, low-cost devices such as Kinect produce dynamic errors. Additionally, the spatial relation between the headset and the tracking system origin also results in an additional tolerance error. A regular SPAAM approach cannot mitigate this problem since its correction matrix is calculated from one aspect angle; this angle changes when one moves around the object. A user experiences this error as drift of the virtual object.

The approach of this effort uses a random forest to address this issue. The abridged version: we compute calibration matrices using SPAAM for multiple viewpoints. Then a random forest is trained with the user's positions as input, and calibration matrices as output. During application runtime, when the user moves, we use the random forest to approximate the best matrix depending on the viewpoint. The random forest yields the appropriate calibration matrix for the relative position of the user.
CHAPTER 4.  RANDOM FOREST FOR DYNAMIC VISUAL REGISTRATION

The following sections describe the random forest and its architecture. Section 4.1 explains its basics, Section 4.2 explains the measurement of the relevant attributes such as the input position for random forest training, Section 4.3 describes the utilized Random Forest, and Section 4.4 and 4.5 elaborate on several design decisions made during this study.

4.1 Random Forests

A random forest is a learning method used for classification, regression, and other tasks. It operates by constructing a multitude of decision trees at training time and outputs the class (classification) or mean prediction (regression) of the individual trees [3]. Each decision tree formulates some set of rules to perform predictions. It predicts an output for the given instance and all the outputs from each decision tree are considered for the final result of the random forest. The random forest approach - combining values from multiple decision trees - prevents overfitting, a common shortcoming of decision trees.

4.2 Construction of a Random Forest

The random forest incorporates multiple decision trees, and each tree trains with a different, random set of input data. For this study, the training data used for the construction of the random forest includes the position $x$ of the user with respect to the object of interest as the input. Each input vector is linked to a particular SPAAM correction matrix which works best for the current position. Thus, the output $y$ of the random forest is the best or approximated SPAAM correction matrix. From an implementation point-of-view, the used random forest stores unique
label corresponding to the related correction matrix. We use labels to represent correction matrices to reduce the resource need per node.

During training, the size of the training set for each tree in the random forest is the size of the original dataset. However, the tuples in each of the training set are randomly chosen from the original dataset. Consequently, duplicate tuples in each training set are possible. However, the number of input features used to determine the split in the training data is two, i.e. two features out of the three are chosen randomly from the input features (position of user w.r.t object) during the training of each tree. This ensures that all trees in the random forest differ from each other.

The prediction of the output correction matrix is done using bagging predict: the outputs from each decision tree (duplicates are removed) are considered to predict the final output from the random forest.

4.3 Determination of input for Random Forest

The input for the random forest is the position of the user w.r.t the object of interest. The position of the user is the position of the HoloLens w.r.t the object. The object is tracked using Kinect and the HoloLens is tracked using the anchor placed near the Kinect. So, we calculate the position of the user using

\[
\text{Input position} = Q_K^{-1}(T_H - T_K)
\]

where \(Q_k\) and \(T_k\) represent the pose of the object w.r.t the Kinect and \(T_h\),the position of the HoloLens w.r.t the anchor. Figure 8 shows the relation of all relevant transformations.
4.4 Training the Random Forest: Top-down data split

Training each tree of a random forest requires one to split the training dataset per node. A typical training procedure follows a top-down approach: starting at the top node, the dataset is split at each tree-level according to the input data. Typically, a brute-force algorithm combines all data samples and finds the splitting pivot point which yields the minimum error; thus, the best outcome.

Since the outcome for this effort is a tensor, a new similarity operator was required. To determine the similarity between two correction matrices, we calculate the angle between the orientations(rotation matrices) of the correction matrices and the distance between the translations of the correction matrices:

\[ Q = Q_1 Q_2^{-1}, \quad angle_{12} = 2 \cos^{-1}(Q \cdot w) \text{ and} \]
\[ dist_{12} = ||t_1 - t_2|| \]
where $Q_1$, $Q_2$, are the rotation matrices represented as quaternions, and $t_1$, $t_2$, the translations from the two correction matrices.

To determine the splitting pivot point, we examined two different approaches:

**Similarities summation:** Here, we add all the angles and the distances calculated between each of the correction matrices in a group on the right and the left side of the pivot point and, the point with the minimum summation value is considered as the split. The main idea behind this method is that the closest matrices, when grouped, will yield the minimum angle differences and distances. In other words, the best split terminates with the smallest value.

\[
\text{divide at pivot } p \text{ where } \sum_{i=0}^{p-2} \sum_{j=i+1}^{p-1} (\text{angle}_{ij} + \text{dist}_{ij}) + \sum_{i=p}^{n-2} \sum_{j=i+1}^{n-1} (\text{angle}_{ij} + \text{dist}_{ij}) \text{ is minimum}
\]

**Clustering similarities:** Here, we count the number of similar matrices in each group by counting the number of angle and distance pairs that lie below a specific threshold on the right and left side of the pivot. The point that yields maximum count is considered for the split. The main idea behind this method is that only those matrices are considered closest that lie within a threshold angle or distance.

**4.5 Calculation of the output matrices**

The objective of this step is to determine an individual output matrix. The leaf node of the random forest usually contains multiple correction matrices. Additionally, multiple trees return
additional correction matrices. Thus, one has to approximate an individual solution from all options. We examined two approaches:

**Nearest solution**: We return the correction matrix corresponding to the output label whose input position is the nearest to the test input position of the user. The main idea behind this approach is that the correction matrix corresponding to the nearest user position will be similar to the correction matrix of the current position.

**Approximation**: We calculated the mean of the correction matrices corresponding to the output labels as the output for the given user position.

$$mean\ correction\ matrix = \begin{bmatrix} \frac{1}{n} \sum X & \frac{1}{n} \sum Y & \frac{1}{n} \sum Z & \frac{1}{n} \sum T \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $X$, $Y$, $Z$ reperesent the rotation of each axis with respect to the fixed coordinate system correction matrices and $T$ is the translation vector.

For both approaches, duplicated labels were consolidated in advance to the process. The final result is one correction matrix.
CHAPTER 5. EXPERIMENTS AND RESULTS

The following sections describe the experiments and the results. The next section starts with an overview, explaining the individual tests executed. The remainder of this sections explains the measurement method, and discusses the results.

5.1 Overview

The entire study approach is experimental. In general, we compared the visual reprojection results that a single SPAAM calibration yields in comparison to the proposed approach. Table 1 shows all experiments executed. It shows experiments conducted with a single SPAAM procedure conducted from four different positions. Then the experiments conducted using Random Forest with different combinations of splitting training data, the calculation of the final output, number of trees used in random forest and the maximum depth for each tree. We also tested the impact of the threshold angle and distance while splitting the training dataset during clustering similarities procedure.

Table 1: List of all experiments conducted

<table>
<thead>
<tr>
<th>label</th>
<th>Procedure Name</th>
<th>Splitting Dataset Procedure</th>
<th>Output Calculation Procedure</th>
<th>Threshold angle</th>
<th>Threshold distance</th>
<th>Number of trees</th>
<th>Max depth of trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS1</td>
<td>Single SPAAM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SS2</td>
<td>Single SPAAM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>SS3</td>
<td>Single SPAAM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>SS4</td>
<td>Single SPAAM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>RF1</td>
<td>Random Forest</td>
<td>Similarity Summation</td>
<td>Nearest</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>RF2</td>
<td>Random Forest</td>
<td>Similarity Summation</td>
<td>Nearest</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>RF3</td>
<td>Random Forest</td>
<td>Similarity Summation</td>
<td>Nearest</td>
<td>-</td>
<td>-</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>
We did multiple experiments to verify statistical robustness as well as to identify outliers.

Note that two datasets were prepared for these experiments. The first dataset is a training dataset. It was used to, first, train the random forest, and secondly, to calculate the single SPAAM calibration matrices. It comprises of 40 data samples. The second dataset is a validation dataset. It was used to compare the single SPAAM calibration results with the random forest results. Thus, no validation dataset was used for training or to obtain a SPAAM calibration matrix.

Each tuple in training dataset contains the user position with respect to the object, a label to uniquely identify this tuple, a quaternion representing the rotation of the correction matrix.
(used only for mathematical calculations), translation vector of the correction matrix followed by the rotation matrix.

Each tuple in the evaluation dataset contains the user position with respect to the object and the name of the image of the object taken from this position (used for projecting pixel points for validating the correction matrices). The random forest returns the correction matrix for all the positions as an output.

Then we use the predicted correction matrices and intrinsic matrix to project pixel points of a corner onto each image to validate the estimated position of the corner using the correction matrix from the random forest.

The ground truth location of each reference point in the evaluation dataset was identified manually using ImageJ.

5.2 Measurement method

Results were obtained from the pixel-wise projection of a reference model on the expected location. We project the points of this reference model into the 2D space using the intrinsic matrix, and determined the error between the real corner points and the projected corner points.

For this effort, we used a rectangular bounding box as a reference model and its projected corner points as accuracy marks. The points and their ground truth location is part of the validation dataset. The delta between the expected location and the ground truth location in pixel is the error. Figure 9 shows an example for one reference point (upper right corner). The
red label indicates the expected position of the reference point. The red cross-marks are the projected points using single SPAAM matrices and the green cross-mark is the projected point using random forest.

Figure 9. Projected points using SPAAM and random forest

5.3 Single Point Active Alignment Method Results

This section presents the results yielded by a single SPAAM matrix calibration. The delta between the real corners and the estimated corners are plotted and the root mean square error (rmse) for the all the points is calculated. Figure 10 shows the results as xy-plot for different single SPAAM matrices. The plot shows the error between the projected point and the ground truth data. The x-axis shows the error into the x-direction, and the y-axis the error into the y-direction. Also, the overall delta between the ground truth data and the projected data is given as root-mean-squared (RMS) error, standard error and mean error.
Figure 10. Delta between the position of real corners and the estimated corners for single SPAAM calibration

The results show that a single SPAAM always yields large distances between the real and the estimated points as the differences between the real and estimated points in the plot are scattered and are far away from zero.

5.4 Best Split: Similarity Summation, Output: Nearest solution Results

This section presents the results yielded by the random forest using the similarity summation for splitting the data during training and the nearest solution for returning the output. The results are obtained by changing the values for the number of trees used in random
forest and the maximum depth of each tree. The delta between the real corners and the estimated corners are plotted and root mean square error (rmse) for the all the points is calculated. Figure 11 shows the results as xy-plots. The parameters altered were the number of trees used in random forest (trees = 3 and 10) and the maximum depth of each tree (max. depth = 3 and 4). The plot shows the error between the projected point and the ground truth data. The x-axis shows the error into the x-direction, and the y-axis the error into the y-direction. Also, the overall delta between the ground truth data and the projected data is given as root-mean-squared (RMS) error, standard error and mean error.
Figure 11. Delta between the position of real corners and the estimated corners using the similarity summation and the nearest solution approach.

The results show that the distance between the estimated points and real points is smaller in comparison to the results of single SPAAM. All the errors between the real and estimated points in the plot are close to zero. Also, the RMSE value is smaller in comparison to the single SPAAM results.

5.5 Best Split: Similarity Summation, Output: Approximation Results

This section presents the results yielded by the random forest using the similarity summation for splitting the data during training, and approximation for returning the output matrix. The results are obtained by changing the number of trees and the maximum depth of each tree. The delta between the real corners and the estimated corners are plotted and the root mean square error (rmse) for the all the points is calculated. Figure 12 shows the results as xy-plots using the random forest approach with different number of trees (trees = 3 and 10), and a varying maximum depth for each tree (max. depth = 3 or 10). The plot shows the error between the projected point and the ground truth data. The x-axis shows the error into the x-direction, and the y-axis the error into the y-direction. Also, the overall delta between the ground truth data and the projected data is given as root-mean-squared (RMS) error, standard error and mean error.
The results show that the distance between the estimated points and real points has reduced in comparison to SPAAM and the earlier results. The plotted errors are much closer to zero and the RMSE value is also smaller in comparison to the previous approaches.

5.6 Best Split: Clustering Similarities, Output: Nearest solution

This section presents the results yielded from the random forest using clustering similarities for splitting the data during training and the nearest solution for returning the output. The results are obtained by changing the values for the parameters threshold angle, distance,
number of trees, and the maximum tree depth. The delta between the real corners and the estimated corners are plotted and root mean square error (rmse) for the all the points is calculated. Figure 13 shows the results as xy-plots. The experiment changed the number of trees per random forest, the maximum depth of each tree (max. depth = 3 or 4), and the distance values. Figure 14 shows the results for a different threshold angle and distance value. The plot shows the error between the projected point and the ground truth data. The x-axis shows the error into the x-direction, and the y-axis the error into the y-direction. Also, the overall delta between the ground truth data and the projected data is given as root-mean-squared (RMS) error, standard error and mean error.
Figure 13. Delta between the position of real corners and the estimated corners using clustering similarities and the nearest solution approach (threshold angle = $12^0$ and threshold distance = 10 cm).
Figure 14. Delta between the position of real corners and the estimated corners using clustering similarities and the nearest solution approach (threshold angle = 17° and threshold distance = 10 cm).

The results show that different threshold angle and distance values do not affect the output significantly since all the plots look similar; the RMSE values are also similar. However, this outcome can be attributed to the used nearest solution approach since the RMSE results here are similar to results described in Section 5.4, which reports the results obtained from the combination of the similarity summation and the nearest solution approach.
5.7 Best Split: Clustering Similarities, Output: Approximation

This section presents the results yielded by random forest using clustering similarities for splitting the data during training and approximation for returning the output. The results are obtained by changing the values for the parameters threshold angle, distance, number of trees, and the maximum tree depth. The delta between the real corners and the estimated corners are plotted and root mean square error (rmse) for the all the points is calculated. Figure 15 shows the results as xy-plots. The experiment changed the number of trees per random forest, the maximum depth of each tree (max. depth = 3 or 4), and the distance values. The plot shows the error between the projected point and the ground truth data. Figure 16 shows the results for a different threshold angle and distance value. The x-axis shows the error into the x-direction, and the y-axis the error into the y-direction. Also, the overall delta between the ground truth data and the projected data is given as root-mean-squared (RMS) error, standard error and mean error.
Figure 15. Delta between the position of real corners and the estimated corners for clustering similarities and approximation approach (threshold angle = 12° and threshold distance = 10 cm)
The results show that the approximation approach for calculating the output matrix yields better results than using the nearest solution approach as the RMSE values are better when approximation approach is used to calculate the resultant matrix compared to nearest solution approach. The threshold values did not show significant impact, however, that can be because the difference between the threshold values is low.
CHAPTER 6. CONCLUSION AND FUTURE WORK

The quantitative and qualitative results of the experiments demonstrate a significant advantage of a random forest in comparison to a single SPAAM calibration. Consequently, the results prove that, first, dynamic calibration yields a better visual experience when using a head mounted display, secondly, that a random forest can facilitate this calibration. Although the results are limited to one test object, the utilized equipment (Kinect) comes along with unusual high point cloud errors. The approach can improve the visual registration by a factor of two, even under these conditions. Thus, we can conclude that the suggested method improves the spatial registration and can effectively reduce the HoloLens tracking error.

Nonetheless, the limited number of tests, the static dataset, and the lack of a user study limit the significance of the results. As a next step, additional tests under dynamic tracking conditions are apt to better assess the performance of the random forest. Also, the primary results are quantitative: AR is a human-computer-interaction technology. Thus, the user's perception of the improvement would reveal its impact on applications in areas such as assembly support and others. Finally, although this study investigated several parameters of a random forest experimentally, its scope did not permit a comprehensive study. Thus, additional parameters studies may allow one to reduce the number of training samples, and to expedite the set-up procedure.
References:


