Plane Registration Leveraged by Global Constraints for Context-Aware AEC Applications

Chen Feng, Vineet R. Kamat

Department of Civil and Environmental Engineering (University of Michigan), 2350 Hayward Street, Ann Arbor, Michigan, USA

Abstract: *In this paper, we propose a new registration algorithm and computing framework, the KEG tracker, for estimating a camera's position and orientation for a general class of mobile context-aware applications in Architecture, Engineering, and Construction (AEC). By studying two classic natural marker-based registration algorithms, Homography-from-detection and Homography-from-tracking, and by overcoming their specific limitations of jitter and drift, our method applies two global constraints (geometric and appearance) to prevent tracking errors from propagating between consecutive frames. The proposed method is able to achieve an increase in both stability and accuracy, while being fast enough for real-time applications. Experiments on both synthesized and real-world test cases demonstrate that our method is superior to existing state-of-the-art registration algorithms. The paper also explores several AEC applications of our method in context-aware computing and desktop augmented reality.*

1 INTRODUCTION

The ability to recover a user's pose (i.e., position and orientation within a certain coordinate system) is critical in many engineering domains such as Augmented Reality (AR), robotics, context-aware computing, and computer vision. In AR, for example, this task is termed as the "registration" problem (Azuma, 1997). In robotics, this task is closely related to "simultaneous localization and mapping" (SLAM) (Klein & Murray, 2007). In computer vision, "structure from motion" (SFM) algorithms are designed to solve this problem with little or no prior knowledge about the environment (Bao and Savarese, 2011). Context-aware engineering applications also face a similar problem where the positioning part is more relevant (Akula et al., 2011). In cinematography, the problem is called "move matching". For simplicity, we will refer to all of these as the registration problem in this paper.

Despite the rapid development of sensor technology—such as the global positioning system (GPS) and inertial measurement units (IMU), as well as angle sensors like the digital magnetic sensor, gyroscope, and accelerometer—this problem remains a challenge. A GPS signal is hardly available indoors, IMU suffers from the drifting effect after a while (Akula et al., 2011), and a magnetic sensor can be hugely affected by the changing environment, especially in challenging environments such as construction sites with all kinds of machines moving around, needless to mention the sensor's annoying jitter effect.

To overcome these technical insufficiencies, especially for indoor environments, infrastructure-based technology has been studied, such as RFID-based indoor tracking (Bekkali et al., 2007) and wireless local area network (WLAN)-based indoor positioning (Khoury and Kamat, 2009; Khoury and Kamat, 2007; Li et al., 2006). Yet these technologies are costly, inefficient, or sensitive to the environment, and lots of work has to be put into the system calibration stage. Further, these technologies' general inability to recover a user's orientation is troublesome for 3D visualization.

However, beyond all of those technologies, it is very interesting to note that a human being can figure out where s/he is to a certain degree of accuracy, given that s/he is familiar enough with that specific region of environment, mostly with the help of visual clues. This intuition inspires us to look into the developments in the computer vision community. Meaningful new tools that confront this registration problem could be developed if we could somehow capture the insight of this human-possessed ability and automate it.

Following this motivation, this paper proposes a new visual registration method called KEG planar object tracker, which essentially recovers the pose of the user, i.e. camera, in real-time from a set of planar markers whose own poses are known, capturing the idea that our tracker is familiar enough with the environment so as to perform an estimation

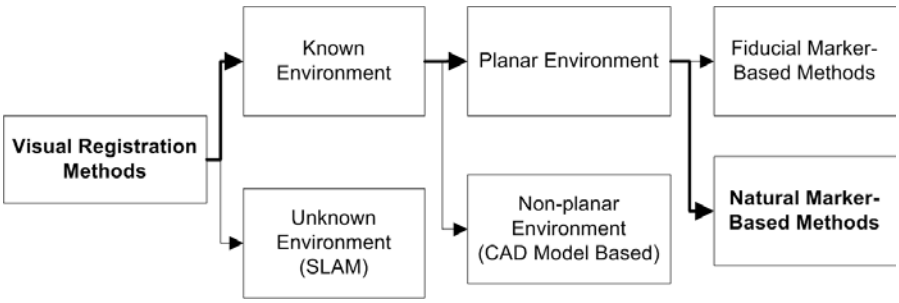


Figure 1 A brief taxonomy of visual registration methods based on assumptions on environment.

of position and orientation, just as humans do. In section 2, we will briefly introduce the state-of-the-art methods in visual registration. In section 3, we will explain in detail how the first computation step of our tracker works. Following is section 4, in which we will explain another class of planar marker-based algorithms that eventually serve as the second step of our tracker. Section 5 explains our main contribution in this paper—an efficient improvement based on the previous two classes of algorithm frameworks. After that, in section 6, we will describe several experiments that demonstrate the superiority, under different objective quality measures, of our algorithm framework. Also, in section 7, we describe two novel AEC applications that deploy our tracking algorithm. Finally, we draw our conclusions in section 8.

2 REVIEW OF PRIOR RELATED WORK

The visual registration problem is actively studied in the computer vision community, and several algorithms have been proposed to address it. Based on their different assumptions on the environment (i.e. the surrounding world where visual registration is going to be performed), these algorithms can be classified into two groups (Lepetit and Fua, 2005): known environment vs. unknown environment (See Figure 1. Our proposed method can be classified under natural marker-based methods). The first group of algorithms is easier to design since the only unknown is the user's pose. In the second group, an estimation of the environment has to be done along with the registration, which is naturally related to the visual SLAM problem that is being explored in the robotics community (Davison et al., 2007).

Generally, SLAM-based methods are more desirable since very few assumptions are made on the environment, which inherently makes those algorithms infrastructure-independent and thus enlarges their sphere of application. Non-visual SLAM-based methods typically rely on measurements from laser or light detection and ranging (lidar), where the cost and even the weight of those devices could be potential disadvantages of applying such technologies. Especially for context-aware computing in civil infrastructure applications (e.g. facility management, bridge inspection), weight is always an important consideration since human inspectors will not be able to comfortably use overly heavy devices. Although those developing visual SLAM algorithms can avoid these drawbacks by using a standard and lightweight web-cam as the main sensor, currently many limitations exist in such an approach (Klein and Murray, 2007), including the requirement of high computational power and small range restriction.

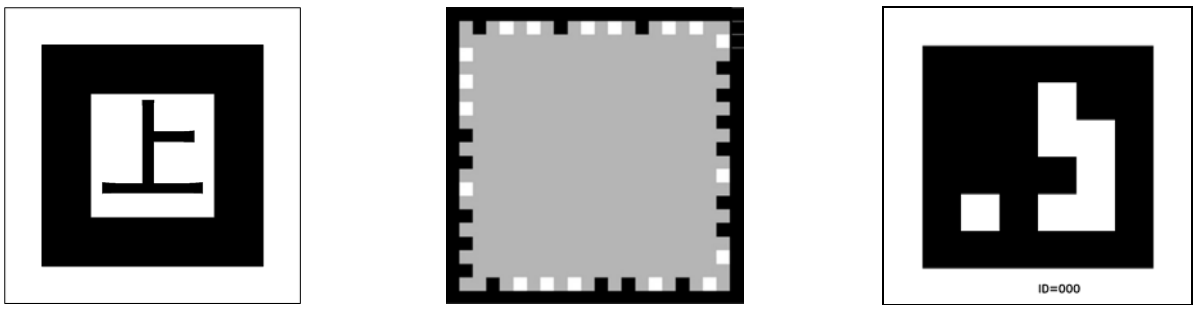
Different from the visual SLAM methods, the known environment methods have been well studied, and many powerful algorithms have been proposed in the last two decades. This makes it more realistic to apply them for solving real-world engineering issues. Within this class of methods, they can be further categorized into two groups: planar environment vs. non-planar environment. The first group is again easier to design because of the simple assumption made regarding the environment—a planar structure with known visual features. While the second group typically assumes known 3D structures with known visual features (usually CAD models), they are more often applied in a controlled environment with limited space, such as a small manufacturing workspace (Drummond and Cipolla, 2002).

The authors choose to take advantage of plane-based methods since planar structures are abundant in buildings, construction sites, and other human-made environments where engineering operations are conducted, which makes this type of method very convenient to apply. In addition, a planar structure can simply be an image printed out on a piece of paper and attached to a wall/floor of a corridor, with nearly zero cost. All of those advantages make this method ideal for application, and merit its investigation.

Plane-based methods can be further classified based on different visual features they adopt: fiducial marker vs. natural marker. In the following two subsections, the two types of markers will be introduced.

2.1 Fiducial Marker

A fiducial marker is composed of a set of visual features that are "easy to extract" and "provide reliable, easy to exploit measurements for the pose estimation" (Lepetit and Fua, 2005). Usually those features are a set of black and



(a) ARToolKit (Kato & Billinghurst, 1999). (b) Frame marker (Wagner et al., 2008). (c) AprilTag (Olson, 2011).

Figure 2 Examples of fiducial markers.

white patterns forming simple geometry by circles, straight lines, or sharp corners and edges.

Fiducial markers have been widely used in AR community for their simplicity and minimal computational requirement. ARToolKit (Kato and Billinghurst, 1999) is one of the earliest and most widely used fiducial marker-based AR systems. Since every ARToolKit marker is bounded by a wide black bounding box, the detection of the marker and registration can be solved by simply taking a threshold of the current input image and then discovering the four outer edges and corners so as to further estimate the camera's pose. A similar method is also adopted in the "frame marker" proposed by (Wagner et al., 2008) (see Figure 2 (a) and (b)).

Different from the simple image-processing algorithms in ARToolKit for detecting marker and estimating pose, AprilTag (Olson, 2011) clusters every pixel of the current input image based on its gradient and location, and then extracts multiple line segments as the boundary of each cluster. Finally, different quadrangles are discovered and used to decode a potential tag/marker ID (see Figure 2 (c)). This newly proposed method is proved through experiments to be superior to ARToolKit and many other state-of-the-art fiducial marker-based methods (Olson, 2011).

2.2 Natural Marker

Distinct from a fiducial marker, a natural marker does not require special predefined visual features. Instead, it treats any visual features in the same way. In this sense, any common image, ranging from a natural view to a company logo, can immediately be used as a natural marker, of course except for some images with fixed spatial frequency (i.e. images with the same color everywhere or repeated patterns). This major difference makes it much easier to set up a natural marker than a fiducial one. Users do not need to separately design special markers; they can simply take advantage of any meaningful pictures related to the problem of interest.

In addition, one major downside to a fiducial marker lies in the fact that it usually depends on the four corner points or edges of the marker's quadrangle to do further registration estimation, which is the reason that fiducial marker-based methods will fail even if one corner is not within view. This disadvantage does not exist in natural marker-based methods; in fact natural markers do not even require a marker image to be rectangular.

A natural marker uses more than four points to perform registration estimation. By detecting an appropriate number of feature points in both marker image and current image, a natural marker-based algorithm will then try to establish correspondences between these two sets of feature points. Once correspondences are established, further estimation can easily be made. Thus lots of algorithms have been proposed to tackle the main issue and most computational intensive part here, i.e. how to find as many correct correspondences as quickly as possible. Again, by the fundamental difference in the way they treat input images, these algorithms form two groups: one group treats each input image independently, which is referred to as a detection-based method, such as (Lowe, 2004; Ozuysal et al., 2007); the other group needs two or more consecutive images as input, which is referred to as a tracking-based method, such as (Jianbo and Tomasi, 1994; Lucas and Kanade, 1981). Since our proposed method evolves from both these algorithm groups, in the following sections (3 and 4) we will explain in detail the process framework of each type of algorithms, as well as how it inspires and is jointly adapted to our proposed algorithm framework.

3 HOMOGRAPHY FROM DETECTION

In either fiducial marker-based or natural marker-based algorithms, the fundamental task is to find the transformation between the marker image plane and the current camera plane which contains that current image frame. Such a transformation, which is called homography, maps points on the marker image to their corresponding points on the current image frame with the following equation:

$$s[x', y', 1]^T = \mathbf{H}[x, y, 1]^T \quad (1)$$

where $\mathbf{H}$ is a 3×3 matrix representing the homography, (x, y) and (x', y') are the corresponding points on the two images, and s is an unknown scaling parameter.

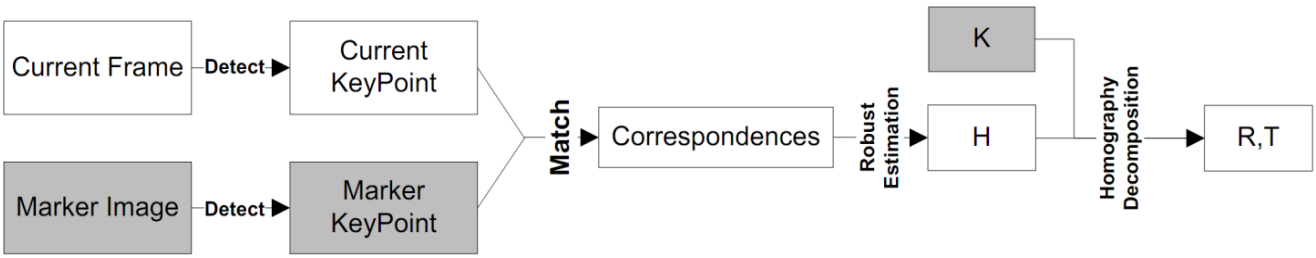


Figure 3 Homography-from-detection algorithm framework.

In fact, the general idea behind a plane-based registration algorithm is the fact that homography between two planes encodes the orientation and position information of one plane relative to another. From this perspective, registration is equivalent to finding the homography between the marker plane and the current camera plane. From projective geometry, one knows that with at least four point correspondences between two planes, their homography can be uniquely determined by solving a set of linear equations (Hartley and Zisserman, 2000).

More complicated than fiducial marker-based algorithms, which take advantage of simple patterns to find correspondences and then estimate homography, natural marker-based algorithms require a lot more effort to solve a correspondence problem.

Figure 3 shows the generic algorithm framework of the homography-from-detection type of methods. The gray components need be loaded or calculated once during a computation, while the white components need to be updated for each new frame. **H** is the homography between the current frame and the marker image. **K** is the camera calibration matrix storing the focal length and some other intrinsic parameters, which can be calibrated in advance. **R** is the rotation matrix representing camera orientation, and **T** is the translation vector representing the position of camera center. For each incoming image frame, the first step is to detect a set of keypoints. Also, at the very beginning, a fixed set of keypoints has to be detected on the marker image. Interest point detection algorithms are usually applied in this step, such as Harris corner detector (Harris and Stephens, 1988), FAST (Rosten and Drummond, 2006), Difference of Gaussian (DoG) (Lowe, 2004), and Laplacian of Gaussian (LoG) (Lindeberg, 1998).

The second step involves a matching problem, i.e. finding corresponding points between two sets of keypoints based on their local appearance. Among the state-of-the-art algorithms, the Scale Invariant Feature Transform (SIFT) algorithm by (Lowe, 2004) is perhaps the most famous and widely used nowadays. Although the SIFT algorithm works very well under large variation of visual conditions—illumination, scale, and rotation change among others—it is computationally time-consuming, which makes it impractical to be applied directly in real-time applications, such as a registration problem, even after applying lot of approximation to SIFT, as proposed in the SURF algorithm (Bay et al., 2008). Ozuysal et al. approach the matching problem from a very different perspective—matching as classification (Ozuysal et al., 2007). Their method, FERNs, differs from SIFT/SURF by the requirement of an offline training stage. Only after a long period of training using the marker image can FERNs recognize different keypoints on that particular marker under different visual conditions. Although the offline training requirement appears to be a disadvantage, FERNs enjoy the high-speed advantage. Recently, other faster methods (Rublee et al., 2011; Taylor et al., 2009) have also been proposed.

As mentioned, since most of these matching algorithms exploit a local feature descriptor, i.e. using image intensity information to describe a keypoint within only a limited neighboring region centered at that keypoint, mismatch is inevitable. In order to avoid most of these false matches, a robust estimation algorithm, such as the famous RANdom Sample And Consensus (RANSAC) (Fischler and Bolles, 1981) or its variants (Chum and Matas, 2005; Torr and Zisserman, 2000), is usually employed to estimate the homography.

3.1 Homography Decomposition

Once homography is found—through matrix decomposition techniques (Benhimane et al., 2005; Malis and Vargas, 2007) the camera position, the translation vector **T**, and orientation—the rotation matrix **R**, can be calculated. A simple yet effective method was proposed by (Simon et al., 2000).

If a point's 3D coordinate is (X,Y,Z) and its image on the camera plane has a 2D coordinate of (x,y), and if it is assumed that the camera is already calibrated so we know the focal length and principle point position that is stored in the **K** matrix, the camera projection model is (Hartley and Zisserman, 2000):

$$\begin{aligned}[x,y,1]^T &\sim \mathbf{P}[X,Y,Z,1]^T \\ &\sim \mathbf{K}[\mathbf{R},\mathbf{T}][X,Y,Z,1]^T,\end{aligned}$$

where "$\sim$" means the two vectors are equal up to a scale parameter, i.e. equal in the sense of projective geometry. Since we know the marker image is a 2D plane, it can be set to be on the X-Y plane without loss of generality. Thus the above projection equation can be rewritten as ($\mathbf{r}_i$ is the i-th column of **R**):

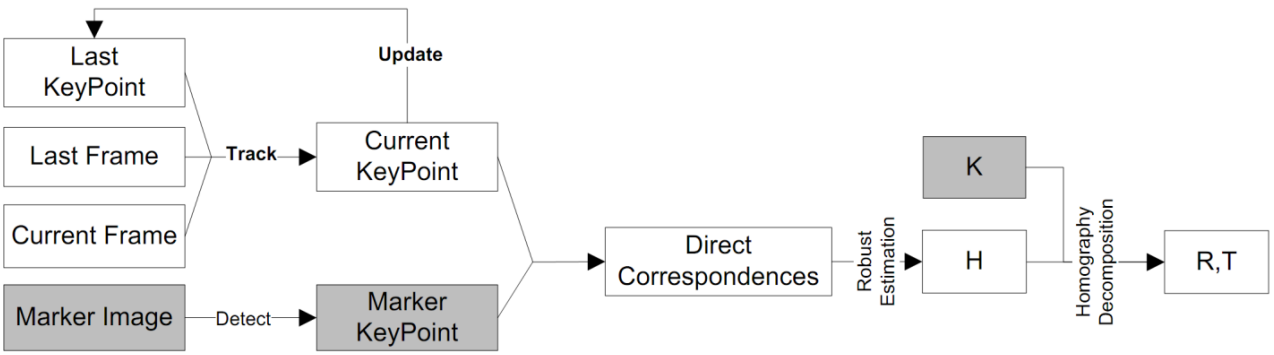


Figure 4 Homography-from-tracking algorithm framework.

$$\begin{aligned}[x,y,1]^T &\sim \mathbf{K}[\mathbf{r}_1,\mathbf{r}_2,\mathbf{r}_3,\mathbf{T}][X,Y,0,1]^T \\ &\sim \mathbf{K}[\mathbf{r}_1,\mathbf{r}_2,\mathbf{T}][X,Y,1]^T \\ &\sim \mathbf{H}[X,Y,1]^T.\end{aligned}$$

From the equation above, one can determine the following equations, which decompose the homography $\mathbf{H}$ between the current frame and the marker image into $\mathbf{R}$ and $\mathbf{T}$, to be:

$$\begin{aligned}\mathbf{R} &= [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_1 \times \mathbf{a}_2], \\ \mathbf{T} &= \mathbf{a}_3,\end{aligned} \tag{2}$$

where $\mathbf{a}_i$ is the i-th column of matrix $\mathbf{K}^{-1}\mathbf{H} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$ and "$\times$" means the cross product. Note again that the actual matrix to be decomposed is $\mathbf{K}^{-1}\mathbf{H}$ rather than $\mathbf{H}$, which means the camera needs to be calibrated in advance to get the $\mathbf{K}$ matrix.

4 HOMOGRAPHY FROM TRACKING

As shown in Figure 4 homography-from-tracking type of methods explore relations between consecutive frames. Since images of two such frames usually look very similar, correspondences needed for homography estimation can easily be maintained by tracking each keypoint around its local neighborhood. Thus this type of methods can circumvent the hardest matching problem, since in this framework, matching between the marker keypoint and the current keypoint to get correspondences is only needed at the very beginning; after that, keypoint correspondences are maintained by a tracking algorithm. Next we will briefly introduce two tracking algorithms that play a crucial role in our proposed method: the Kanade-Lucas-Tomasi (KLT) feature tracker and Efficient Second-order Minimization (ESM) algorithm.

4.1 Kanade-Lucas-Tomasi Feature Tracker

Proposed by (Lucas and Kanade, 1981), the KLT tracker's ultimate goal is to find the feature point displacement within two consecutive frames. It assumes that during these two frames, the local appearance of the feature point $\mathbf{x}$ does not change, and that the displacement is small. Thus in order to find the optimal displacement, these two assumptions of the invariance in local appearance can be mathematically represented as:

$$\begin{aligned}&\arg\min_{\mathbf{p}} \sum_{\mathbf{x}\in neighbor(\mathbf{z})} \left[I_1(\mathbf{x}+\mathbf{p}) - I_0(\mathbf{x})\right]^2 \\ &\Leftrightarrow \arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}\in neighbor(\mathbf{z})} \left[I_1(\mathbf{x}+\mathbf{p}+\Delta\mathbf{p}) - I_0(\mathbf{x})\right]^2.\end{aligned} \tag{3}$$

In equation (3), I_0 is the last frame, I_1 is the current frame, and $\mathbf{z}$ is the location of the point to be tracked in the image frame I_0. Note that $\mathbf{x},\mathbf{z},\mathbf{p},\Delta\mathbf{p}$ are all 2×1 vectors. Here, an image is a function that takes a 2D point as input and outputs a real intensity value, i.e. $I:\Re^2 \to \Re$. In order to solve this minimization problem, the KLT tracker applies the first-order Taylor expansion on the image function $I_1(\cdot)$ around $\mathbf{x}+\mathbf{p}$ in equation (3), leading to:

$$\arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}\in neighbor(\mathbf{z})} \left[\nabla I_1(\mathbf{x}+\mathbf{p})\Delta\mathbf{p} + I_1(\mathbf{x}+\mathbf{p}) - I_0(\mathbf{x})\right]^2, \tag{4}$$

where $\nabla I_1(\mathbf{x}+\mathbf{p})$ is the 1×2 image gradient of the current frame at location $\mathbf{x}+\mathbf{p}$.

This becomes a standard least-square problem with close-form solution. Since the KLT tracker uses a first-order approximation of the image function during the derivation, an iterative process is adopted here (see Algorithm 1). Note that this is a local tracking algorithm since in equation (3) and (4) only the neighborhood of the tracked point is $\mathbf{z}$ considered.

4.2 Efficient Second-order Minimization

As we discussed in section 4.1, the KLT tracker actually formulates the point tracking problem as a minimization problem. Its motion model is very simple, a 2D displacement. Similarly, if using the 2D homography as the motion model, and using second-order approximation of the image function, one will get a global refinement algorithm with a faster convergence rate, i.e. the ESM algorithm proposed by (Benhimane and Malis, 2004; Benhimane et al., 2005).

Algorithm 1 KLT algorithm that tracks a single keypoint $\mathbf{z}$ from last frame I_0 to current frame I_1.

Initiate $\mathbf{p} = [0,0]^T$.
Iterate:

1. Compute the error image $I_0(\mathbf{x}) - I_1(\mathbf{x}+\mathbf{p})$ around the neighbor of $\mathbf{z}$
2. Compute the image gradient of the current image ∇I_1 around the neighbor of $\mathbf{z}$
3. Compute $\Delta\mathbf{p}^*$ by solving equation (4)
4. Update $\mathbf{p} = \mathbf{p} + \Delta\mathbf{p}^*$

until $\|\Delta\mathbf{p}^*\|_{\text{„}}$ ò.

Algorithm 2 ESM global refinement algorithm.

Initiate $\mathbf{p} = \mathbf{0}$.
Pre-compute $J_W J_H$ as in equation (9), and also image gradient of template image T.
Iterate:

1. Warp current image I with $H(\mathbf{p})$, result in the warped image $I(H(\mathbf{p}))$
2. compute image gradient of the warped image $I(H(\mathbf{p}))$
3. compute J_{esm} as in equation (10)
4. compute error image $I(H(\mathbf{p})\cdot\mathbf{x}) - T(\mathbf{x})$
5. solve the least-square problem in equation (8)
6. update the homographic warp $H(\mathbf{p}) \leftarrow H(\mathbf{p})H(\Delta\mathbf{p}^*)$

until $\|\Delta\mathbf{p}^*\|_{\text{„}}$ ò.

Firstly, the optimization equation is:

$$\begin{aligned} &\arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \left[I(H(\mathbf{p})\cdot\mathbf{x}) - T(\mathbf{x})\right]^2 \\ &\Leftrightarrow \arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}} \left[I(H(\mathbf{p})H(\Delta\mathbf{p})\cdot\mathbf{x}) - T(\mathbf{x})\right]^2, \end{aligned} \tag{5}$$

where $H(\mathbf{p})$ is a homography function that takes an 8×1 parameter vector $\mathbf{p}$ as input and outputs a 3×3 homography matrix, satisfying that $H(\mathbf{0})$ is an identity matrix, and $\Delta\mathbf{p}$ is the updating parameter of the initial guess on optimal parameter $\mathbf{p}$; also, $H(\mathbf{p})\cdot\mathbf{x}$ means mapping the 2D point $\mathbf{x}$ using the homography $H(\mathbf{p})$ by equation (1). Notice that in this equation $\mathbf{x}$ is not limited to any local region. Instead, $\mathbf{x}$ can be any pixel location in the template image T, i.e. the marker image in our framework, which is why we say this is a global refinement. Then a second order Taylor expansion of the image function $I(H(\mathbf{p})H(\cdot)\cdot\mathbf{x})$ around updating parameter $\Delta\mathbf{p} = \mathbf{0}$ is performed:

$$\arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}} \left[I(H(\mathbf{p})\cdot\mathbf{x}) - T(\mathbf{x}) + J(\mathbf{0})\Delta\mathbf{p} + \frac{1}{2}\Delta\mathbf{p}^T M(\mathbf{0})\Delta\mathbf{p} \right]^2, \tag{6}$$

where $J(\mathbf{0})$ is the 1×8 Jacobian matrix of the function $I(H(\mathbf{p})H(\cdot)\cdot\mathbf{x})$, and $M(\mathbf{0})$ its 8×8 Hessian matrix, all evaluated at $\Delta\mathbf{p} = \mathbf{0}$ and location $\mathbf{x}$.

Next, let's consider the novelty of ESM. We can again apply a first-order approximation of the Jacobian matrix $J(\Delta\mathbf{p}) = J(\mathbf{0}) + \Delta\mathbf{p}^T M(\mathbf{0})$ and substitute it into equation (6) to get:

$$\arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}} \left[I(H(\mathbf{p})\cdot\mathbf{x}) - T(\mathbf{x}) + \frac{1}{2}\left(J(\mathbf{0}) + J(\Delta\mathbf{p})\right)\Delta\mathbf{p} \right]^2. \tag{7}$$

Thus we get a second-order approximation without calculating the Hessian matrix, whose computation is time-consuming. Let $J_{esm}(\Delta\mathbf{p}) = \frac{1}{2}\left(J(\mathbf{0}) + J(\Delta\mathbf{p})\right)$, then the optimization becomes:

$$\arg\min_{\Delta\mathbf{p}} \sum_{\mathbf{x}} \left[J_{esm}(\Delta\mathbf{p})\Delta\mathbf{p} + I(H(\mathbf{p})\cdot\mathbf{x}) - T(\mathbf{x})\right]^2. \tag{8}$$

If we know how to compute the $J_{esm}(\Delta\mathbf{p})$ without knowing $\Delta\mathbf{p}$, our problem is again a standard least-square problem similar to equation (4).

So the only problem now is the calculation of $J_{esm}(\Delta\mathbf{p})$. Let's consider the first half of it. Applying the chain rule, we have:

$$\begin{aligned} J(\mathbf{0}) &= \nabla I(H(\mathbf{p})H(\Delta\mathbf{p})\cdot\mathbf{x})\,|_{\Delta\mathbf{p}=\mathbf{0}} \\ &= \nabla I(H(\mathbf{p}))\nabla(H\cdot\mathbf{x})\,|_{H=H(\mathbf{0})=I}\ \nabla H(\mathbf{p})\,|_{\mathbf{p}=\Delta\mathbf{p}=\mathbf{0}} \\ &= J_I J_W J_H, \end{aligned} \tag{9}$$

where J_I is the image gradient of the warped image $I(H(\mathbf{p}))$ at location $\mathbf{x}$, J_W is the Jacobian of the homographic mapping function $W(\mathbf{x};\mathbf{p}) = H(\mathbf{p})\cdot\mathbf{x}$, and J_H depends on the choice of the parameterization of the

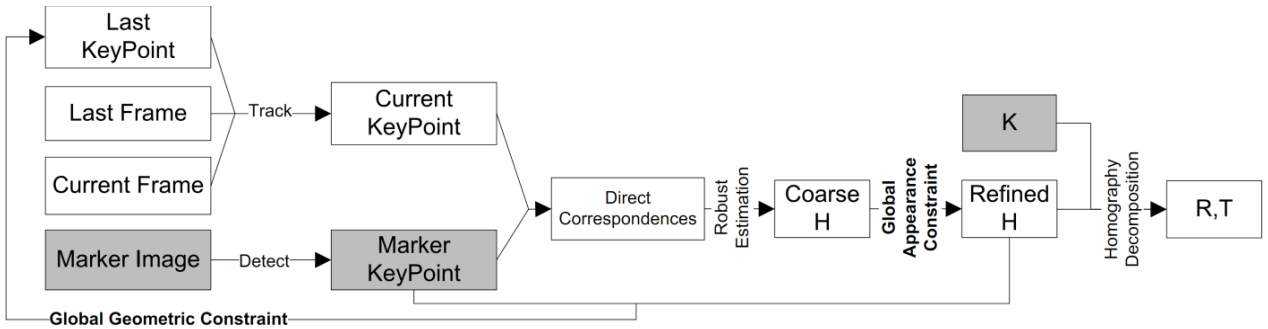


Figure 5 Proposed algorithm framework.

homography. J_W and J_H can be pre-computed once and for all.

In order to prevent direct calculation of $J(\mathbf{\Delta p})$, since $\mathbf{\Delta p}$ is the variable to be optimized, Benhimane et al. chose to use a Lie Algebra method for parameterizing the homography (Benhimane and Malis, 2004). Thanks to this method, the calculation of $J(\mathbf{\Delta p})$ can be circumvented by $J(\mathbf{\Delta p}) = J_T \cdot J_W \cdot J_H$, where J_T is the image gradient of the template image T. This means:

$$J_{esm}(\mathbf{\Delta p}) = \frac{1}{2}(J_I + J_T)J_W J_H. \quad (10)$$

Since, in the above equations, the computation of J_I, J_T, J_W and J_H only requires the location $\mathbf{x}$ and the $\mathbf{p}$ as the current guess of homography parameters, without the need of knowing $\mathbf{\Delta p}$, our previous problem is solved. In summary of the above derivation, one can get the ESM global refinement algorithm (see Algorithm 2).

5 GLOBAL GEOMETRIC AND APPEARANCE CONSTRAINTS

The advantage of homography-from-detection methods lies in the fact that since they treat each image frame separately, as we have shown in Figure 3, estimation can be totally wrong at one particular frame, and the following frames won't be affected at all. However, the problem with methods such as SURF and FERNs is that, in order to speed up the time-consuming matching step in detection, lots of approximations are adapted. This makes the homography estimation very unstable, resulting in a very annoying jitter effect if adopted in AR applications, i.e. the augmented object appears to be shaking in the scene (Kamat and El-Tawil, 2007). In our experiments, even if the camera is fixed, the estimated camera position and orientation could have very large variance.

In a different approach, homography-from-tracking methods, such as ESM and KLT, compare the current frame with the previous one in order to track the change. Although they benefit from the relatively higher tracking speed that improves their frame rate, one critical problem of this group of methods is that every tracker suffers from the drifting effect, i.e. the updated position of the tracked point actually differs to some extent from its true new position, and will thus eventually fail. The drifting effect will lead to large error in homography estimation, since these drifting errors usually do not follow Gaussian distribution and shall be seen as systematic errors that are changing dynamically. Also, the greater the number of points failing to be tracked, the larger the variance that the estimated camera pose could have. Thus the augmented objects could be in a wrong position and shaking at the same time.

Our new framework (Figure 5) integrates the homography-from-detection and homography-from-tracking frameworks, utilizing their strong points and circumventing their short-comings. In general, our framework starts by the original homography-from-detection framework. Once the marker image is detected along with a rough estimation of the homography, we immediately move into a coarse-to-fine framework. Only when the track is somehow lost will this procedure be repeated. Within our new framework, and firstly via the original tracking algorithm (KLT), a coarse homography could be found. Then, it would be refined by a global optimization algorithm (ESM). Finally the refined homography would be used to correct the positions of the set of points to be tracked in the next frame, which is inspired by our following analysis of the cause of the drifting effect.

5.1 Drifting Effect Analysis

After analyzing the drifting effect in homography-from-tracking methods, we found that it is actually an error accumulation issue. During the tracking between every two consecutive frames, the error introduced by any tracking algorithm is accumulated. After a while, this accumulation could directly lead to the tracking of a wrong local target or even to the failure of the tracker. After realizing this, one pertinent question to ask is: is there any way to correct the error before the next tracking is actually performed? It was found that this is possible, which led to the design of our proposed framework. To gain more understanding of the

Algorithm 3 KEG algorithm framework.

1. Detect N keypoints $\{\mathbf{x}_{ref}\}$ on marker image T.
2. Apply Fiducial Marker method (AprilTag) or Homography-from-detection algorithm (SURF), try to find the marker image and its corresponding homography $\mathbf{H}_{refined}$. If found, go to step 6; otherwise, re-do step 2.
3. Take a new incoming frame I_{new}, the last frame I_{old}, and the old keypoint positions $\{\mathbf{x}_{old}\}$, perform local tracking (KLT) and output new keypoint position $\{\mathbf{x}_{new}\}$.
4. Perform robust estimation (RANSAC) on correspondent keypoint array $\{\mathbf{x}_{ref}\}$ and $\{\mathbf{x}_{new}\}$ and output $\mathbf{H}_{coarse}$.
5. Apply global appearance constraint by equation (12) and output $\mathbf{H}_{refined}$.
6. Validate $\mathbf{H}_{refined}$ by similarity (zero-mean normalized cross-correlation) threshold (equation (14)). If valid, $\mathbf{H}_{refined}$ can be output for homography decomposition by equation (2); otherwise, meaning loss-of-track, go to step 2.
7. Update position of keypoints $\{\mathbf{x}_{new}\}$ using global geometric constraint by equation (13).
8. Replace I_{old} with I_{new}. Replace $\{\mathbf{x}_{old}\}$ with $\{\mathbf{x}_{new}\}$. Go to step 3 until no new incoming frames.

cause of the drifting effect, the detailed error analysis is shown, as follows.

Firstly, the error source of any tracking algorithm, such as KLT, can be seen as composed by the following terms:

$$\hat{\mathbf{x}}_{new} = \mathbf{x}_{old} + \Delta\mathbf{x} + \varepsilon_d + \varepsilon_g \tag{11}$$

where $\hat{\mathbf{x}}_{new}$ is the updated position estimated by KLT, $\mathbf{x}_{old}$ is the true original position, $\Delta\mathbf{x} = \mathbf{x}_{new} - \mathbf{x}_{old}$ is the true displacement, ε_d is the systematic drifting error, and ε_g is the rest of the error, which is assumed to follow some Gaussian distribution. Here, $\Delta\mathbf{x}$ is caused by physical movement between the camera and the scene, ε_g is mainly caused by camera CCD sensor noise, and ε_d is usually caused by the tracking algorithm and other complicated reasons, such as the fact that KLT will be affected a lot when the camera is moving too fast, which leads to motion blur and thus violates KLT's underlying assumptions.

The second step of homography-from-tracking methods applies robust estimation algorithms (e.g. RANSAC) to estimate $\mathbf{H}_{coarse}$ from the array of tracked points $\{\hat{\mathbf{x}}_{new}\}$ and their corresponding points on marker image. However, even though RANSAC can eliminate a lot of outliers if the absolute value of error $|\varepsilon_d + \varepsilon_g|$ exceeds some threshold, and further, can eliminate the Gaussian error ε_g by a final least-square estimation on the outlier-free subset of correspondent points, there still remains a part of systematic drifting error ε_d not handled and thus propagated into $\mathbf{H}_{coarse}$. In the homography-from-tracking framework, neither ε_d nor ε_g are corrected during the update step, so these errors are accumulated, which can cause a large drift even after a few frames of tracking.

5.2 Error Correction by Global Constraints

One natural way to reduce the effect of ε_d is to apply the **global appearance constraint**, as shown in equation (12),

$$\begin{aligned} \mathbf{\Delta p}^* &= \arg\min_{\mathbf{\Delta p}} \sum_{\mathbf{x}} \left[I(\mathbf{H}_{coarse} H(\mathbf{\Delta p}) \cdot \mathbf{x}) - T(\mathbf{x}) \right]^2, \\ \mathbf{H}_{refined} &= \mathbf{H}_{coarse} H(\mathbf{\Delta p}^*), \end{aligned} \tag{12}$$

which essentially means that the original marker image will look the same as the image rectified from the current frame by estimated homography $\mathbf{H}$. Before this step, all of the information used by the KLT tracker is local, while ε_d is systematic, therefore a global optimization based on the whole marker image, i.e. the global appearance constraint represented by equation (12), will theoretically eliminate all the systematic error and $\mathbf{H}_{coarse}$ can serve as a good optimization starting point.

After the drifting error is eliminated when estimating the refined homography $\mathbf{H}_{refined}$, we can easily correct tracking errors and update keypoint positions to be filled into the next tracking iteration by the homography mapping:

$$\bar{\mathbf{x}}_{new} = \mathbf{H}_{refined} \mathbf{x}_{ref} \tag{13}$$

where $\mathbf{x}_{ref}$ are keypoint positions on the original marker image; we refer to this as applying the **global geometric constraint** (for it relies on the prior knowledge that all keypoints lie in the same plane). Since the estimated $\mathbf{H}_{refined}$ is already theoretically error-free, updating using the above equation (13) instead of equation (11) prevents tracking error from propagating into the tracking of the next frame, and thus increases the tracking stability.

Besides the improvement in accuracy, our algorithm also enjoys an increase in tracking speed. Because we have a global refinement step, we do not require the local tracking algorithm such as KLT to be very accurate by reducing the number of iterations of KLT algorithms that result in larger

error $\left|\varepsilon_d + \varepsilon_g\right|$. Since the direct result of KLT is just a coarse homography serving as an ESM optimization starting point, a certain amount of error can be tolerated and will be theoretically eliminated after global refinement (ESM). Similarly, since the time complexity of a local tracking algorithm such as KLT is usually positively correlated to the number of points to be tracked, we can decrease the number of keypoints to be tracked.

We refer to our method as the KEG (KLT Enhanced by Global constraints) tracker, and the complete algorithm framework is described in Algorithm 3. It's worth noting that KLT, ESM, and RANSAC, as well as the initial detection method (AprilTag/SURF), are replaceable components in our approach. This makes our method very flexible and easy to be extended by new algorithms (as long as they serve the same purpose). We will offer detailed comparisons in section 6 showing that, even though our framework involves more steps, its performance in accuracy, stability, and speed is increased as compared to the state-of-the-art algorithms.

6 EXPERIMENTAL RESULTS

In order to validate our method and compare it to state-of-the-art algorithms, we did several experiments on both real-world and synthesized video sequences (in which we knew the ground-truth of the camera pose). Experiments were all conducted on a desktop computer with an eight-core 2.8 GHz Intel Core i7 CPU with 6 GB memory. Also, all of the video sequences have a frame size of 640×480, which is the commonly adopted size of commercial webcams.

In all of the test cases to be shown in the following, for the purpose of showing the necessity of the three core components in KEG—local tracker (K-step), global refinement (E-step), and error correction (G-step) —and proving its superiority to state-of-the-art methods, we ran 7 different algorithms over those test cases:

1. KEG with AprilTag as initialization method (referred to as A+KEG).
2. No global appearance constraints applied; others are the same as 1 (A+K G).
3. No global geometric constraints applied; others are the same as 1 (A+KE).
4. No global constraints applied, representing homography-from-tracking method (A+K).
5. AprilTag, representing fiducial marker-based method (A).
6. AprilTag with global appearance constraints applied (A+ E).
7. FERNs, representing homography-from-detection method (FERNs).

For the homography-from-detection component, we used our own C++ implementation of AprilTag (http://code.google.com/p/cv2cg/), which originates from the java implementation by April Lab (http://april.eecs.umich.edu/wiki/index.php/AprilTags) at the University of Michigan (Olson 2011). Our KEG algorithm is open-source, and the C++ code can be found at http://code.google.com/p/cv2cg/. For comparison with state-of-the-art homography-from-detection methods, we adopted the well-known and widely used Open Source Computer Vision (OpenCV) library (http://opencv.willowgarage.com) implementation of the FERNs method, which also provides the implementation of KLT. For the ESM method, we used the binary library provided by INRIA Sophia-Antipolis at http://esm.gforge.inria.fr/ESMdownloads.html.

We also looked at different performance metrics so as to have a comprehensive understanding of the performance of these algorithms:

Duration: The time to process each frame, reflecting the speed of the algorithm. This metric is crucial for real-time applications.

NCC: The zero-mean normalized cross-correlation between the original marker image I_1 and the rectified image I_2 by $\mathbf{H}_{refined}$, which can be calculated by:

$$NCC(I_1, I_2) = \frac{1}{n}\sum_{\mathbf{x}} \frac{[I_1(\mathbf{x}) - m_1][I_2(\mathbf{x}) - m_2]}{\sigma_1 \sigma_2} \tag{14}$$

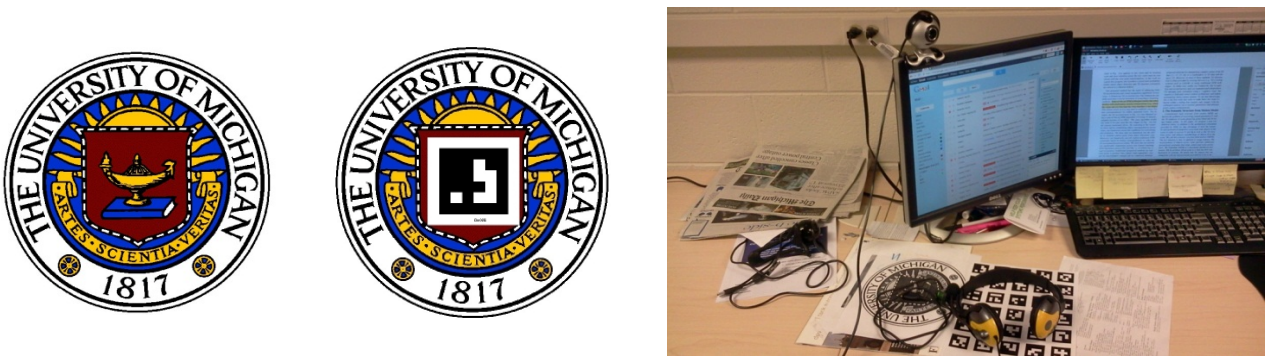
where n is the total number of pixels of image I_1 or I_2, and m_i and σ_i are the mean value and standard deviation of intensity of image I_i. Obviously, if I_1 and I_2 are exactly the same, their NCC index should be one, and the larger their difference, the smaller the NCC index. This means NCC is a good similarity index (Ladikos et al., 2007). This index is also used in KEG to determine whether it loses track or not by a simple threshold of 0.5; if at any frame the NCC index is smaller than 0.5, it is regarded as a loss-of-track frame.

LOT: The total number of loss-of-track frames. This metric represents a registration algorithm's stability to some extent.

T-RMS/R-RMS: The root mean square error between estimated camera position/orientation and ground truth. This metric represents the absolute accuracy of a registration algorithm, and is calculated by:

$$\begin{aligned} T-RMS &= \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left\|\mathbf{T}_i - \hat{\mathbf{T}}_i\right\|^2}, \\ R-RMS &= \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left\|e_i - \hat{e}_i\right\|^2}, \end{aligned} \tag{15}$$

where k is the total number of frames of a test case, $\mathbf{T}_i$ and $\hat{\mathbf{T}}_i$ are the i-th frame's ground truth and estimated position vector of dimension 3×1, and e_i and $\hat{e}_i$ are the i-th frame's ground truth and estimated Euler angle vector of



(a) UM Logo. (b) Our Marker Image. (c) Background Image of Synthesized Test Cases.

Figure 6 Marker image composition.

dimension 3×1, respectively. Since these two indices require ground truth data, they are only examined for synthesized test cases.

UOT: We also propose this new index for estimating the extent of jitter effect of a registration algorithm, i.e. the unsmoothness-of-tracking, taking advantage of the NCC index by

$$d_i = NCC_{i+1} - NCC_i, \forall i = 1,2,\cdots,k-1,$$
$$UOT = \sqrt{\frac{1}{k-1}\sum_{i=1}^{k-1}(d_i-\mu)^2}, \mu = \frac{1}{k-1}\sum_{i=1}^{k-1} d_i, \quad (16)$$

where NCC_i is the NCC index of the i-th frame, which essentially means that UOT index is the standard deviation of the difference between consecutive NCC indices. In MATLAB, this can be simply calculated by "*std(diff(ncc))*." A stable registration algorithm should give a UOT index value as small as possible.

In all of the test cases, our marker image is composed of a 16 bits AprilTag of ID equal to zero (Figure 2c) and a natural image, the logo of University of Michigan (Figure 6a), that is rich in features (Figure 6b). Note that using AprilTag here does not mean that our method is fiducial marker based method. As explained in step 2 of Algorithm 3, applying either fiducial marker based method or Homography-from-detection based method (SURF) could serve as a starting point of our method in the first video frame. However, even though AprilTag is not a necessary component here, we choose to add it to support multiple markers, which is very useful in many applications to be explained in section 7.

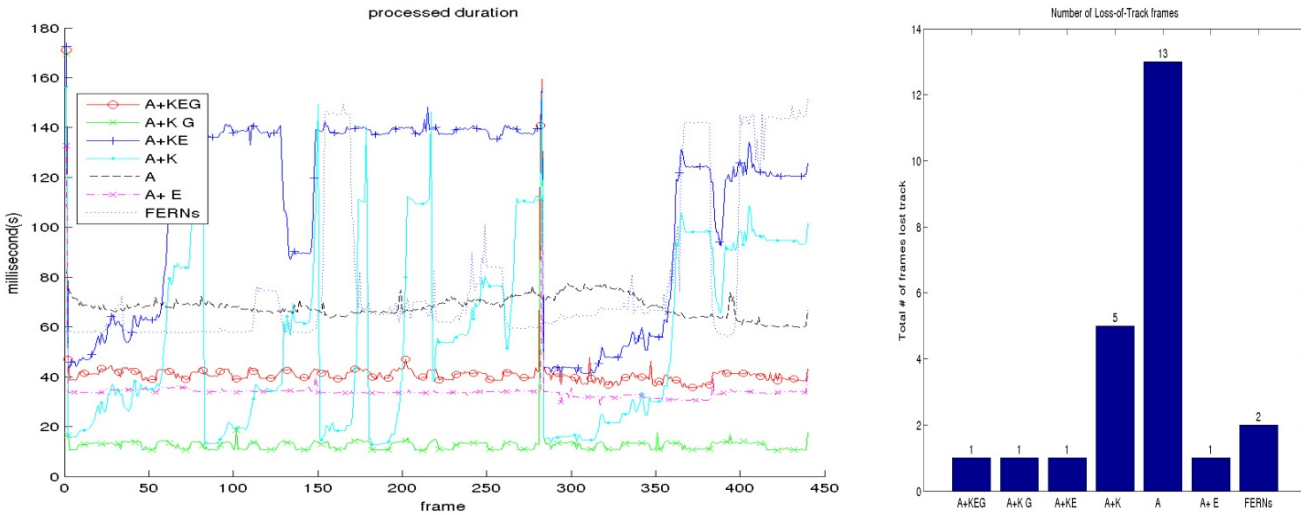
6.1 Synthesized Test Cases

We have synthesized three test cases in OpenGL, using our marker image and a static real-world image as background (Figure 6c). The first test case, S-1, simulates a random camera movement of both position and orientation change with 440 frames. The second test case, S-2, simulates 421 frames of the situation in which the camera moves around the marker image with a fixed distance. The last test case, S-3, simulates 304 frames of the situation in which the camera first moves close to the marker image and then far away, and the camera plane is parallel to the marker image plane. The different performance measures of different algorithms are shown in Figure 7, Figure 8 and Figure 9.

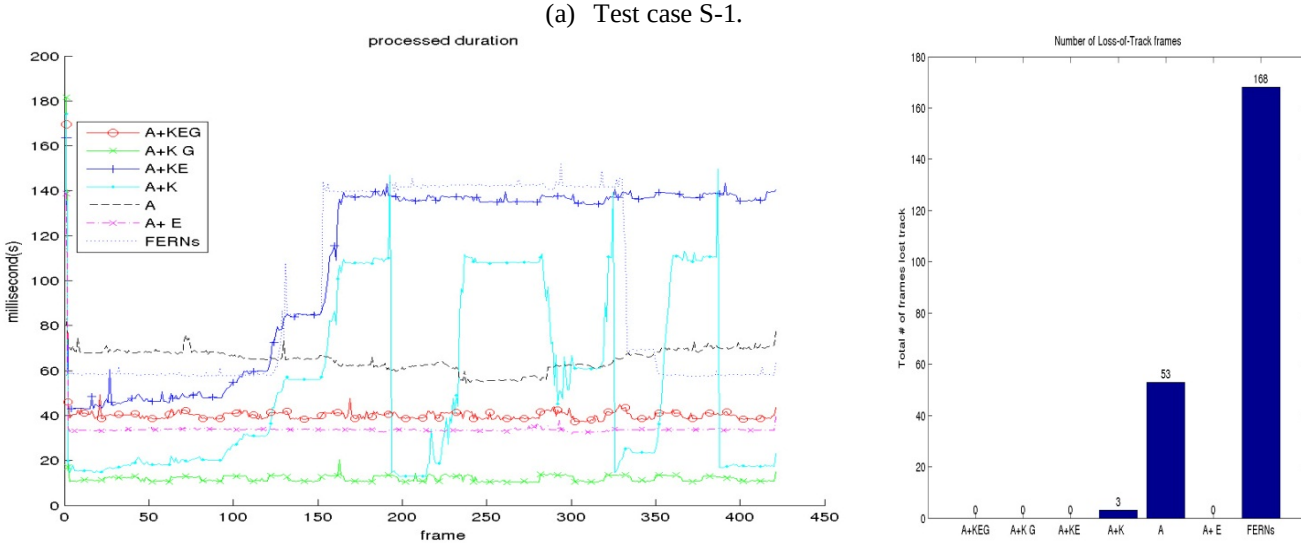
From the left column of Figure 7 one can see that the average processing time of A+KEG is about 40 milliseconds, which is even faster than AprilTag. While other algorithms have very unstable processing time and are mostly slower than A+KEG, the only exception is A+KG, which is as expected since it has no global refinement step. These curves prove that the KEG method does fit for real-time applications and that it can process images at 20 frames-per-second or faster.

From the left column of Figure 8 one can find that, even though sometimes AprilTag might have a slightly higher NCC value, in most cases, the NCC curve of A+KEG is the upper bound for the other algorithms, especially performing better than the state-of-the-art algorithm, FERNs.

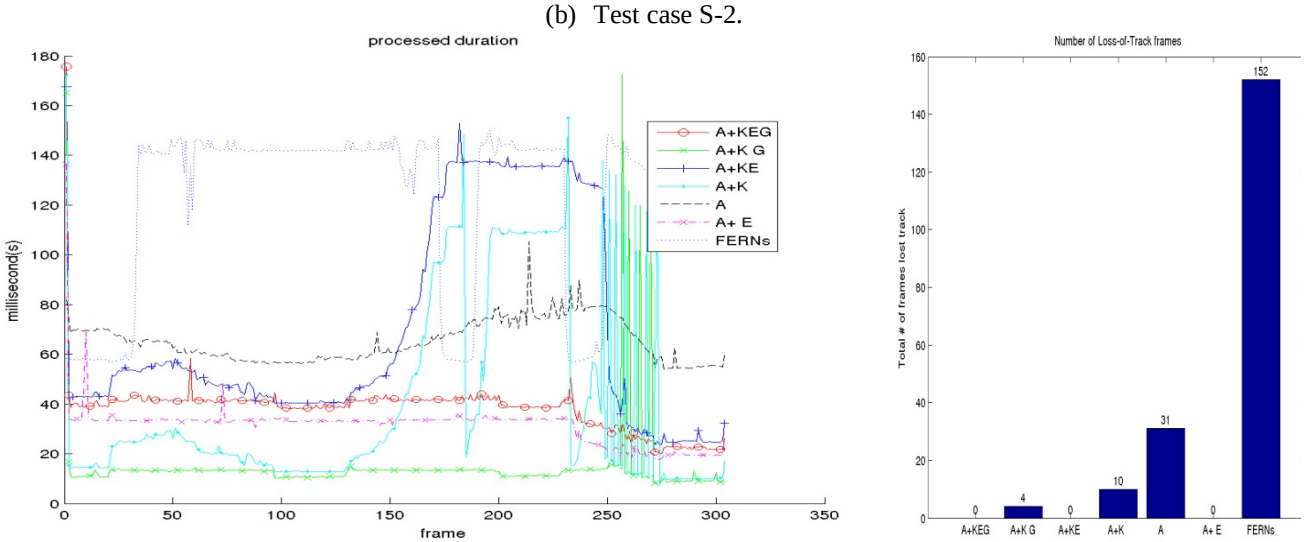
From the right column of Figure 7 and Figure 8, one can figure out that the A+KEG method has fewer loss-of-track frames, showing its ability to track longer, and lower UOT index, showing its smoothness in tracking—an important feature if applied in AR. Also A+KEG is more accurate, by giving less T-RMS/R-RMS errors in Figure 9. Notice that here we assume the radius of our synthesized marker is 20 cm (which was the real size when it was printed out on an A4 sheet of paper in the real-world test cases). In this configuration, the KEG tracker's maximum working distance can be as far as 3 meters, and its maximum working Euler angle can be about 85 degree offset from the marker image's normal direction. If an even larger working distance is desirable, a higher resolution camera and bigger marker can be adopted.



(a) Test case S-1.

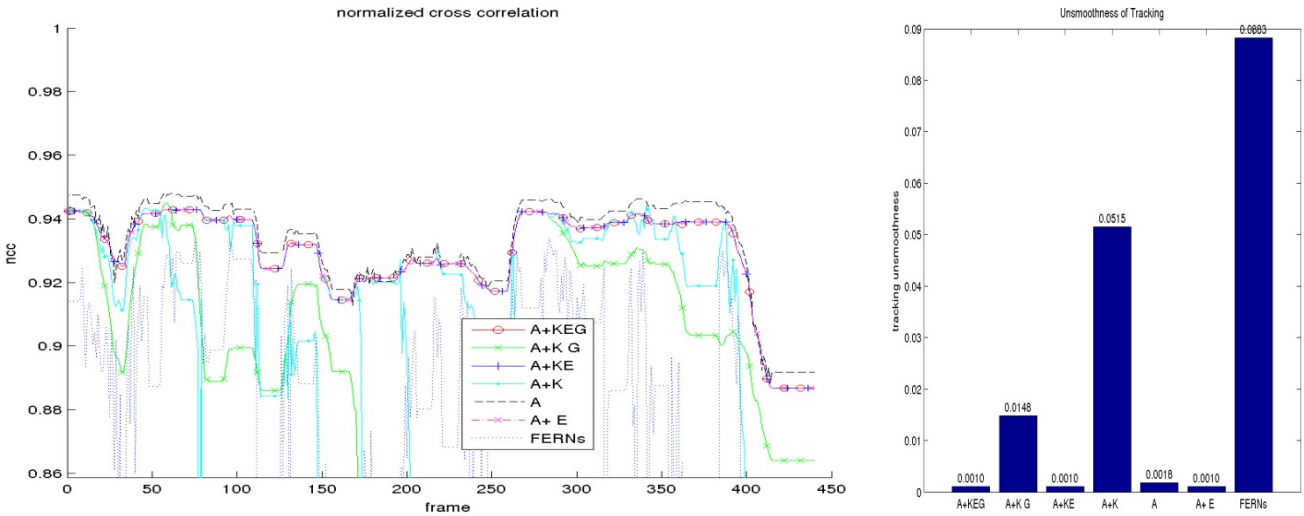


(b) Test case S-2.

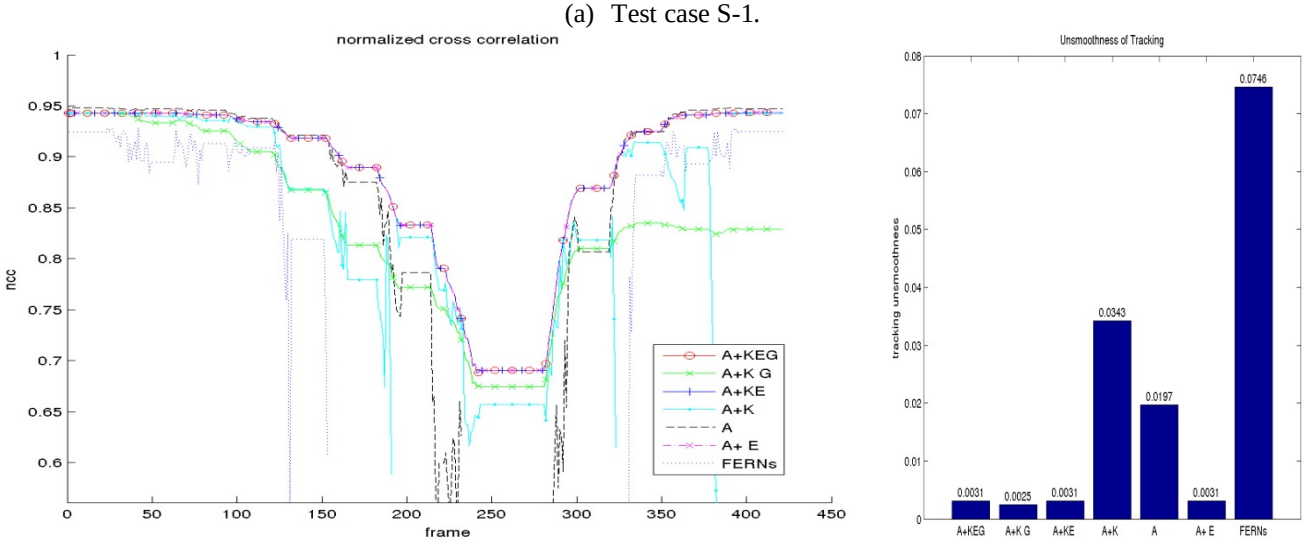


(c) Test case S-3.

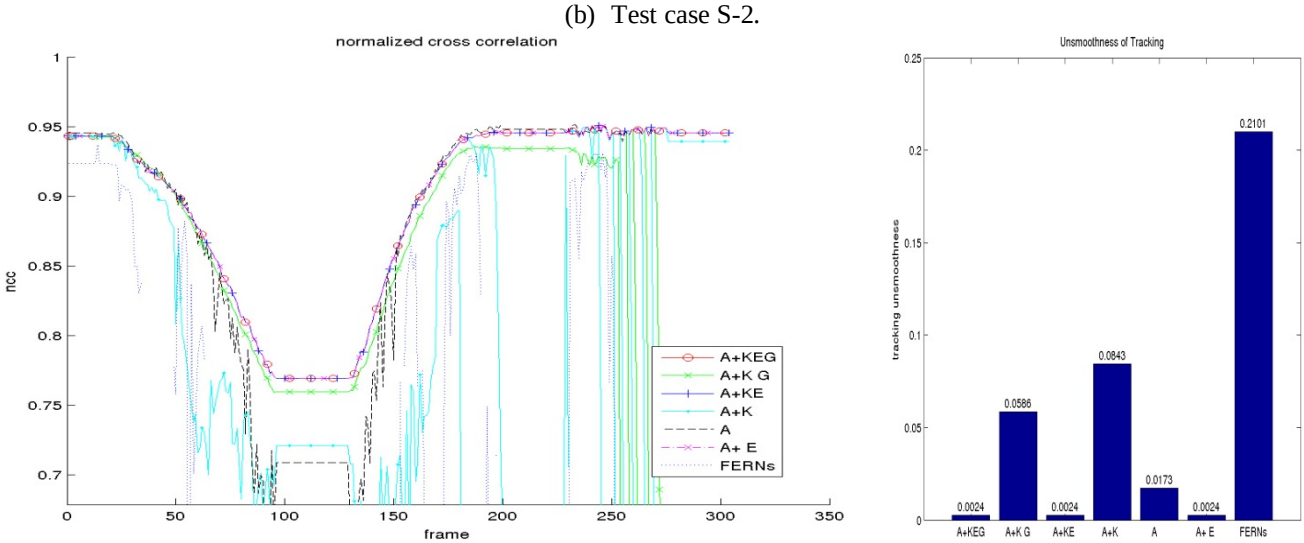
Figure 7 Duration curves (left) and LOT bars (right) for synthesized test cases (best viewed in color).



(a) Test case S-1.

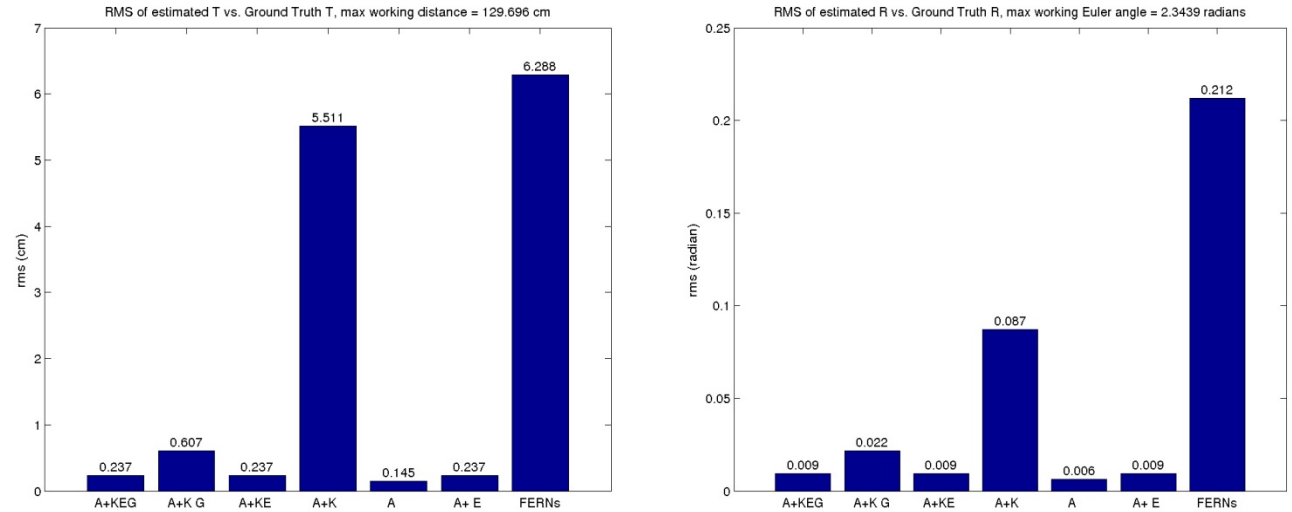


(b) Test case S-2.

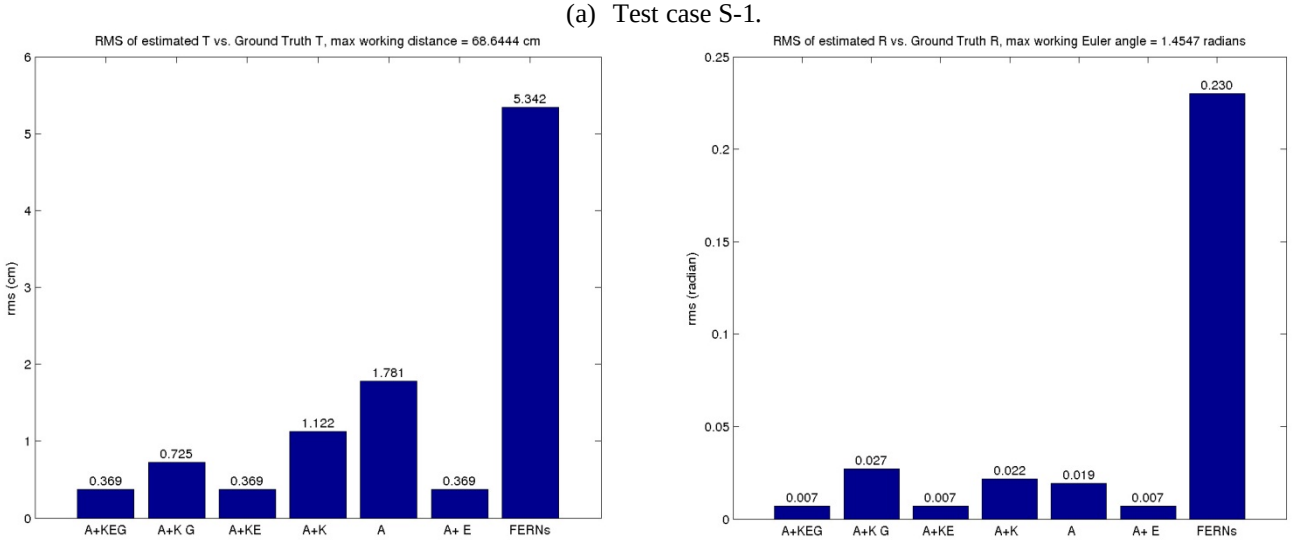


(c) Test case S-3.

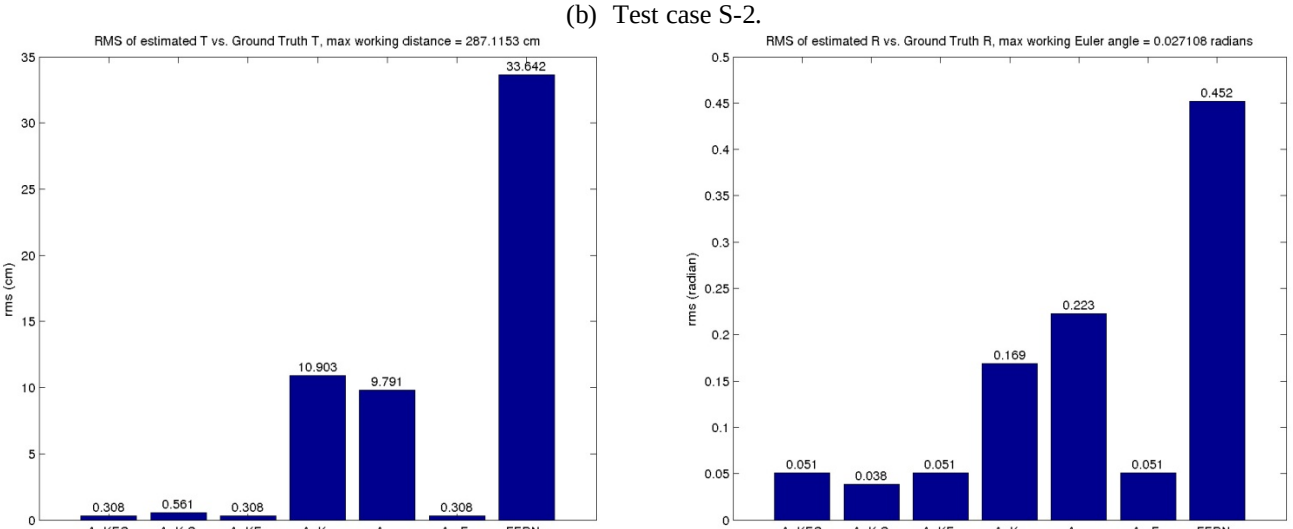
Figure 8 NCC (left) curves and UOT (right) bars for synthesized test cases (best viewed in color).



(a) Test case S-1.

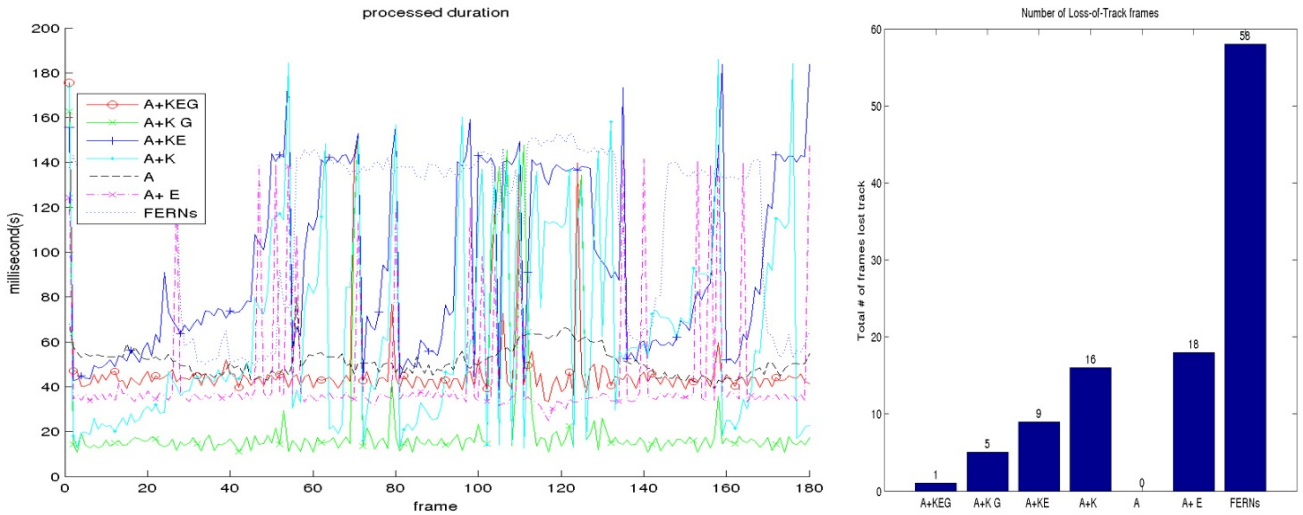


(b) Test case S-2.

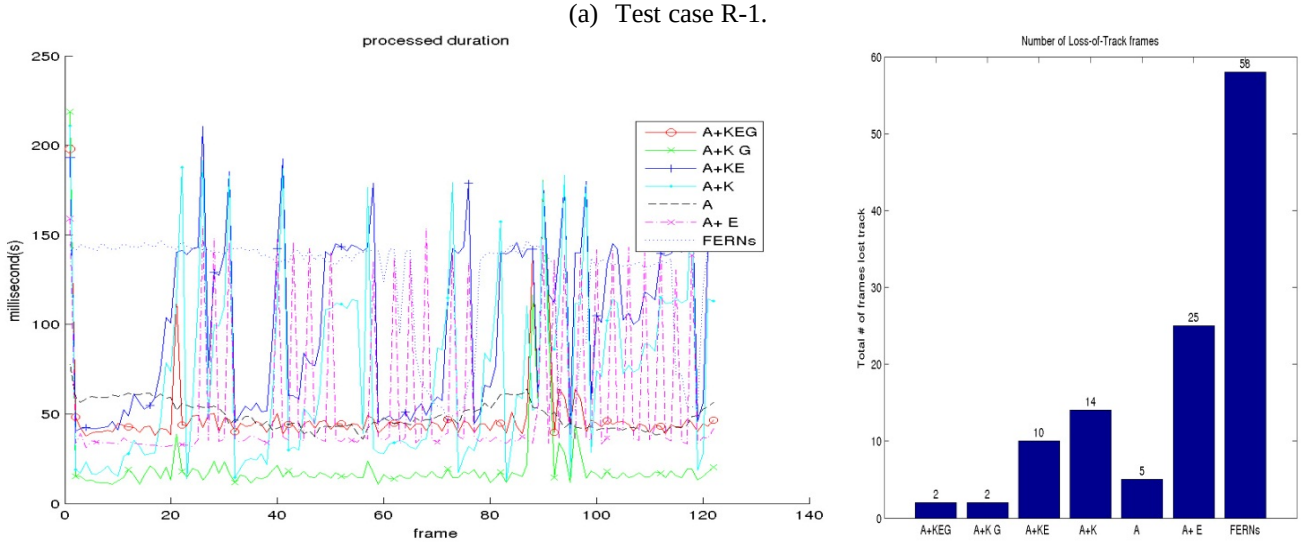


(c) Test case S-3.

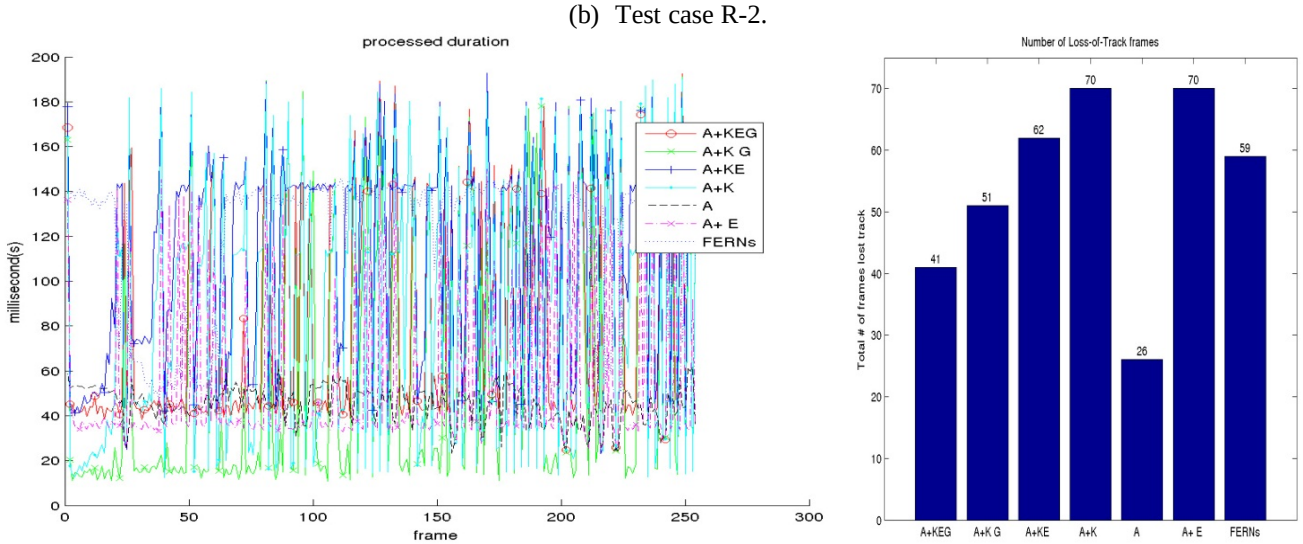
Figure 9 T-RMS (left) and R-RMS (right) bars for synthesized test cases.



(a) Test case R-1.

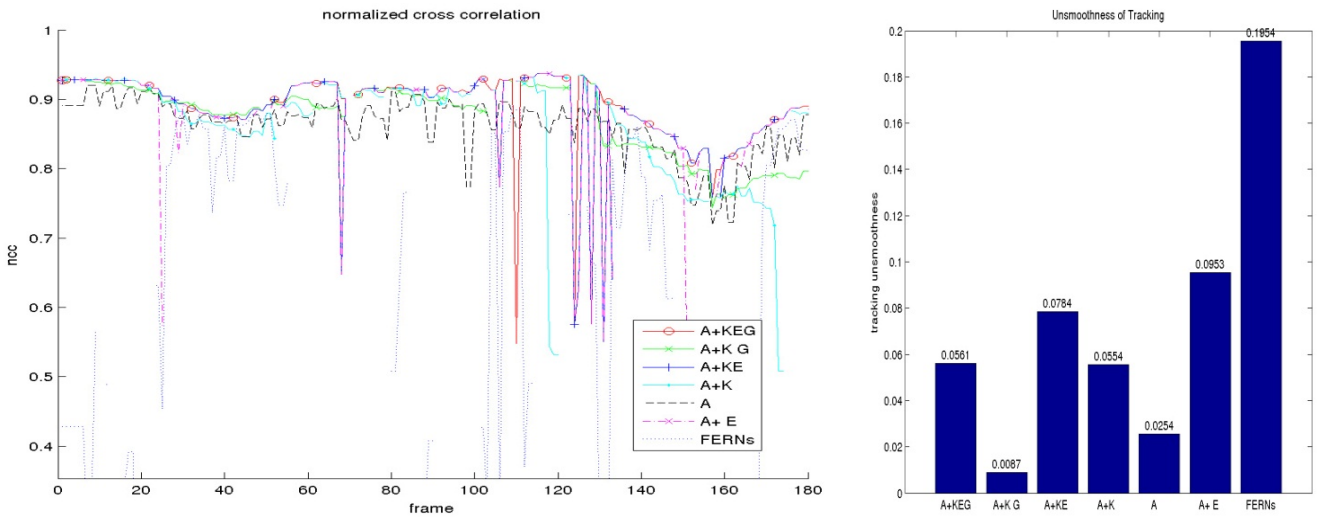


(b) Test case R-2.

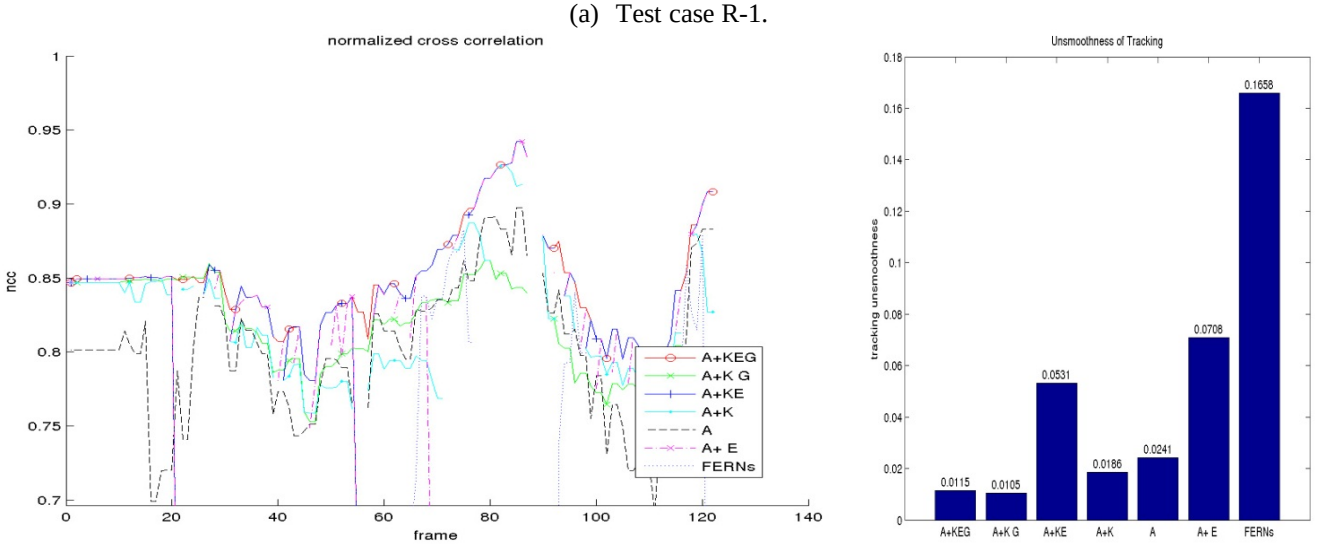


(c) Test case R-3.

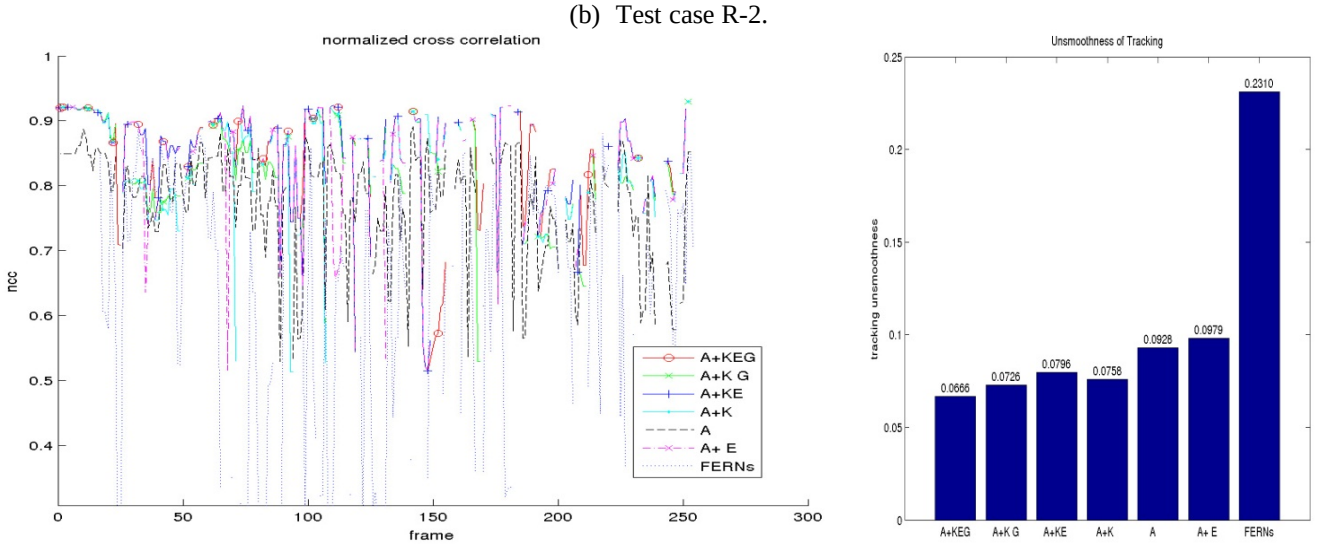
Figure 10 Duration curves (left) and LOT bars (right) for real-world test cases (best viewed in color).



(a) Test case R-1.

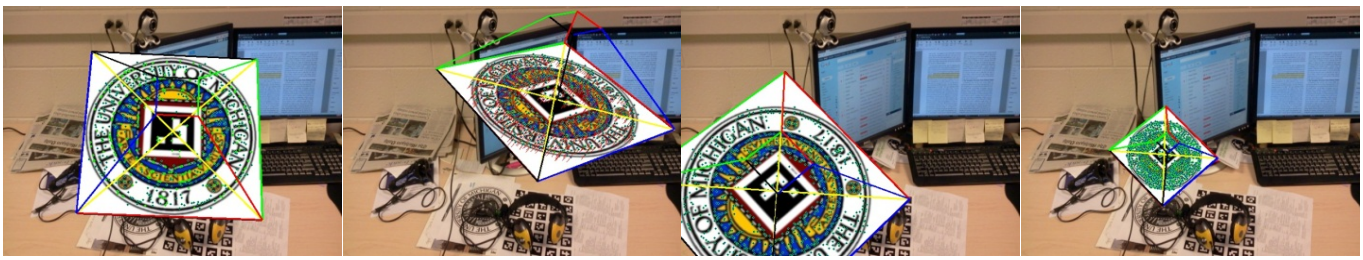


(b) Test case R-2.

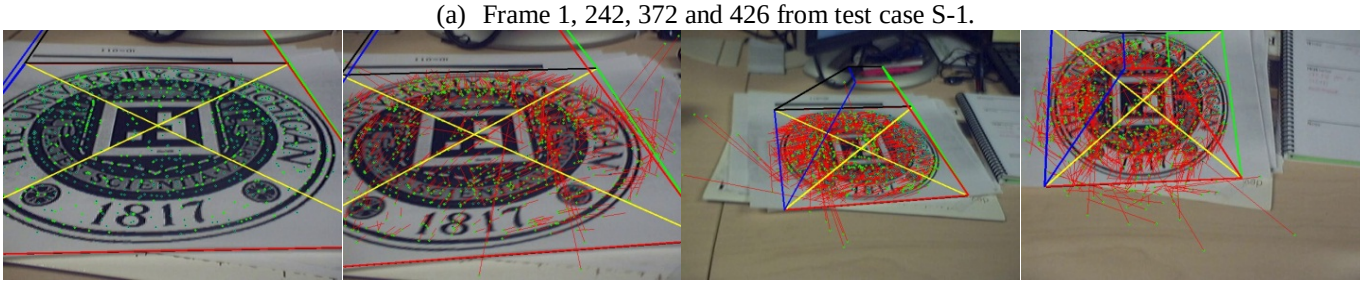


(c) Test case R-3.

Figure 11 NCC (left) curves and UOT (right) bars for real-world test cases (best viewed in color).



(a) Frame 1, 242, 372 and 426 from test case S-1.



(b) Frame 1, 20, 38 and 92 from test case R-2.

Figure 12 Visualization of A+KEG registration results of some representative frames.

6.2 Real-world Test Cases

We also recorded three real-world video sequences for a test. The first sequences (R-1) has 180 frames, the second (R-2) 122 frames, and the third (R-3) 254 frames. The R-3 test purposefully has a lot of shaking in the camera movements; this is to test tracking performance in a very challenging situation. Note that in real-world test cases, the ground truth of the camera pose is unknown, so T-RMS and R-RMS are not examined here. Another difference between real-world and synthesized test cases is that noises introduced by the webcam sensor in real-world test cases will affect the accuracy of tracking.

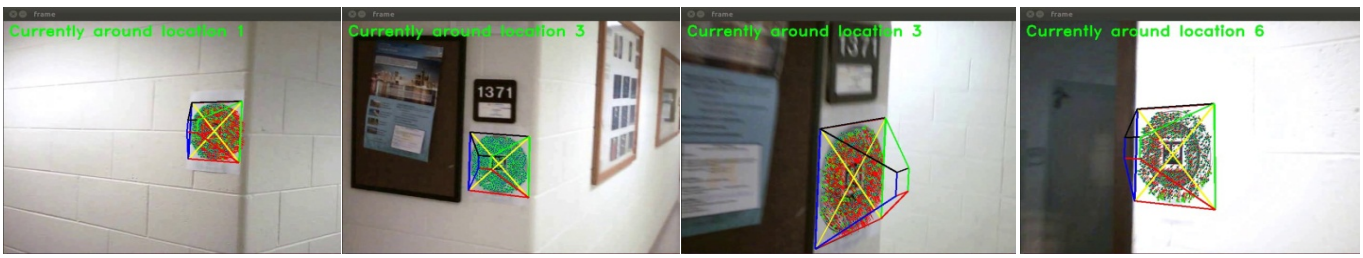
From Figure 11 one can see a similar performance analysis as in the synthesized test cases, which firmly proves our claim that the KEG method outperforms the state-of-the-art methods FERNs and AprilTag in speed, accuracy and stability. It's also worth noting that by comparing the algorithm configuration between A+KEG, A+KG, A+KE, and A+K, one can conclude that the three core steps of KEG are all crucial, and that the KEG method cannot achieve the same performance while missing any one of the three components. Another interesting observation is that the FERNs algorithm in real-world test cases is affected a lot by the noisy measurements. Comparing the left column of Figure 7 and Figure 10, it can be seen that the processing time per frame increased from 60 to 150 milliseconds on average, which drops its frame rate to only about 7 frames-per-second.

Figure 12 shows 4 representative frames under different challenging visual conditions such as large rotation or scale change (picked from test case S-1), or partial occlusion (picked from test case R-2), using the A+KEG method. The color pyramid represents the registration result, $\mathbf{R}$ and $\mathbf{T}$ of the camera relative to the marker image (since it is a 3D pyramid, without correct $\mathbf{R}$ and $\mathbf{T}$, it is impossible to be rendered correctly). The red trails show the position update from $\mathbf{x}_{old}$ to $\bar{\mathbf{x}}_{new}$.

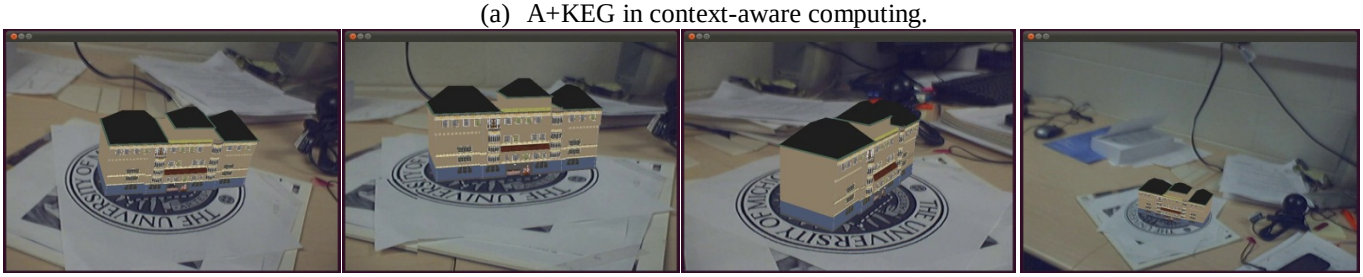
7 APPLICATIONS

As noted in the Introduction, this algorithm has several potential applications in many different areas. One example application we implemented is context-aware computing. Indoor context-aware computing has been studied in AEC for its ability to speed up information delivery in many aspects, including construction site inspection/monitoring and facility management (Aziz et al., 2005; Behzadan et al., 2008; Khoury and Kamat, 2009; May et al., 2005). Prior approaches for indoor ubiquitous tracking utilize an inertial measurement device, which suffers from its drifting effect. By (Akula et al., 2011), a context-aware computing system integrated with both GPS and inertial measurement device is developed, requiring human intelligence to recognize certain predefined locations to manually correct the drifting error caused by the inertial measurement device.

In our application, manual error correction was naturally replaced by automated correction using the A+KEG method, as shown in Figure 13a (see video in http://www.youtube.com/watch?v=Cnvr3l104wM). The green text shows that the algorithm successfully recognizes different locations by composing a natural photo (UM logo in this case) with different AprilTags. Once the inspector is within the effective range of the KEG marker image, the marker is automatically detected and then the inspector's pose relative to the marker is continuously estimated. Similar to (Akula et al., 2011), both the location and orientation of these predefined markers in the global coordinate system are known and stored in a database. Therefore the inspector's pose in the global coordinate system can be determined, as well. Benefiting from the



(a) A+KEG in context-aware computing.



(b) A+KEG in desktop AR.

Figure 13 Two example applications of the KEG tracker.

KEG tracker, this application can provide automatic regional drifting error correction instead of manual point correction. This application can thus further facilitate information delivery on construction sites or in indoor building environments.

Another interesting application is to apply this algorithm in tabletop augmented reality. Figure 13b (see video in http://www.youtube.com/watch?v=8Y8Mlh7jhsY) shows a desktop environment AR showcase of a 3D building design. Since the KEG tracker has the ability to quickly detect and accurately maintain the tracking of a marker image without requiring that the marker image be fully in sight (as required by ARToolkit), it can easily be adapted into tabletop collaborative AR applications (Dong and Kamat, 2011) to support better interactive design demonstration or visual simulation for construction planning.

8 CONCLUSIONS

After studying the two different types of natural marker-based registration algorithms and analyzing the cause of the drifting and jitter effects in both homography-from-tracking and homography-from-detection methods, a new natural marker-based registration algorithm framework, KEG tracker, is proposed, combining the advantages of those two, and circumventing their shortcomings. In theoretical analysis, we found out that the drifting effect is an error that occurs because of an accumulation problem. We solved that problem by applying two global constraints: a geometric and appearance constraint.

Our experiments on both synthesized and real-world test cases prove that the KEG method is fast enough for real-time applications (about 40 milliseconds for processing one 640×480 image), and also more accurate and stable than state-of-the-art algorithms such as FERNs and AprilTag. When the radius of the KEG marker is 20 cm printed out on a sheet of A4 paper, and the webcam provides an image resolution of 640×480 pixels, the KEG tracker's maximum working distance can be as far as 3 meters, and its maximum working Euler angle can be about 85 degrees offset from marker image's normal direction. If a larger working distance is desirable, a higher resolution camera and bigger marker can be deployed.

We also explored potential applications of the new tracker, such as context-aware computing, for replacing manually drifting error correction, and augmented reality for tabletop 3D visual simulation.

In the future, one direction for further research is applying more object recognition techniques so that, without the need of composing a fiducial marker (AprilTag), our method could more naturally support multiple marker recognition and tracking. Extending our method to a 3D environment, such that no planar structure assumption is needed, could also be a very interesting research direction. In addition, specific AEC applications of the tracker can be explored, such as pose estimation for construction equipment.

ACKNOWLEDGMENTS

This presented research was funded by the US National Science Foundation (NSF) via Grant CMMI-0927475. The writers gratefully acknowledge NSF's support. Any opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF or the University of Michigan.

REFERENCES

Akula, M., Dong, S., Kamat, V., Ojeda, L., Borrell, A., & Borenstein, J. (2011), Integration of infrastructure based positioning systems and inertial navigation for ubiquitous context-aware engineering applications, *Advanced Engineering Informatics*, pp. 640–655.

Aziz, Z., Anumba, C. J., Ruikar, D., Carrillo, P. M., & Bouchlaghem, D. N. (2005), Context aware information delivery for on-site construction operations, *Proceedings of 22nd CIB-W78 Conference on Information Technology in Construction*, pp.321–332.

Azuma, R. (1997), A survey of augmented reality, *Presence:Teleoperators and Virtual Environments, 6*(4), pp. 355–385.

Bao, S. Y., & Savarese, S. (2011), Semantic Structure from Motion, *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'11)*, pp. 2025–2032.

Bay, H., Tuytelaars, T., & Gool, L. v. (2008), SURF: speeded-up robust features, *Proceedings of 9th European Conference on Computer Vision (ECCV'08)*, 110, pp.346–359.

Behzadan, A., Aziz, Z., Anumba, C., & Kamat, V. (2008), Ubiquitous location tracking for context-specific information delivery on construction sites, *Automation in Construction, 17*(6), pp. 737–748.

Bekkali, A., Sanson, H., & Matsumoto, M. (2007), RFID indoor positioning based on probabilistic RFID map and Kalman filtering, *Proceedings of 3rd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMOB'07)*, p. 21.

Benhimane, S., & Malis, E. (2004), Real-time image-based tracking of planes using efficient second-order minimization, *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1, pp. 943–948.

Benhimane, S., Malis, E., Rives, P., & Azinheira, J. (2005), Vision-based control for car platooning using homography decomposition, *Proceedings of 2005 IEEE International Conference on Robotics and Automation (ICRA'05)*, pp. 2161–2166.

Chum, O., & Matas, J. (2005), Matching with PROSAC-progressive sample consensus, *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1, pp. 220–226.

Davison, A., Reid, I., Molton, N., & Stasse, O. (2007), MonoSLAM: Real-time single camera SLAM, *IEEE Transactions on Pattern Analysis and Machine Intelligence, 29*(6), pp. 1052–1067.

Dong, S., & Kamat, V. R. (2011), Collaborative Visualization of Simulated Processes Using Tabletop Fiducial Augmented Reality, *Proceedings of 2011 Winter Simulation Conference*, pp. 828-837.

Drummond, T., & Cipolla, R. (2002), Real-time visual tracking of complex structures, *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(7), pp. 932–946.

Fischler, M., & Bolles, R. (1981), Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM, 24*(6), pp. 381–395.

Harris, C., & Stephens, M. (1988), A combined corner and edge detector, *Proceedings of 4th Alvey Vision Conference*, 15, p. 50.

Hartley, R., & Zisserman, A. (2000). *Multiple view geometry in computer vision:* Cambridge University Press.

Jianbo, S., & Tomasi, C. (1994), Good features to track, *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pp. 593–600.

Kamat, V., & El-Tawil, S. (2007), Evaluation of augmented reality for rapid assessment of earthquake-induced building damage, *Journal of computing in civil engineering, 21*, pp. 303–310.

Kato, H., & Billinghurst, M. (1999), Marker tracking and hmd calibration for a video-based augmented reality conferencing system, *Proceedings of 2nd IEEE and ACM International Workshop on Augmented Reality*, pp. 85–94.

Khoury, H. M., & Kamat, V. (2007), WLAN Based User Position Tracking for Contextual Information Access in Indoor Construction Environments, *Proceedings of 2007 ASCE International Workshop on Computing in Civil Engineering*, pp. 838–845.

Khoury, H. M., & Kamat, V. R. (2009), Indoor User Localization for Context-Aware Information Retrieval in Construction Projects, *Automation in Construction, 18*(4), pp. 444–457.

Klein, G., & Murray, D. (2007), Parallel tracking and mapping for small AR workspaces, *Proceedings of 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, (ISMAR'07)*, pp. 225–234.

Ladikos, A., Benhimane, S., & Navab, N. (2007), A real-time tracking system combining template-based and feature-based approaches, *Proceedings of International Conference on Computer Vision Theory and Applications*, pp. 325–332.

Lepetit, V., & Fua, P. (2005), Monocular Model-Based 3D Tracking of Rigid Objects, *Foundations and Trends in Computer Graphics and Vision, 1*(1), pp. 1–89.

Li, B., Salter, J., Dempster, A., & Rizos, C. (2006), Indoor positioning techniques based on wireless LAN, *Proceedings of 1st IEEE International Conference on*

Wireless Broadband and Ultra Wideband Communications, pp. 13–16.

Lindeberg, T. (1998), Feature detection with automatic scale selection, *International Journal of Computer Vision, 30*(2), pp. 79–116.

Lowe, D. (2004), Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision, 60*(2), pp. 91–110.

Lucas, B., & Kanade, T. (1981), An iterative image registration technique with an application to stereo vision, *International joint conference on artificial intelligence*, 3, pp. 674–679.

Malis, E., & Vargas, M. (2007), Deeper understanding of the homography decomposition for vision-base control, *Research Report RR-6303, INRIA.*

May, A., Mitchell, V., Bowden, S., & Thorpe, T. (2005), Opportunities and challenges for location aware computing in the construction industry, *Proceedings of 7th international conference on Human computer interaction with mobile devices & services*, pp. 255–258.

Olson, E. (2011), AprilTag: A robust and flexible visual fiducial system, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'11)*, pp. 3400–3407.

Ozuysal, M., Fua, P., & Lepetit, V. (2007), Fast Keypoint Recognition in Ten Lines of Code, *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'07)*, pp. 1–8.

Rosten, E., & Drummond, T. (2006), Machine learning for high-speed corner detection, *Proceedings of European Conference on Computer Vision (ECCV'06)*, pp. 430–443.

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011), ORB: an efficient alternative to SIFT or SURF, *Proceedings of International Conference on Computer Vision (ICCV'11)*, pp. 2564–2571.

Simon, G., Fitzgibbon, A., & Zisserman, A. (2000), Markerless tracking using planar structures in the scene, *Proceedings of IEEE and ACM International Symposium on Augmented Reality (ISMAR'00)*, pp. 120–128.

Taylor, S., Rosten, E., & Drummond, T. (2009), Robust feature matching in 2.3 us, *Proceedings of 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'09)*, pp. 15–22.

Torr, P., & Zisserman, A. (2000), MLESAC: A new robust estimator with application to estimating image geometry, *Computer Vision and Image Understanding, 78*(1), pp. 138–156.

Wagner, D., Langlotz, T., & Schmalstieg, D. (2008), Robust and unobtrusive marker tracking on mobile phones, *Proceedings of 7th IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR'08)*, pp. 121–124.