Improving Camera Pose Estimation for Indoor Marker-less Augmented Reality

Sud Sudirman and Abdennour El-Rhalibi
School of Computing and Mathematical Sciences
Liverpool John Moores University, UK
{s.sudirman, a.elrhalibi}@ljmu.ac.uk

***Abstract*—Vision-based Augmented Reality (AR) techniques rely heavily on Computer Vision algorithms for most of their tasks. It is understood that these algorithms require numerous parameters to function and their values can affect their outputs. Oftentimes the results vary greatly when different parameters were used and as a result, the performance of the AR technique that utilises them varies accordingly as well. This paper aims at improving the performance of AR techniques by employing a novel algorithm to adjust the parameters automatically during runtime. More specifically, the proposed technique works on improving the camera pose estimation stage, arguably one of the most crucial stages, of such AR systems.**

***Keywords*—Augmented Reality, Feature Detector, Camera Pose Estimation, Computer Vision.**

I. Introduction

Augmented Reality, or AR for short, is a technology that allows superimposing of virtual contents or information onto real world images or videos. Augmented Reality, and its sibling Virtual Reality (VR), are subsets of different realities in the Reality-Virtuality continuum suggested in [1]. Whereas the goal of Virtual Reality is to create entire virtual worlds that a user can interact with, in an AR system virtual objects are displayed in such a way that they appear to coexist with the real ones.

AR has many possible applications in a wide variety of fields including education, public health, construction, manufacturing and entertainment [2]. It has attracted the attention of researchers and practitioners since it was coined in 1992 by Tom Caudwell [3]. The popularity of AR has been growing in the past 20 years thanks to a number of new techniques and open source implementations such as ARToolKit [4] which uses 3D graphics overlay onto a video feed by using object detection and tracking algorithms. The use of AR technology in computer entertainment was pioneered with the release of ARQuake, an interactive outdoor AR collaboration system [5] in 2002. ARQuake is an Augmented Reality version of the popular First Person Shooter game *Quake*. It uses a head-mounted display, mobile computer, head tracker and a GPS system to provide inputs to control the game.

With the growing popularity of smartphones, AR technology has reached a new height. The presence of an on-board camera and the relatively powerful processor that smartphones possess allow videos to be captured and processed effortlessly. The release of Wikitude [6] and the porting of ARToolkit to Adobe Flash have further accelerated the growth of AR popularity using smartphones.

The AR technology has somewhat reached a stage where many commercial applications and gadgets use it as their main selling point. For example Layar [7], a mobile phone app, uses AR technology to embed commercial information onto videos captured by mobile phones. It works by using a combination of the mobile phone's camera, compass and GPS data to identify the user's location and field of view, retrieve data based on those geographical coordinates, and overlay that data over the camera view. Microsoft's first attempt at Augmented Reality, dubbed Microsoft HoloLens [8], has recently attracted a large number of interests when it was showcased in Electronic Entertainment Expo (E3) 2015 in Los Angeles.

AR techniques can be classified into two major categories namely vision-based AR and location-based AR. Location-based AR uses the ability of a particular device to locate its position in the world, e.g., by means of GPS, and then retrieve relevant information to that location. This information is then superimposed into the output of their device's camera to allow a more natural data presentation compared to just using a map alone. This type of AR can be used to help the users' everyday tasks such as finding their way around a city.

On the other hand, vision-based AR relies mostly on processing the data that is extracted from the images or video frames taken by the device. This type of AR involves a number of techniques that lend heavily from computer vision (CV) to the extent where research progress in AR depends on the progress of the latter. Research progress in computer vision is moving rapidly with many scholars believe that major and ground-breaking progresses are still in the development in a foreseeable future. The same argument can be said also regarding the AR technology with new techniques and algorithms proposed regularly in literatures.

The objective of this paper is to look at a specific aspect of AR process that uses computer vision algorithms, namely feature detection and tracking, and suggest a method to improve the subsequent process to produce a better camera pose estimate. The paper will start by reviewing the current and state-of-the-art AR techniques in the literature and identify a research problem in marker-less AR. This problem is analysed and a hypothesis is made in section 3. This paper's solution to the problem is presented in section 4. Experiment results and analysis are given in section 5 together with the proof of the hypothesis.

II. Literature Review and Analysis

Vision-based AR process involves a number of techniques that borrows heavily from computer vision and image processing. The overall process in this type of AR is illustrated in Figure 1.

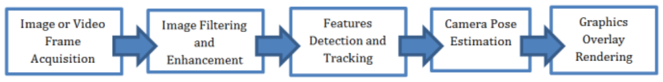
Image or Video Frame Acquisition
Image Filtering and Enhancement
Features Detection and Tracking
Camera Pose Estimation
Graphics Overlay Rendering

Figure 1. Steps in Vision-based Augmented Reality

To start with, an image or video frame is captured from the camera device. If the AR is performed by a smartphone, it may come with a software library that allows this step to be done relatively easily. The resulting image may need pre-processing to clean up some noise, to enhance the features of interest, or to remove some distortions that are prevalent in camera devices. This is then followed by the application of image features detection and tracking algorithms. This is a very important stage in AR as the result of this step determines the accuracy and speed of the entire process. It is argued that good feature detection and tracking algorithms which detect objects reliably and quickly would make a successful AR program [9]. The result of this step is then fed through an algorithm that determines the pose of the camera with respect to the scenery. This will give the view-projection matrix that can be used to render virtual information on the image as a 3D system.

Depending on what and how image features are detected and tracked, vision-based AR can be further classified into two categories namely *marker-based AR* and *marker-less AR*.

Marker-based AR utilises markers that are placed in the scene and within the field-of-view of the camera to help guide the camera pose estimation process. The markers are often referred to as *fiducial markers* because their position and orientation relative to the scenery are fixed. The markers are often planar markers and typically have strong (high contrast and defined orientation) features such as long edges or corners between black and white regions.

AR techniques that belong to this category put strong emphasis on the marker design. The most popular type of marker design is square marker because the square feature would allow a precise localisation of the markers through its four corner points. The inner region of the marker can contain either binary code or an arbitrary binary pattern that is useful for identification and orientation estimation. Included in this category is ARToolkit technique [4]. The markers consist of wide black border with an arbitrary image in the centre region. The image is stored in a database which is then, by means of pattern matching technique, is detected and localised. The approach, while very popular in the academic community, has a relatively high false positive detection error. In addition, if multiple markers were used the technique would exhibit a rather high inter-marker confusion rate. The technique was further improved, and rebranded as ARToolkitPlus [10], by incorporating automatic setting of global threshold values used to detect the markers. The technique also utilises an error detection and correction technique to improve the robustness of the detection process over its predecessor.

Another notable technique in this marker-based AR category is Aruco [11]. The technique uses a dictionary of markers that are automatically generated using an algorithm that maximises inter-marker classification difference. Instead of using the markers from this dictionary individually, the technique uses multiple markers in a rectangular, checker-board-style panel. This would make it possible for camera pose estimation to be carried out even when partial occlusion of the panel occurs.

In marker-based AR algorithms, edges are the most popular type of image features to detect [9]–[11]. This is because the markers have typically prominent and strong regions with high contrast boundaries which respond well to edge detectors. The resulting output is then used to identify image segments which in turn can be matched with the marker database. Since the markers in the database are stored in a normal frontal view, comparing them with image segments in the current frame would require removal of perspective-projection transformation. This is done by calculating the homography matrix for each possible pair. The result of this matching process would be a set of number that corresponds to the confidence of an image segment belonging to a specific marker. By taking the maximum value of each pairing and applying a threshold to it, the algorithm finds the best matching marker for each segment. The homography matrices of those best match pairs are then used to estimate the overall camera pose.

Marker-based techniques belonging to this category is by far the most popular AR technique, as detection and tracking of fiducial objects are relatively easy to perform and allows high speed and precise AR. These techniques, however, have severe limitation basically due to its strict requirement of having to use the fiducial markers. In practice, it may not always be possible to attach these markers onto the scenery given the locality and distance of the object from camera. Furthermore, even when the markers can be placed, they have to be within view and the sensing range of the camera.

The alternative to using fiducial markers would be to perform the camera pose estimation via detection and tracking of features from naturally occurring objects. The features could be corners, pattern or textures. AR technologies that detect and track these feature belong to the second category in AR classification, namely marker-less AR.

Marker-less AR differs from marker-based AR because it does not rely on the use of artificial markers to help salient features detection in the scene. Instead, marker-less AR works by detecting features that are readily available from the natural objects in the scene and attempt to create a *model* or a *map* from the scenery in order to represent the world as it is viewed by the camera. The model can be described as a hypothesis of the layout of objects in the scenery in relation to the camera view. This can then be matched to the real model, by means of registration technique, to get the camera pose. If there is no such real model of the scene exists, then a technique is needed to approximate the initial model and subsequently update and improve the initial model by tracking image features in scenes viewed from different camera poses. This problem is often referred to as *Simultaneous Localization and Mapping* or

SLAM. Techniques which attempt to solve this problem are grouped into a class of techniques known, in the AR context, as *Extensible Tracking*. This type of technique has garnered much attention of many researchers, in particular those in mobile robotics area.

SLAM is a well-known and well-researched problem in autonomous robotics. The dominant approach to solve this problem is introduced in a seminal paper [12] by Smith et al. The paper proposed a concept of a stochastic map, which is a representative for spatial information of the viewed scenery and a method of building and incrementally update it as new information is obtained. The approach has since been revisited and improved a number of times with examples including FastSLAM [13], MonoSLAM [14] and RT-SLAM [15].

A notable extension to the extensible tracking approach in SLAM was proposed in [16]. In this approach, which is known as Parallel Tracking and Mapping or PTAM, the tracking and mapping processes were split and handled as parallel threads to allow the use of computationally expensive batch optimisation technique that makes the approach more suitable for real-time operation. The technique was reported to have produced very good results in implementing a marker-less AR. However, due to the constraints in the algorithm used, the technique is only suitable mostly for indoor or small workspace environment.

A different approach to marker-less AR by means of image registration technique is adopted by a number of researchers [17]–[19]. Marker-less AR with this approach has some underlying similarity with marker-based AR. However, instead of using pre-defined markers, a snapshot of the scenery is used as a reference image. This reference image normally contains a flat region on which the 2D x-y plane of the virtual axis can be based on. Image features in this reference image and in subsequent frames are then detected, and an image registration technique is used to match and track these features. The technique proposed in this paper will focus on issues affecting this approach of marker-less AR to improve its performance.

III. Marker-less AR and Problem definition

A. Problem definition

As can be seen from the review of the current state-of-the-art AR in the previous section, marker-less AR provides the most potential and better use of AR in day-to-day life. On the other hand, the technique is less reliable and more prone to external factors, hence the need to further improve the technique's reliability and robustness is always present.

As is also the case with the rest of vision-based AR techniques, marker-less AR techniques rely heavily on the computer vision algorithms used to pre-process, detect and track features, and estimate the camera's pose. Computer vision algorithms often use a number of parameters when producing their outputs, with typical examples include threshold values, width of Gaussian function, and range of resolutions etc. The result of the feature detection process depends largely on the values of these parameters. Furthermore, a good set of parameter values is variable as it often depends on many real-time and run-time conditions such as lighting, surface textures, and even poses. As such there is a need to estimate these best or optimal values before the camera pose estimation process can be done. With that argument in mind, we are going to test the following supposition:

Hypothesis 1.

The reliability and robustness of an AR technique to external real-time and run-time conditions can be improved by using an algorithm to dynamically set the parameters used by the employed computer vision algorithms.

To put that argument into perspective and to help us formulate a solution, we will look deeper into what these techniques are. We will start by looking at the different feature detection algorithms, followed by feature tracking algorithms and camera pose estimation algorithms.

B. Feature Detection

Image features are essentially a set of important information that can be used to describe the image or a subset of the image. They are immensely important in the area of computer vision because they can be used to mark interest areas in the image and can be used to compare one image to another. There are two types of image features namely global and local features.

Local features are computed at different locations in the image using only small support area of around the location point. As such, local features describe only the image in the context of that small subset and nothing else. That means even when the other parts of the image undergo changes, as long as the support area remains the same, local features would more likely not be affected. This is one of the strong points of local features over global features because they are robust to occlusion, or changes in camera pose, etc. Examples of local features are corners, edges, and texture descriptors.

Global features, on the other hand, are derived from the entire image. They have the ability to generalise the entire image into one single feature vector. An example of global features is image code. Image code is a compressed form of the image using an appropriate coding technique that preserves the high level information of the image contents. It is also not uncommon to have a global feature which is a collection of many local features such as shape descriptors, contours descriptors, texture descriptors, etc.

Comparatively, local features are more widely used than global feature in AR application. Many marker-based AR uses edges whereas many marker-less AR prefers use corners. One of the first corner detectors is Harris detector [20] in 1988 and since there are numerous more developed including GFTT [21], FAST [22], ORB [23], MSER [24], SIFT [25] and SURF [26]. Every feature detection algorithm uses parameters to adjust its output with the most common ones are threshold values, support area width, and Gaussian/Laplacian function width.

C. Feature Matching and Tracking

Tracking of features is performed by means of detecting the features in one frame (often referred to as reference frame) and the subsequent frames and perform feature matching between them. The best matched pairs of features from both the reference frame and a subsequent frame are likely to correspond to the same object in both frames hence feature matching process is essential in identifying parts of an image which correspond to those in a reference image. This process is considered as one of the most computationally expensive part of many computer vision algorithms. This is because it involves searches for closest matches between pairs of high-dimensional vectors.

When comparing two sets of features, the simplest solution is to use a *brute force* approach that compares every feature in one set to every feature in the other and keeping track of the "best so far" match. This approach, sometimes also referred to in literature as the *naive* approach, results in an $O(N^2)$ computational complexity where N is the number of feature in each frame (assuming the same number in each). There are a number of proposed algorithms that aim at improving the computational efficiency of feature matching process. One of the most popular and more traditional technique is kd-tree [27]. This technique uses exact nearest neighbour search and works very well for low dimensional data but quickly loses its effectiveness as dimensionality increases. The popularity of the kd-tree technique has seen a number of proposals to further improve the algorithm including [28], [29]. A more recent matching technique that gains popularity amongst computer vision researchers is proposed by Muja and Lowe in [30]. It is based on nearest neighbour matching algorithm using priority search and adaptive partitioning of the kd-tree. Feature matching algorithms also use parameters to adjust their output with the most common ones are threshold values, search width, initial partition and neighbour size.

D. Camera Pose Estimation

There are two major approaches to camera pose estimation techniques, namely *Relative Orientation* and *Planar Homography*. Relative Orientation is a technique that calculates the position and orientation of a camera relative to another from correspondences among five or more ray pairs. A ray pair is defined as the vectors that originate from a fixed and visible point in the scenery to the centre of the camera positions. If the sceneries were taken by the same camera which was moved and re-orientated then the technique can be used to calculate the transform matrix of the camera movement. It is an active research area as the problem is a very important one in photogrammetry [31]–[35]

While Relative Orientation approach generally takes any feature points in the image without many constraints over their placement, the algorithm is very complex and is very sensitive to feature detection errors. While Planar Homography works in a similar fashion as Relative Orientation approach, it requires the features to be tracked to be on the same 2D plane. Planar Homography is defined as a projective mapping from one 2D plane to another (typically image plane). This constraint simplifies the problem and allows more straightforward calculations to be made. Details on this topic is outside the scope of this paper but interested readers are advised to look up the relevant textbook [36] and survey paper [37].

At its core, camera pose estimation is a parameter estimation problem. The algorithm would have to find a model or a set of parameters that best fits the transform from the reference pose to the new pose. To do this, an efficient matching technique such as RANSAC or *RANdom SAmple Consensus* [38] can be used. This technique works by selecting random number of features to match initially to fit an initial matching parametric model. When the parameters of the model had been calculated the algorithm then works out how many features from the entire set would with this model within a pre-defined tolerance. If the result exceeds a pre-defined threshold, then that model is used, otherwise the process is repeated for a set number of time. This technique produces better matching result when there are outliers in the feature sets compare to a naïve or brute force approach.

E. Summary

As can be seen from the analysis given in the previous sub section, there are many parameters used that can affect the overall outcome of an AR system. There is a need to study how these parameters can affect the final outcome. This further justifies the necessity to test and prove the hypothesis stated earlier. The design of the solution to test this hypothesis is described in the next section.

IV. Solution Design

A. Solution Design

The solution to prove the hypothesis has the following requirement. First, it has to employ a suitable feature detection algorithm that allows direct control over the number of detected features. This is important since a set of features that contains too many outliers could worsen the process of feature matching and consequently the camera pose estimation. Having a means to control this would allow us to directly measure the quality of the AR output at different situations.

Secondly, it has to have quantifiable means to measure the confidence of the camera’s pose estimation process. A high confidence value would mean that the resulted pose has a high degree of accuracy and can be used for updating the system knowledge or model of the camera system in addition to using the pose for the next steps in AR. A low confidence means that the current estimated pose should not be used to update the model.

This has led to a constraint to be imposed on the solution which requires the scenery to be of static environment and it should have significant planar area for reference (Z=0) X-Y plane such as table top or floor space.

The computer vision algorithms to be used are as follow:

- SIFT will be used as the chosen feature detection algorithm due to its ability to directly control the number of features to be detected. In addition, SIFT features can be described very expressively to allow more natural and faster matching process that is more suitable for AR applications.
- RANSAC algorithm is to be used for feature matching and tracking due to its speed and robustness to outliers.
- Planar Homography algorithm is to be used to exploit the existence of planar region in assisting the camera pose estimation.

The solution would make use of a standard inexpensive webcam to acquire the images. The inner working of this type of camera can be described by the pinhole camera model [36]. It generally accepted that the model introduces significant distortion in the image due to cheap lenses and camera production errors. These issues however can be easily solved by calibrating the camera to find out its intrinsic parameters and use them to correct the image frame.

The solution consists of two stages. The first one is the semi-automatic parameter adjustment. This stage will search in the parameter space a value that maximise a given criteria. It requires the user to manually identify planar convex region in the reference frame by marking the region's corners. It then proceeds to the feature detection process. Once they have been detected, any features whose locations are inside the planar region are marked and their numbers are counted. The algorithm used to check if a point is inside a polygon is given in Algorithm 1.

Algorithm 1. Feature location check

```
Input: point p∈ℜ², vertices V∈ℜ².
Output: status o ∈ {0, 1}   # 0: outside, 1: inside
1.   previous ← -1
2.   current ← -1
3.   for each Vᵢ∈V
4.        j ← (i+1) % |V|    # % denotes remainder operation
5.        e ← Vⱼ - Vᵢ
6.        q ← p - Vᵢ
7.        s ← e × q          # vector cross product
8.        if s < 0
9.            current ← 0
10.       else if s > 0
11.           current ← 1
12.       else
13.           current ← -1
14.       end if
15.       if current = -1
16.           o ← 0
17.           end algorithm
18.       else if previous ≠ current
19.           o ← 0
20.           end algorithm
21.       else if previous=0
22.           previous ← current
23.       end if
24.  end for
25.  o ← 1
26.  end algorithm
```

Once they are known, a ratio between that number and the total number of features is calculated and compared to a desired ratio R_0. The general idea is to have as many features inside the marked region as possible while keeping those outside as few as possible. Features that lie outside the region would likely serve as outliers in the feature matching and planar homography processes and reduce the accuracy of camera pose estimate. The detailed step of this stage of the solution is illustrated by the flowchart shown in Figure 2.

When the calculated ratio exceeds the desired ratio, the parameter value and the locations of the features inside the region are recorded. These will be used in subsequent stages and to be referred to as $\mathcal{q}$ and $\mathcal{P}_0$ respectively.

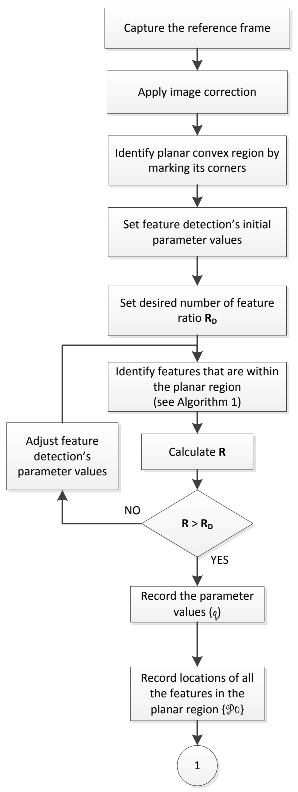
Capture the reference frame
Apply image correction
Identify planar convex region by marking its corners
Set feature detection's initial parameter values
Set desired number of feature ratio $\mathbf{R_D}$
Identify features that are within the planar region (see Algorithm 1)
Adjust feature detection's parameter values
Calculate **R**
NO
$\mathbf{R > R_D}$
YES
Record the parameter values ($\mathcal{q}$)
Record locations of all the features in the planar region {$\mathcal{P}0$}
1

Figure 2. Stage 1 - steps to obtain the feature detection parameter values.

In stage 2, the algorithm will use subsequent frames to build a camera pose model $\mathcal{H}_0$ which is essentially a view-projection matrix. Its value will be updated as more knowledge of the scenery is known. The stage begins by setting $\mathcal{H}_0$ to identity – which signifies the absence of changes in camera pose relative to the reference image.

In the next step, a new frame or image of the scenery is taken and feature detection algorithm is applied to this image using the same parameter value $\mathcal{q}$ obtained in the previous stage. The locations of the resulting features are stored and would be referred to as $\mathcal{P}_1$. Subsequently, feature matching and camera pose estimation processes are performed to produce a hypothesis $\mathcal{H}$ of the new camera pose. This is then combined with the previous model to calculate the cumulative model $\mathcal{H}_n$.

To get a measure of confidence on the accuracy of this hypothesis, a distance measure $\mathcal{D}$ is calculated between the real feature location $\mathcal{P}_1$ and the transform of the reference feature location, denoted as $\mathcal{P}_0^{\mathrm{T}}$ using the cumulative hypothesis $\mathcal{H}_n$.

$$\mathcal{D} = \frac{\sum_n (|dx_n| + |dy_n|)}{n} \tag{1}$$

Where dx_n and dy_n are the normalised horizontal and vertical difference respectively and are calculated as:

$$dx_n = \frac{\mathcal{P}0^T{}_n.x - \mathcal{P}1_n.x}{w}$$

$$dy_n = \frac{\mathcal{P}0^T{}_n.y - \mathcal{P}1_n.y}{h}$$

$$\mathcal{P}0^T = \mathcal{P}0.\mathcal{H}_n$$

In the above equations, *w* and *h* denote the width and height of the images respectively. Eq. 1 which is used to calculate $\mathcal{D}$, uses L1 or Manhattan distance to avoid the more expensive calculation associated with the higher order distance such as Euclidian. The value of $\mathcal{D}$ is a measure of confidence of the calculated camera pose hypothesis $\mathcal{H}$. A small $\mathcal{D}$ value ($\rightarrow 0$) means that transform points and actual points match well hence high confidence in the camera pose estimation, whereas a large $\mathcal{D}$ value ($\rightarrow 1$) would mean that the transform points and actual points match poorly, hence low confidence in the camera pose estimation. The detailed step of the solution is described in the flowchart shown in Figure 2.

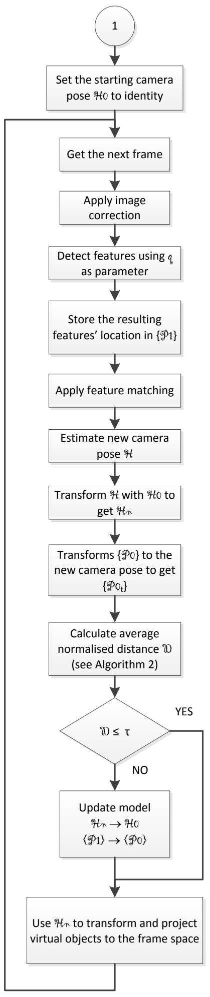
1
Set the starting camera pose $\mathcal{H}0$ to identity
Get the next frame
Apply image correction
Detect features using q as parameter
Store the resulting features' location in $\{\mathcal{P}1\}$
Apply feature matching
Estimate new camera pose $\mathcal{H}$
Transform $\mathcal{H}$ with $\mathcal{H}0$ to get $\mathcal{H}_n$
Transforms $\{\mathcal{P}0\}$ to the new camera pose to get $\{\mathcal{P}0_t\}$
Calculate average normalised distance $\mathcal{D}$ (see Algorithm 2)
$\mathcal{D} \leq \tau$
YES
NO
Update model $\mathcal{H}_n \rightarrow \mathcal{H}0$ $\{\mathcal{P}1\} \rightarrow \{\mathcal{P}0\}$
Use $\mathcal{H}_n$ to transform and project virtual objects to the frame space

Figure 3. Stage 2 - Steps to updating camera pose model

B. *Experiment setup and evaluation methodology*

The experiment uses ten unique sceneries with each consisting up to twenty different frames taken from different camera poses and views and taken at three varying lighting conditions. The algorithm is implemented using OpenCV on Windows platform.

To evaluate the results visual comparisons between with and without applying the algorithm will be shown. The initial values for the feature detection parameter are chosen manually to get the best pose estimation result in one lighting condition. The test then varies the lighting condition and run the first experiment without adjusting the parameter values using the proposed algorithm. The experiment is then repeated under the same lighting condition using the proposed algorithm.

V. Experiment Results and Conclusion

The experiment yields results that show the proposed algorithm can produce more stable camera pose estimation compared to when it is not applied. Using this algorithm, the planar region is detected correctly in all the frames in the ten sceneries taken in all three lighting conditions compared to just on average of only about 58% without. The average value of $\mathcal{D}$ is 0.27 and 0.002 for without and with the algorithm applied respectively. Visual evidence of this can be seen on the sample images shown in Figure 4 to Figure 9.

The increase in the average correct pose estimation rate when the algorithm is applied proves the previous hypothesis that the reliability and robustness of an AR technique to external real-time and run-time conditions can be improved by using an algorithm to dynamically set the parameters used by the employed computer vision algorithms.

References

[1] P. Milgram and F. Kishino, "Taxonomy of mixed reality visual displays," *IEICE Trans. Inf. Syst.*, vol. E77-D, no. 12, pp. 1321–1329, 1994.

[2] D. Van Krevelen and R. Poelman, "A Survey of augmented reality technologies," *Int. J. Virtual Real.*, vol. 9, no. 2, p. 1, 2010.

[3] T. P. Caudell and D. W. Mizell, "Augmented reality: an application of heads-up display technology to manual manufacturing processes," in *Proceedings of the 25th Hawaii International Conference on System Sciences*, 1992, vol. ii, pp. 659–669.

[4] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *Proceeding of the 2nd IEEE and ACM International Workshop on Augmented Reality*, 1999, pp. 85–94.

[5] W. Piekarski and B. H. Thomas, "ARQuake: The outdoor augmented reality gaming system," *Commun. ACM*, vol. 1, no. 45, pp. 36–38, 2002.

[6] Wikitude GmbH., "Wikitude," 2015. [Online]. Available: http://www.wikitude.com/. [Accessed: 25-Jun-2015].

[7] Blippar, "Layar," 2015. [Online]. Available: https://www.layar.com/. [Accessed: 25-Jun-2015].

[8] Microsoft, "Microsoft HoloLens," 2015. [Online]. Available: https://www.microsoft.com/microsoft-hololens/en-us. [Accessed: 25-Jun-2015].

[9] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Real-time detection and tracking for augmented reality on mobile phones," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 3, pp. 355–368, 2010.

[10] D. Wagner and D. Schmalstieg, "ARToolKitPlus for pose tracking on mobile devices," in *Proceedings of 12th Computer Vision Winter Workshop CVWW07*, 2007, pp. 139–146.

[11] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognit.*, vol. 47, no. 6, pp. 2280–2292, 2014.

[12] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous robot vehicles*, vol. 4, Springer New York, 1990, pp. 167–193.

[13] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "Fastslam: An efficient solution to the simultaneous localization

and mapping problem with unknown data association," *J. Mach. Learn. Res.*, vol. 4, no. 3, pp. 380–407, 2004.

[14] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, 2007.

[15] B. Williams, G. Klein, and I. Reid, "Real-time SLAM relocalisation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2007, pp. 1–8.

[16] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceeding of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234.

[17] H. Li, M. Qi, and Y. Wu, "A real-time registration method of augmented reality based on SURF and optical flow," *J. Theor. Appl. Inf. Technol.*, vol. 42, no. 2, pp. 281–286, 2012.

[18] Y. Uematsu and H. Saito, "Vision-based registration for augmented reality with integration of arbitrary multiple planes," in *Image Analysis and Processing - ICIAP 2005*, Springer, 2005, pp. 155–162.

[19] T. Guan and C. Wang, "Registration based on scene recognition and natural features tracking techniques for wide-area augmented reality systems," *IEEE Trans. Multimed.*, vol. 11, no. 8, pp. 1393–1406, 2009.

[20] C. Harris and M. Stephens, "A combined corner and edge detector," in *Procedings of the Alvey Vision Conference*, 1988, pp. 50–55.

[21] J. Shi and C. Tomasi, "Good features to track," in *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593 – 600.

[22] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3951 LNCS, 2006, pp. 430–443.

[23] E. Rublee and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *Proceeding of IEEE International Conference on Computer Vision*, 2011, pp. 2564–2571.

[24] M. Donoser and H. Bischof, "Efficient maximally stable extremal region (MSER) tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 553–560.

[25] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[26] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008.

[27] J. H. Freidman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, 1977.

[28] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," in *Proceedings of the 5th ACM-SIAM Sympos. Discrete Algorithms*, 1994, pp. 573–582.

[29] J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 1000–1006.

[30] M. Muja, "Scalable nearest neighbour methods for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. April, pp. 1–14, 2014.

[31] B. K. P. Horn, "Relative orientation," *Int. J. Comput. Vis.*, vol. 4, no. 1, pp. 59–78, 1990.

[32] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. H. € Ollerer, "Model estimation and selection towards unconstrained real-time tracking and mapping," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 6, pp. 825–838, 2014.

[33] H. Stewénius, C. Engels, and D. Nistér, "Recent developments on direct relative orientation," in *ISPRS Journal of Photogrammetry and Remote Sensing*, Elsevier, 2006, pp. 284–294.

[34] P. Sturm and T. Bonfort, "How to Compute the Pose of an Object without a Direct View?"

[35] V. Rodehorst, M. Heinrichs, and O. Hellwich, "Evaluation of relative pose estimation methods for multi-camera setups," *Int. Arch. Photogramm. Remote Sens.*, pp. 135–140, 2008.

[36] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge University Press, 2004.

[37] A. Agarwal, C. V. Jawahar, and P. J. Narayanan, "A survey of planar homography estimation techniques," 2005.

[38] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A paradigm for model fitting with applicatlons to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381 – 395, 1981.

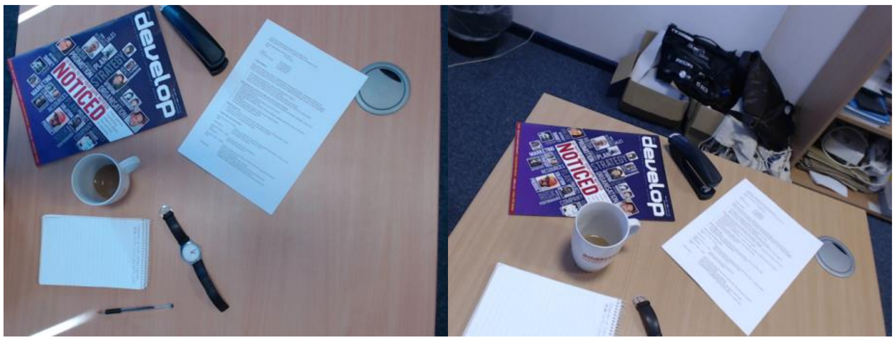

Figure 4. One of the reference images (left) and one of its corresponding scene image (right)

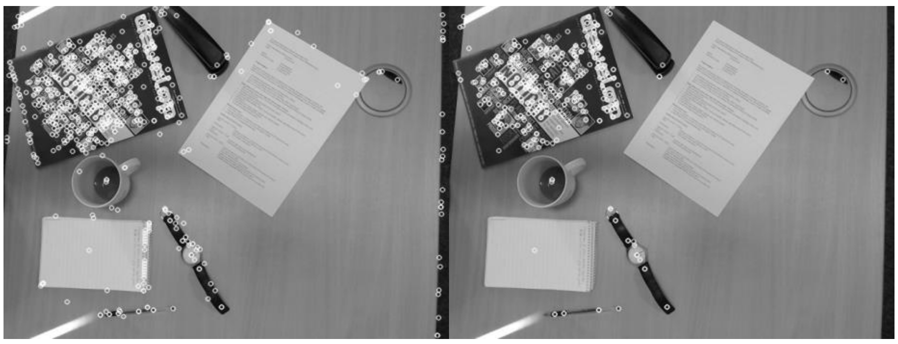

Figure 5. Detected features on the reference image without (left) and with (right) using the algorithm

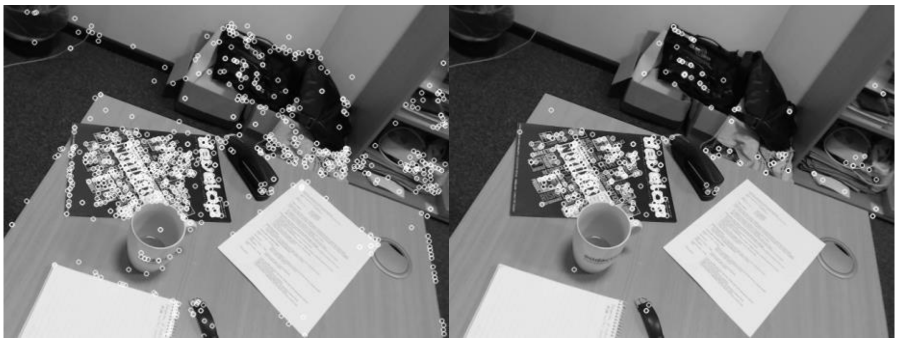
Figure 6. Detected features on the scene image without (left) and with (right) using the algorithm

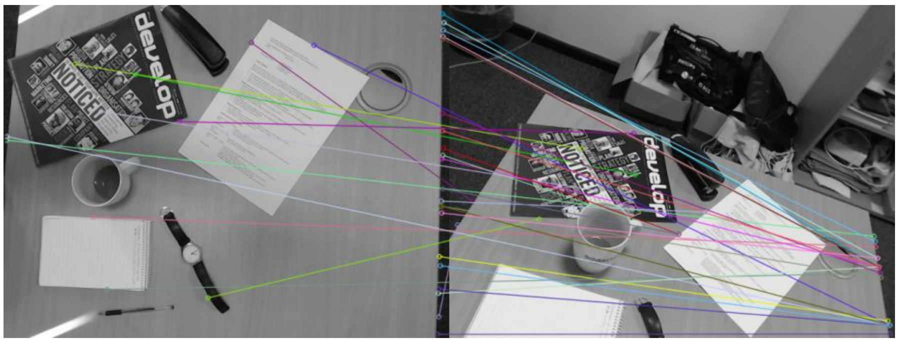
Figure 7. Matched features without the algorithm applied

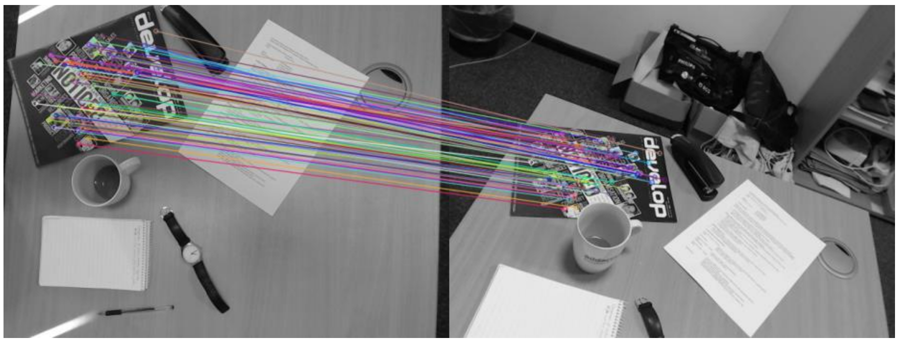
Figure 8. Matched features with the algorithm applied

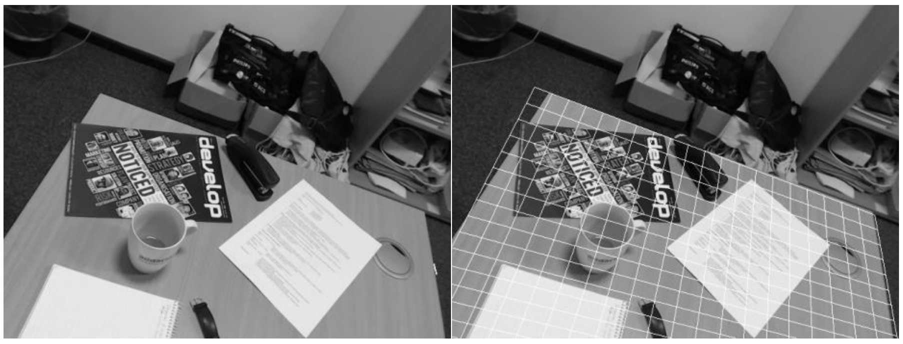
Figure 9. Approximated 2D plane on the scene image without (left) and with (right) using the algorithm. Note the small white line on the right edge of the desk in the left image where the incorrectly estimated plane is.