A Plane Tracker for AEC-automation Applications

Chen Feng *, and Vineet R. Kamat

Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, USA

** Corresponding author (cforrest@umich.edu)*

Purpose 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). **Method** By studying two classic types of natural marker-based registration algorithms, homography-from-detection and homography-from-tracking, and 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. **Results & Discussion** 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.

***Keywords**: information technology, AR registration, tracking, context-aware computing*

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, addressed with different terminology. 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[1], 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 and wireless local area network (WLAN)-based indoor positioning. Yet these technologies are either costly 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.

Following this intuition, 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 of position and orientation, just as humans do. Firstly, the state-of-the-art methods in visual registration will be briefly introduced. Then among these methods, two important types of planar marker-based algorithms are discussed. The section following that explains the main contribution of this paper—an efficient improvement based on the previous two classes of algorithm frameworks, leading to the KEG tracker. Afterwards, several experiments are shown to demonstrate the superiority, under different objective quality measures, of KEG tracker. Also, two novel AEC applications that deploy KEG tracking algorithm is introduced.

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[2]: known environment vs. unknown environment. The first group of algorithms is less computation-consuming and easier to design since the only unknown is the user's pose. Because they have been well studied, and many related powerful algorithms have been proposed in the last

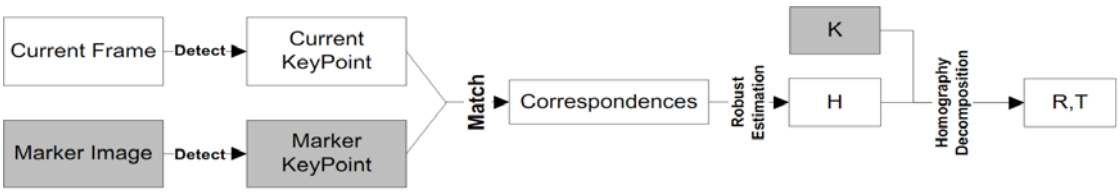


Fig. 1. Homography-from-detection algorithm framework.

two decades, it's 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. And the second group is more often applied in a controlled environment with limited space, such as a small manufacturing workspace.

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.

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"[2]. Usually those features are a set of black and white patterns forming simple geometry by circles, straight lines, or sharp corners and edges. Well known fiducial markers include ARToolKit[3] and the newly proposed AprilTag[4].

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, almost any common image, ranging from a natural view to a company logo, can immediately be used as a natural marker. This major difference makes it much easier and more natural 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.

Again, by the fundamental difference in the way they treat input images, natural marker-based methods form two groups: one group treats each input image independently, which is referred to as a detection-based method, such as[5,6]; the other group needs two or more consecutive images as input, which is referred to as a tracking-based method, such as[7,8]. Since our proposed method evolves from both these algorithm groups, in the following sections 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.

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, called as homography, maps points on the marker image to their corresponding points on the current image frame with the following equation:

$$s[x', y', 1]^T = H[x, y, 1]^T$$

where *H* is a 3x3 matrix representing the homography, (x, y) and (x', y') are the corresponding points on the two images, and *s* is an unknown scaling parameter.

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[9].

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.

Fig. 1 shows the generic algorithm framework of the homography-from-detection type of methods. The gray components need be loaded or calculated once

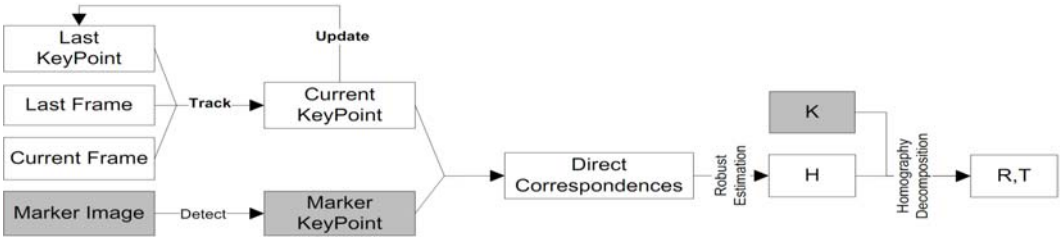


Fig. 2. Homography-from-tracking algorithm framework.

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.

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[5] is perhaps the most famous and widely used nowadays. Although the SIFT algorithm works very well under large variation of visual conditions, it is computation-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. FERNs[6], differs from SIFT 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 FERNs and other similar methods enjoy the high-speed advantage, their relatively low recognition rate make them less ideal in registration problem, as to be shown by our experiments.

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[10] (RANSAC) is usually employed to estimate the homography.

Once homography is found—through matrix decomposition techniques the camera position, the translation vector *T*, and orientation—the rotation matrix *R*, can be calculated. In our algorithm, a simple yet effective method[11] was adopted.

HOMOGRAPHY FROM TRACKING

As shown in Fig. 2 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. In fact, there are two such tracking algorithms that play crucial roles in our proposed method: the Kanade-Lucas-Tomasi (KLT) feature tracker[7] and Efficient Second-order Minimization (ESM) algorithm. 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. Then in order to find the optimal displacement, the algorithm formulates a least square problem to achieve a fast solution. While KLT's motion model being fairly simple, ESM[8] uses the 2D homography as motion model, and uses second-order approximation of the image function, thus leading to a global refinement algorithm with a faster convergence rate.

GLOBAL GEOMETRIC/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 Fig. 1, 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. In our experiments, even if the camera is fixed, the estimated camera position and orientation could have very large variance.

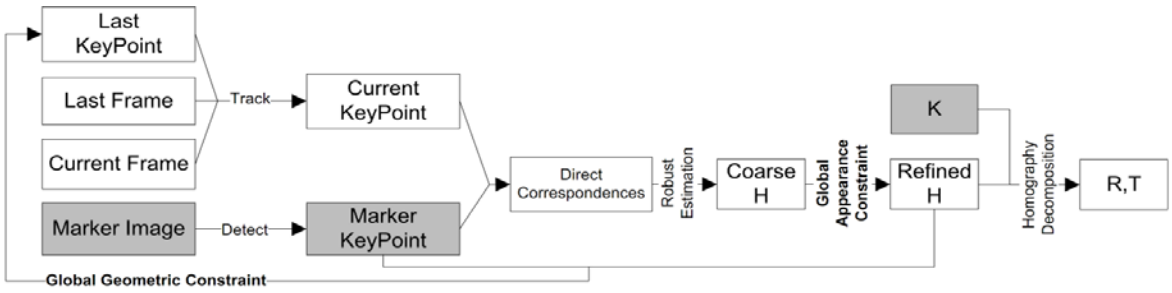


Fig. 3. KEG algorithm framework.

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 (Fig. 3) 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.

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 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:

$$\hat{\mathbf{x}}_{new} = \mathbf{x}_{old} + \Delta\mathbf{x} + \varepsilon_d + \varepsilon_g \quad (1)$$

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 RANSAC to estimate 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 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.

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 Fig. 3, which essentially means that the original

Algorithm 1: 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 (SIFT), try to find the marker image and its corresponding homography $H_{refined}$. 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 keypoints' positions $\{\mathbf{x}_{old}\}$, perform local tracking (KLT) and output new keypoints' positions $\{\mathbf{x}_{new}\}$.
4. Perform robust estimation (RANSAC) on correspondent keypoint array $\{\mathbf{x}_{ref}\}$ and $\{\mathbf{x}_{new}\}$, then output H_{coarse}.
5. Apply global appearance constraint by ESM and output $H_{refined}$.
6. Validate $H_{refined}$ by similarity (zero-mean normalized cross-correlation, equation (3)) threshold. If valid, $H_{refined}$ can be output for homography decomposition; otherwise, meaning loss-of-track, go to step 2.
7. Update positions of keypoints $\{\mathbf{x}_{new}\}$ by equation (2).
8. Replace I_{old} with I_{new}. Replace $\{\mathbf{x}_{old}\}$ with $\{\mathbf{x}_{new}\}$. Go to step 3.

marker image should look the same as the image rectified from the current frame by estimated homography *H*. Before this constraint, all of the information the KLT tracker used is local, while ε_d is systematic, therefore a global optimization based on the whole marker image will theoretically eliminate all the systematic error and H_{coarse} can serve as a good optimization starting point.

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

$$\bar{\mathbf{x}}_{new} = H_{refined} \cdot \mathbf{x}_{ref} \quad (2)$$

where $\mathbf{x}_{ref}$ is a keypoint's position 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 $H_{refined}$ is already theoretically error-free, updating using the above equation (2) instead of equation (1) 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 $|\varepsilon_d + \varepsilon_g|$. 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 1. 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 next section 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.

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 and similar performances were achieve on lower end computers. Also, all of the video sequences have a frame size of 640x480, 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, which originates from the java implementation by April Lab at the University of Michigan[4]. 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 implementation of the FERNs method. For the ESM method, we used the binary library provided by INRIA 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 marker image I_1 and the rectified image I_2 by $H_{refined}$, which can be calculated by:

$$NCC(I_1, I_2) = \frac{1}{n} \sum_{\mathbf{x}} \frac{[I_1(\mathbf{x}) - m_1]\,[I_2(\mathbf{x}) - m_2]}{\sigma_1 \sigma_2} \quad (3)$$

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_1. 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. 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:

$$T_{RMS} = \sqrt{\frac{1}{k} \sum_{i=1}^{k} \| T_i - \hat{T}_i \|^2}, , R_{RMS} = \sqrt{\frac{1}{k} \sum_{i=1}^{k} \| E_i - \hat{E}_i \|^2}$$

where k is the total number of frames of a test case, T_i and $\hat{T}_i$ are the i-th frame's ground truth and estimated position vector of dimension 3x1, and E_i and $\hat{E}_i$ are the i-th frame's ground truth and estimated Euler angle vector of dimension 3x1, 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

$$\begin{aligned} d_i &= NCC_{i+1} - NCC_i, \forall i = 1, 2, \cdots, k-1, \\ UOT &= \sqrt{\frac{1}{k-1} \sum_{i=1}^{k-1} (d_i - \mu)^2}, \; \mu = \frac{1}{k-1} \sum_{i=1}^{k-1} d_i, \end{aligned}$$

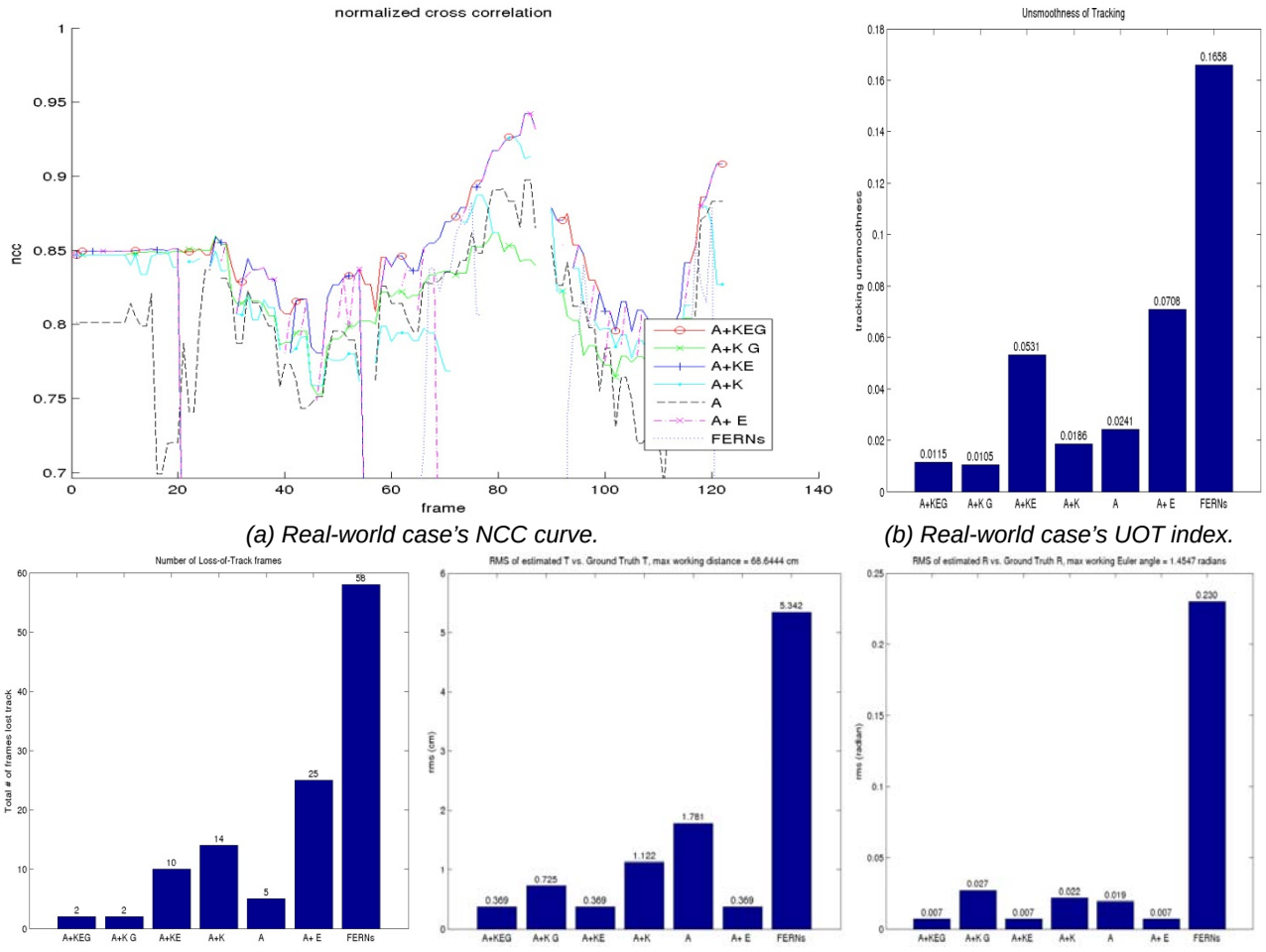
where NCC_i is the NCC index of the i-th frame, which essentially means that UOT index is the standard deviation of the difference between consecutive NCC indices. In MATLAB, this can be simply calculated by *"std(diff(ncc))."* A stable registration algorithm should give a UOT index value as small as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

In all of the test cases, our marker image is composed of a 16 bits AprilTag of ID equal to zero and a natural image, the logo of University of Michigan that is rich in features. The synthesized test case is rendered in OpenGL, using our marker image and a static real-world image as background. And the real-world test case is recorded using a Logitech webcam.

Experiments shows that the average duration per frame of A+KEG is about 40 milliseconds, which is even faster than AprilTag (60~70 milliseconds). 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 results prove that the KEG method does fit for real-time applications with process speed of 20 frames-per-second or faster.

Fig. 4 (a) shows that in most of the time, 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. This means KEG tracker has the highest quality of tracking.

Fig. 4 (b) and (c) demonstrate 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 Fig. 4 (d) and (e). 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) Real-world case's NCC curve. *(b) Real-world case's UOT index.*

(c) Real-world case's LOT index. *(d) Synthesized case's T-RMS index.* *(e) Synthesized case's R-RMS index.*

Fig. 4. Experiment results of comparison between different algorithms.

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[1,13]. Prior approaches for indoor ubiquitous tracking utilize an inertial measurement device, which suffers from its drifting effect. By Akula et al.[1], 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 IMU.

In our application, manual error correction was naturally replaced by automated correction using the A+KEG method, as shown in Fig. 5 left. 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.'s method, 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 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 lies in tabletop augmented reality. Fig. 5 right 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[12] to support better interactive design demonstration or visual simulation for construction planning. Note that our KEG algorithm is open-source, so the C++ codes and demonstration videos for the above two applications could be found at our lab's webpage (http://pathfinder.engin.umich.edu/).

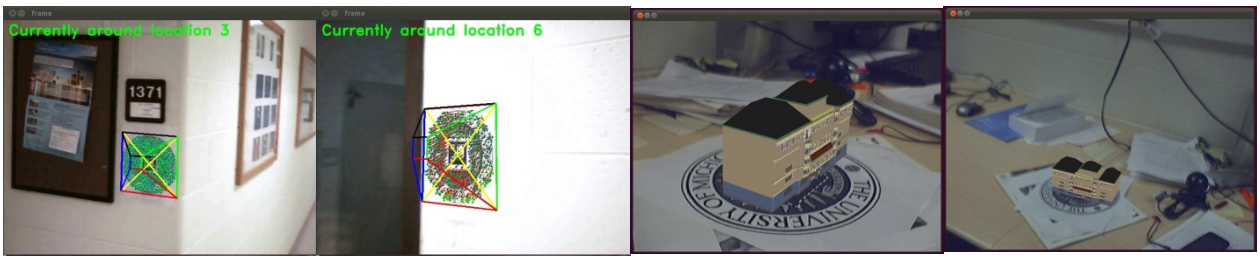


Fig. 5. A+KEG in context-aware computing (left) and table taletop AR (right).

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, KEG tracker, is proposed, combining the advantages of those two, and circumventing their shortcomings. In theoretical analysis, we found that the drifting effect is an error that occurs because of an accumulation problem. We solved the problem by applying two global constraints: a geometric and appearance constraint.

Experiments on both synthesized and real-world cases prove that the KEG method is fast enough for real-time applications, and also more accurate and stable than state-of-the-art algorithms such as FERNs and AprilTag, even when the marker image is partially occluded or viewed from very long distance.

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.

References

1. Akula, M., Dong, S., Kamat, V., Ojeda, L., Borrell, A., and Borenstein, J., "*Integration of infrastructure based positioning systems and inertial navigation for ubiquitous context-aware engineering applications*," Advanced Engineering Informatics, 2011.
2. Lepetit, V., and Fua, P., "*Monocular Model-Based 3D Tracking of Rigid Objects*," Foundations and Trends in Computer Graphics and Vision, 2005.
3. Kato, H., and Billinghurst, M., "*Marker tracking and hmd calibration for a video-based augmented reality conferencing system*," in Proceedings of 2nd IEEE and ACM International Workshop on Augmented Reality, 1999.
4. Olson, E., "*AprilTag: A robust and flexible visual fiducial system*," in *Proceedings of the International Conference on Robotics and Automation*, 2011.
5. Lowe, D., "*Distinctive image features from scale-invariant keypoints*," International journal of computer vision, Vol. 60(2), pp. 91-110, 2004.
6. Ozuysal, M., Fua, P., and Lepetit, V., "*Fast keypoint recognition in ten lines of code*," in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2007.
7. Lucas, B., and Kanade, T., "*An Iterative Image Registration Technique with an Application to Stereo Vision*," in Proceedings of the 7th International Joint Conference on Artificial Intelligence, 1981.
8. Benhimane, S., and Malis, E., "*Real-time image-based tracking of planes using efficient second-order minimization*," in Proceedings of International Conference on Intelligent Robots and Systems, 2004.
9. Hartley, R., and Zisserman, A., *Multiple view geometry in computer vision*, Cambridge University Press, 2000.
10. Fischler, M., and Bolles, R., "*Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*," Communications of the ACM, Vol. 24(6), pp. 381-395, 1981.
11. Simon, G., Fitzgibbon, A., and Zisserman, A., "*Markerless tracking using planar structures in the scene*," in Proceedings of IEEE and ACM International Symposium on Augmented Reality, 2000.
12. Dong, S., and Kamat, V., "*Collaborative Visualization of Simulated Processes Using Tabletop Fiducial Augmented Reality*," in Proceedings of Winter Simulation Conference, 2011.
13. Behzadan, A., Aziz, Z., Anumba, C., and Kamat, V., "*Ubiquitous location tracking for context-specific information delivery on construction sites*," Automation in Construction, Vol. 17(6), pp. 737-748, 2008.