Hybrid Feature Tracking and User Interaction for Markerless Augmented Reality

Taehee Lee*
Department of Computer Science
University of California, Los Angeles

Tobias Höllerer†
Four Eyes Laboratory, Department of Computer Science
University of California, Santa Barbara

ABSTRACT

We describe a novel markerless camera tracking approach and user interaction methodology for augmented reality (AR) on unprepared tabletop environments. We propose a real-time system architecture that combines two types of feature tracking methods. Distinctive image features of the scene are detected and tracked frame-to-frame by computing optical flow. In order to achieve real-time performance, multiple operations are processed in a multi-threaded manner for capturing a video frame, tracking features using optical flow, detecting distinctive invariant features, and rendering an output frame. We also introduce a user interaction for establishing a global coordinate system and for locating virtual objects in the AR environment. A user's bare hand is used for the user interface by estimating a camera pose relative to the user's outstretched hand. We evaluate the speed and accuracy of our hybrid feature tracking approach, and demonstrate a proof-of-concept application for enabling AR in unprepared tabletop environments using hands for interaction.

Keywords: position and orientation tracking technology, vision-based registration and tracking, interaction techniques for MR/AR

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality; I.4.8 [Image Processing and Computer Vision]: Scene Analysis

1 INTRODUCTION

In recent years, Augmented Reality (AR) research has focused on various sub-areas of interest. Highly developed techniques in Graphics and Virtual Reality now allow more realistic visualizations of AR overlays. Accurate tracking systems, input devices, and computer vision techniques improve the registration of images and a user's interaction in AR setups. However, most commonly used AR systems still either require sophisticated hardware tracking solutions, or at least fiducial markers.

Now that mobile computing devices have been widely deployed to customers, the interest on mobile AR is increasing. As the need for mobility is growing larger, devices are becoming smaller and easier to carry with a user. With the help of enhanced computing power, small devices now have enough capability to process complex visual computations. For example, a cellphone with a camera and a wide screen provides an inexpensive AR device for ordinary customers. Emerging wearable computers are also good platforms for enabling AR anywhere users may go. This paper introduces a method to bring inexpensive AR to unprepared environments.

This paper presents new work on markerless computer-vision-based tracking for arbitrary physical tabletop environments. It also showcases the use of this tracking technology for implementing the idea of an augmented desktop [16, 5] without the need for fiducial markers. 3D coordinate systems can be established on a tabletop surface at will, using a user's outstretched hand as a temporary initialization pattern. We build upon an existing framework for fingertip and hand pose tracking [20]. Users can establish a coordinate system for the tabletop AR environment simply by placing their outstretched hand onto the working plane surface.

We detect distinctive image features of the scene and track them using a hybrid tracking mechanism by combining distinctive features and optical flow. We use the distinctive features to recognize a scene, providing a scheme to resume a stabilized AR experience in the same or a different place. In addition, a user's hand is used as a user interface for placing virtual objects in the AR environment and for interacting with them.

1.1 Related Work

While there are many systems available to start AR in a prepared environment, we want to lower the barrier to initiating AR systems, so that users can easily experience AR anywhere [11]. Considering a mobile user entering a new workplace, such as an office desk environment, we would like to enable the user to start using AR without spending much effort on setting up the environment. Several marker-based AR systems [15, 7] have shown successful registration of virtual objects on top of cardboard markers. Markers also provide a tangible user interface that users can physically handle or place in the scene. However, markers inevitably occlude other existing objects in the environment, causing limited space for interactions, especially in cluttered desktop environments.

Interactive "tabletop" systems aim to provide an enhanced user experience for tasks carried out on a horizontal working plane, which can either be a conventional desk- or tabletop, or an interactive display surface. Tabletops based on various setup technologies have been proposed, including projector/camera combinations [30, 13], augmented reality using head-worn displays or projector/display combinations [25, 2, 16, 23], and interactive touchscreens of various kinds [4, 9]. Enabling AR for unprepared tabletop environment by employing tangible interaction with virtual and physical objects on the tabletop is very much at the heart of the concept.

With regard to tangible user interfaces, Handy AR [20] has shown that a user's bare hand can replace a cardboard marker [15, 7] for local coordinate estimation. Especially for highly mobile users, hands become the most easily available user interface device for mobile and wearable computers. Thus, we employ the Handy AR approach in our work for providing an initial camera pose estimation with scale information, and also as a user interface for interactions.

Markerless AR using natural features such as planes, edges, or corner points has been demonstrated for camera pose estimation [27, 6, 28]. With the assumption of a planar scene, computing a homography between frames provides an efficient tracking mechanism for registering augmentations with a scene [19]. In order to cope with general scenes, simultaneous localization and mapping (SLAM) in robotics has been employed to track the camera pose while building a 3D map of the scene [3]. Recently, [17] demonstrated a more robust markerless AR system by separating tracking

*e-mail: taehee@cs.ucla.edu
†e-mail: holl@cs.ucsb.edu

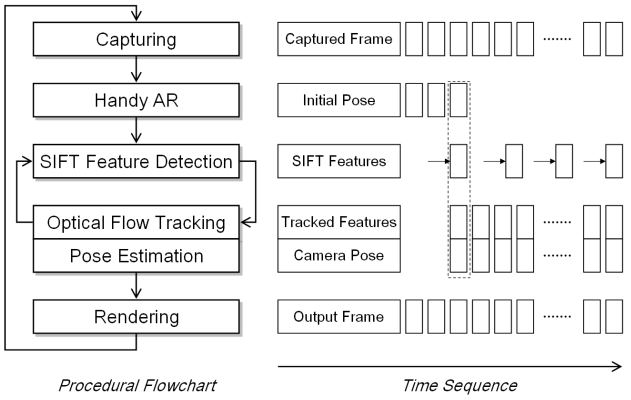


Figure 1: Procedural flowchart and conceptual time sequence for each tracked entity, establishing a coordinate system for a tabletop AR environment, using Handy AR, and our hybrid feature tracking method.

and map building tasks instead of SLAM. These methods, however, require manual calibration or a known-size object in order to provide initial metric scale information, which we want to minimize the cost for user interaction.

For fast and accurate tracking, optical-flow-based algorithms are widely used [12]. In order to arrive at stable tracking, landmark features are often used in order to recover from accumulated errors or complete tracking failures. While image patches [3] can be matched as templates to locate landmark features, they can be ambiguous in the presence of repetitive patterns, due to their fixed sizes. Recent result from [17] showed that using a large number of multiple-scale image patches is useful to robust camera tracking in a small AR environment. While more distinctive image features that are invariant to scale and orientation [22, 1] can be detected robustly for tracking over multiple frames, their complex computation makes the approach inappropriate for real-time applications that require around 30 frames per second(fps) [28]. For real-time performance that is at the same time robust and reliable in accuracy, a hybrid approach of combining these different types of tracking methods has been identified as a promising future research direction [21]. We are presenting such an approach within the application scope of tabletop working planes.

The rest of this paper is structured as follows: In Section 2, we describe how to combine hand posture recognition with hybrid feature tracking for establishing a coordinate system for tabletop AR environments. In Section 3, we show experimental results and evaluations and introduce a proof-of-concept application. In Section 4, we discuss the merits and limitations of the proposed approach. We present conclusions and future work in Section 5.

2 Method Description

We establish a coordinate system for an unprepared tabletop AR environment using fingertip tracking, and introduce a hybrid feature tracking method that combines landmark feature detection and optical flow-based feature tracking in a real-time approach. A camera pose is estimated from the tracked features relative to the 3D points that are extracted from the scene while establishing the coordinate system for the environment. An overall flowchart for this approach is illustrated in Figure 1. The established coordinate system is then propagated to the environment and the tracking region is expanded as we detect new features in the scene incrementally. We also use the hand tracking method for user interaction with virtual objects in AR.

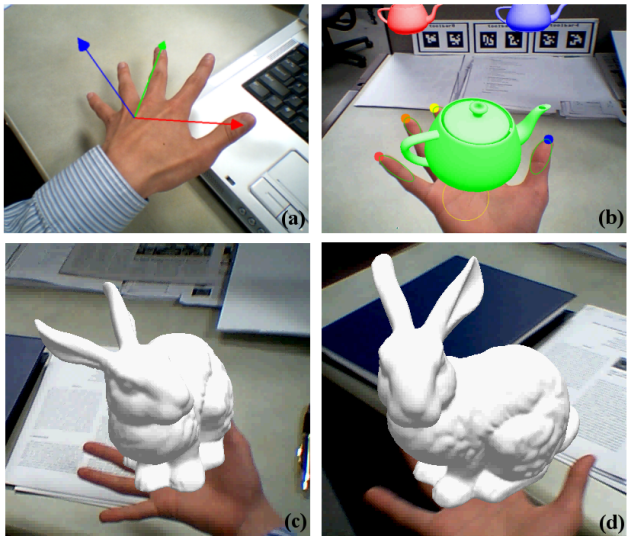


Figure 2: Snapshots of Handy AR: (a) the hand's coordinate system, (b) selecting and inspecting world-stabilized augmented objects, and (c),(d) inspecting a virtual object from various angles.

2.1 Handy AR Review

Using the "Handy AR" approach [20], we estimate a 6 degree-of-freedom camera pose, substituting a cardboard marker with a user's outstretched hand. The hand is segmented by a skin-color-based classifier [14] with an adaptively learned skin color histogram [18]. While the hand is tracked over time, fingertips are located on the contour of the hand by finding points that have high curvature values. The fingertips are accurately fitted to ellipses and are used for estimating a camera pose. Five tracked fingertips in a known pose (measured for each user as a once-in-a-lifetime off-line calibration step) provide more than the minimum number of point correspondences for a pose estimation algorithm [31]. The estimated camera pose is then used for rendering virtual objects on top of the hand as shown in Figure 2, and is used as an initial pose for further 3D scene acquisition and camera tracking as shown in Figure 3. In this way, users do not have to carry any tracking device or fiducial marker for AR with them.

2.2 Initializing a Coordinate System

Using the Handy AR system, we enable a user to initialize a coordinate system for augmented reality using the simple gesture of putting the hand on the desktop surface. While the hand rests on the surface plane, features around the hand are detected in order to establish the global coordinate system of a tabletop AR environment. Figure 3 shows the steps of propagating the coordinate system: The camera pose from the Handy AR is used to unproject the detected landmark image features to the plane, calculating their world coordinates. While tracking the features for each frame, the camera pose is estimated from feature point correspondences. Once the features in the scene are detected and the world coordinate system is initialized, the user may move the hand out of the scene and start interactions with the AR environment. Note that the area covered by the hand exhibits several features points as shown in Figure 3b. After moving the hand out of the view, as in Figure 3c, the features previously clinging to the hand are not detected any more. Instead, there is new space to detect more features that will be filled in over the next consecutive frames, which allows us to estimate the camera pose more robustly while expanding the tracking region.

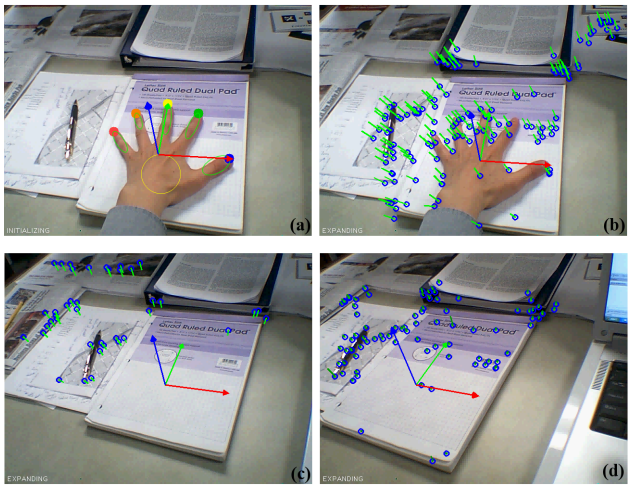


Figure 3: Establishing a coordinate system using Handy AR (a), propagating it to the scene as we detect scene features (b). After moving the hand out of the scene (c), new features are detected from the area that the hand occluded (d).

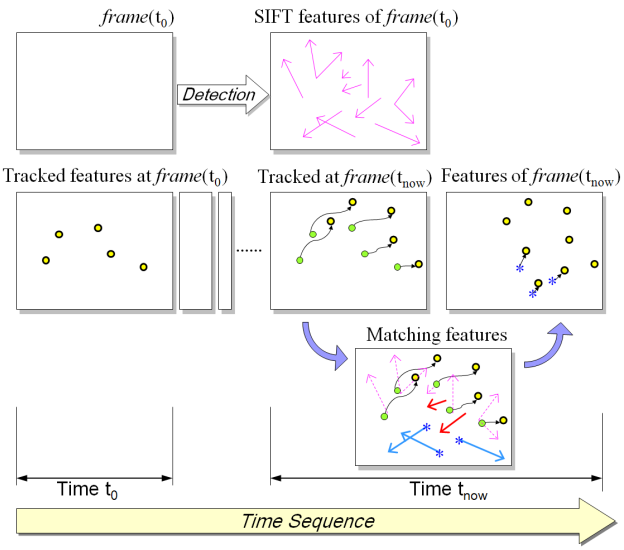


Figure 4: Illustration of tracking events in frame sequences for our hybrid feature tracking mechanism.

2.3 Distinctive Landmark Feature Detection

In order to detect distinctive landmark features in the AR environment, the scale invariant feature transform [22] is applied and keypoints are extracted from a captured video frame. The local extrema of the Difference-of-Gaussian (DoG) in the scale space are detected as keypoints and their descriptors are represented as histograms of orientations, providing scale and orientation invariant features. Hereafter, we refer these keypoints as SIFT features. SIFT features are shown to be useful both for drift-free tracking and for initially recognizing a previously stored scene [28].

Given a captured video frame, newly detected SIFT features are matched to a reference frame's features using an approximate Best-Bin-First (BBF) algorithm as in [22]. The point correspondences between the features in the captured frame and the reference frame are used for estimating a camera pose [31], based on the world coordinates of the SIFT features in the reference frame. This process of matching new SIFT features to previously stored features provides a simple track-by-detection method [21], which is used by our hybrid feature tracking mechanism that will be described in the next section.

When a user initiates a reference frame by putting a hand on the surface, the SIFT features are unprojected to a plane that is parallel to the hand. According to the estimated camera pose from Handy AR, the 3D locations of the features are computed in a new world coordinate system:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = P_{3\times 4} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{1}$$

where $(x\ y\ 1)^T$ and $(X\ Y\ Z\ 1)^T$ are homogeneous representations of the SIFT features in the image plane and the world coordinate system respectively, and $P_{3\times 4}$ is the projection matrix of the camera, whose fixed intrinsic parameters are assumed to be known through an initial calibration step. In our implementation, we assume the Z value to be a certain height ($Z = 0$ places the tabletop plane just below the hand), and derive equations to calculate the X and Y world coordinates of each SIFT feature from the (x, y) positions in the captured image.

In order to match the SIFT features more accurately, we use the RANSAC algorithm [8], eliminating outliers for pose estimation. RANSAC also removes features on vastly non-planar objects or orthogonal planes, as long as the majority of the visible features remain on the working plane. Thus, the planarity condition is maintained even in cases that consist of more complex geometry than a single working plane.

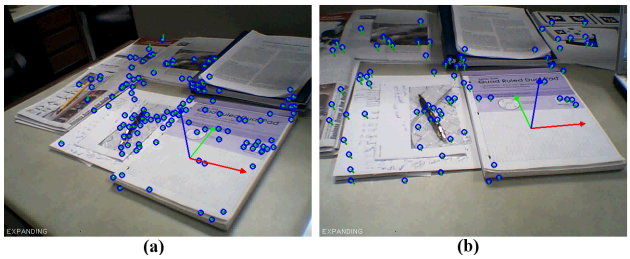
Since today's available computing power is insufficient to detect and match SIFT features for every frame in real time, we limit the search space and perform SIFT detection asynchronously in a separate thread. In our implementation, we reduce the scale of the search space by skipping the initial image doubling step (cf. section 3.3 in [22]). Computation time is reduced at the cost of rejecting small high-frequency features. The challenge to use SIFT detection as fast as the processing power allows is met by our multi-threaded framework running a video capturing thread and a SIFT processing thread separately as explained in Section 2.6.

2.4 Hybrid Feature Tracking

In order to track the detected features described in the previous section, an optical flow-based tracking algorithm is used for each pair of frames. The interest points in one frame are tracked with respect to the previous frame in 2D by iteratively computing fast optical flow using image pyramids [12]. Instead of the interest points detected by [26], our tracking uses distinctive landmark features (i.e. SIFT features). Therefore, the hybrid feature tracking is a combination of SIFT feature detection and optical flow-based tracking algorithms.

There are two general approaches to distinctive image feature tracking: a single-threaded approach and a multi-threaded approach. In [28], a single thread processes the detection and the tracking sequentially. However, a problem arises because of the relatively long computation time for SIFT feature detection (on the order of 10-20 real-time frames). As a result, the sequential approach stalls tracking significantly when applied in real-time applications.

Therefore, this paper proposes a multi-threaded approach for the hybrid feature tracking, in which separate threads are designated for detecting SIFT features and for tracking the features frame-to-frame via optical flow. New interest points are solely introduced by SIFT detection and the interest points to track are selected from the detected SIFT features in a multi-threaded manner. The problem that previously stalled the tracking during computation of the SIFT features is solved by the nature of multi-threading, sharing



(a) (b)

Figure 5: Expanding a tracking region while moving a camera to look at the scene from different perspectives.

time slices among the different tracking activities. However, another problem arises: the frame that SIFT features are found in lies in the past at the time the SIFT feature information becomes available (cf. Figure 4). Consequently, the frame for optical flow tracking needs to be matched with the past frame, in which SIFT features were detected.

We match the features between such frames as follows: The target frame for SIFT feature detection is saved as $frame(t_0)$ at time t_0, while fast optical-flow-based tracking is performed on subsequent frames until $frame(t)$ at time t. Suppose that at time t_{now} (typically half a second later) the SIFT detection is finished. At this moment, we have a set of freshly detected SIFT features from $frame(t_0)$ and another set of features that have been tracked by optical flow from t_0 until t_{now}. The SIFT features are matched to the previous positions of tracked features, illustrated as dotted arrows and curved trails of features in Figure 4, respectively. Additionally, new interest points from unmatched SIFT features are added to $frame(t_0)$ if they are more than a threshold distance (i.e. 10 pixel distance in our implementation) apart from any other currently tracked feature, illustrated as '$*$' in the figure. Then the newly added features of $frame(t_0)$ are tracked to $frame(t_{now})$ by computing optical flow directly between the two frames. In this way, newly detected SIFT features are matched to the tracked features.

During these matching and tracking procedures, features could have been dropped due to unsuccessful optical flow tracking and the RANSAC algorithm during pose estimation as described in the previous section. Discarding such outliers helps the tracking system to increase its accuracy of camera pose estimation.

This hybrid feature tracking algorithm provides a way of estimating camera pose effectively and robustly in real time. In our implementation, the SIFT detection thread runs at about 2.1 fps, and optical flow at 26.3 fps when tracking and expanding a region. A more detailed analysis of our approach is presented in Section 3 and discussed in Section 4.

2.5 Expanding a Tracking Region

When a coordinate system is established in a specific reference frame, scene tracking would ordinarily be limited to the overall field of view covered by that specific captured video frame. In addition to the limited field of view, the initial frame contains the hand image, as shown in Figure 3b. Hence, we want to introduce new features dynamically as we move the hand away or shift physical objects around on the workspace. In order to increase the robustness of tracking and the coverage area for a tabletop workspace, it is necessary to expand a tracking region incrementally, covering large viewing angles from different perspectives. This is also helpful for reliable feature detection and tracking, although the distinctive image features [22] are already invariant to some changes in rotation and scale.

Given an estimated camera pose, newly detected distinctive features are unprojected to a plane according to Equation (1) in Section 2.3. They are added to the set of features that represents the

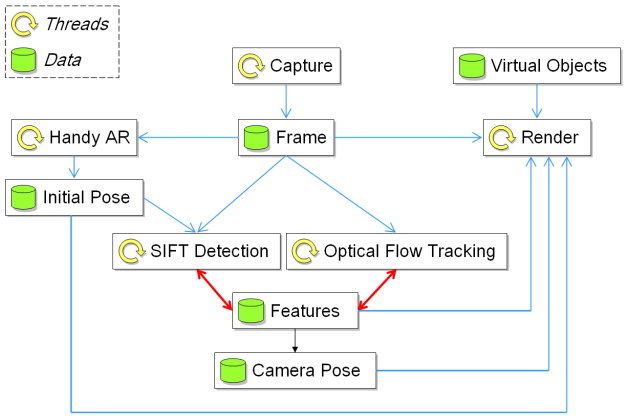


Figure 6: Diagram of dependency among threads and data. Since the SIFT detection thread and the Optical Flow Tracking thread critically share the feature data, synchronization is required.

scene for camera tracking, expanding the covered region. As shown in Figure 5, SIFT features are detected at multiple scales and orientations from different perspectives, increasing robustness of our hybrid feature tracking considerably, compared to methods relying solely on optical flow. After a user establishes a coordinate system using one of their hands as described in the previous section, the hand-covered region exhibits no features due to previous occlusion (Figure 3c). Also, new viewing angles may introduce new areas of the tabletop that were previously unexposed. By detecting new features in such areas, as shown in Figure 5, we increase robustness and enable users to interact on a larger workspace than the original reference frame, providing a useful, growing tabletop AR environment.

2.6 Multi-threaded Approach

In order to provide real-time performance, we process operations in a multi-threaded manner. We divide the necessary tasks in our AR framework into functional groups: capturing a video frame, performing Handy AR, detecting SIFT features, tracking the features using optical flow and performing pose estimation, and rendering an output frame. Separate threads are allocated to each group, as indicated by rows in Figure 1.

Since these threads share some common data, such as a frame image, a set of SIFT features, and camera pose parameters, as shown in Figure 6, we must synchronize some operations within these threads. Among all threads, the SIFT detection thread and the fast feature tracking thread have the strongest need for synchronization, because they are to combine the detected and tracked features seamlessly. Since the SIFT detection procedure takes about ten times as long as optical flow tracking, we detect features *asynchronously*, store them in the SIFT thread's local variables and just synchronize the two threads at the time of feature matching.

Since a captured video frame is used by every thread, we synchronize the frame data. However, declaring access to a frame to be a critical section would prevent the capturing thread from proceeding quickly in a non-blocking way. Instead, we store a timestamp with each captured frame, providing the threads with information to distinguish frames, implementing *soft* synchronization. For example, the SIFT detection thread and the optical flow tracking thread communicate with each other on which frame is ready for feature detection by timestamp comparison. The last known good SIFT detector frame is stored with its timestamp until it is updated by the SIFT thread.

The rendering thread may be synchronized with other threads depending on the data it displays. For example, if features are to

Table 1: Average frame rates of each thread (fps)

	Interaction Steps			
Threads	Establish	Expand	Select	Place
Capture	46.8	33.2	47.1	17.3
Render	37.7	28.0	37.9	18.6
HandyAR	26.5	Idle	30.8	
SIFT	2.1			
Tracking	Idle	26.3	Idle	13.8

be displayed (for debug purposes, as in our video), the rendering thread also needs to be synchronized for feature data access. The behavior of our multi-threaded framework is examined in Section 3 (see Figure 7). In addition to synchronizing the threads, we also pay attention to adjusting their execution priorities for tuned performance. Higher priority for the capturing thread allows for low-latency playback. Yielding to other threads is crucial to maintain a regular appropriate frame rate for each thread.

2.7 Recognizing a Scene

As we described in Section 2.3, distinctive features are detected in the scene while a coordinate system for a new tabletop environment is established.

These features are also used for recognizing different previously stored scenes, for resuming tracking after the user looked away from the scene temporarily, and for recognizing a scene from various viewing angles.

We match the detected SIFT features between an active frame and previously stored features, following the object recognition algorithm of [22]. The scene that has the maximum number of matched features among all scenes is recognized to be equivalent to the current scene when the number of matched features exceeds a certain threshold. This recognition step is performed when a user looks at a new place so that the system fails to detect enough features from the previously tracked scene. In our implementation, the scene recognition time increases proportionally to the number of saved scenes.

Once a scene is recognized, the AR space can be appropriately set up according to a user's purpose: Augmentations can be fixed to the environment so that a user experiences different AR spaces in different physical environments, or users can carry "their AR desktop" with them to any new tabletop environment to resume previous interactions. In the latter case, they may have to adjust the positions of virtual objects if those coincide with locations of physical objects. The hand-based user interface methodology for such adjustments is presented in Section 3.2.

3 Results

We tested our implementation of hybrid feature tracking with regard to the speed and robustness of the system. We especially examined the multi-threaded framework's behavior with respect to real-time performance. Accuracy was tested by monitoring the achieved registration for example AR applications, placing and stabilizing virtual objects within a tabletop scene. The results show that our method is useful for easily establishing a coordinate system for tabletop environments and is robust against drift errors and able to swiftly recover from tracking failures. Our test system uses a smaller SIFT search space than existing implementations [10, 22], and it runs our hybrid feature tracking method using the described multi-threaded framework for real-time AR.

The experiments were performed on a small laptop computer with a 1.8GHz CPU, using a USB 2.0 camera with 640×480 resolution. For Handy AR, the internal computations were processed

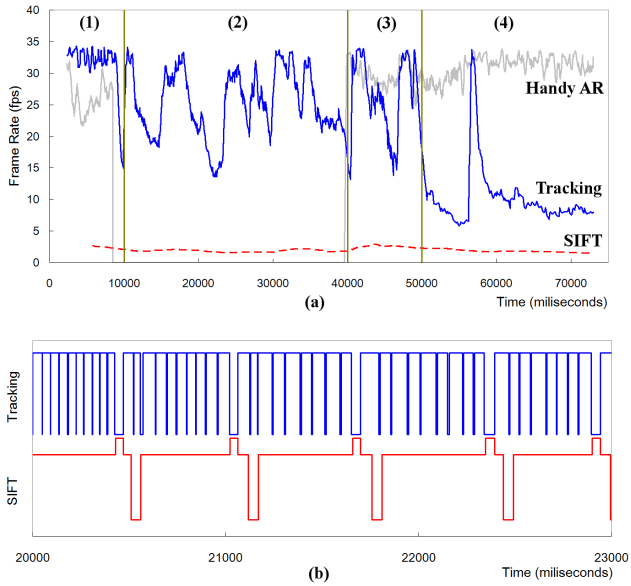


Figure 7: (a) Frame rates of threads processing Handy AR (gray), SIFT detection (dashed red), and feature tracking (blue) over time. (b) SIFT detection (red) and feature tracking (blue) threads are partially synchronized.

in 320×240 resolution for faster speed while still maintaining reasonable accuracy [20]. Intrinsic camera parameters were calibrated in a one-time offline step using a 6×8 checkerboard pattern and the OpenCV [12] implementation of [31].

3.1 Evaluations

Evaluations for our system were performed with regard to the speed of the system and robustness of the hybrid feature tracking algorithm. As for the speed, the goal was to achieve real-time performance (at least 15 frames per second for interactive systems), and to strive for 30fps. Table 1 lists the average frame rates of each thread for the different interaction steps of a typical tabletop AR session: establishing a coordinate system, expanding a tracking region, selecting a virtual object, and placing it to the AR environment. Overall, threads run at around real-time frame rate during the establishing, expanding, and selecting phases. The capturing thread and the rendering thread run at around real-time speed, providing fast feedback. The Handy AR system runs at 26.5fps at the beginning stage of establishing a coordinate system and at 30.8fps while interacting with virtual objects stabilized in the environment. After propagating the coordinate system to the environment, the hybrid feature tracking threads (SIFT detection and optical flow tracking) provide real-time pose estimation updates at around 26.3fps. Because the rendering thread is running at a similarly fast speed, the user perceives fast and smooth registration of augmentations. When a user is placing a selected virtual object into an AR environment, the frame rate drops because of increased occlusion and need for synchronization in the scene. However, every thread still runs at interactive frame rates (i.e. around 15fps), including both Handy AR and feature tracking.

We measured the behavior of multiple threads running as a single framework as shown in Figure 7a, while a user performed the same sequence of actions mentioned above; over time: (1) establishing a coordinate system (initial 10 seconds), then (2) expanding the tracking region (30 seconds), (3) selecting an object as looking away from the tracked region (10 seconds), and (4) moving objects into the environment (25 seconds). The SIFT detection thread runs

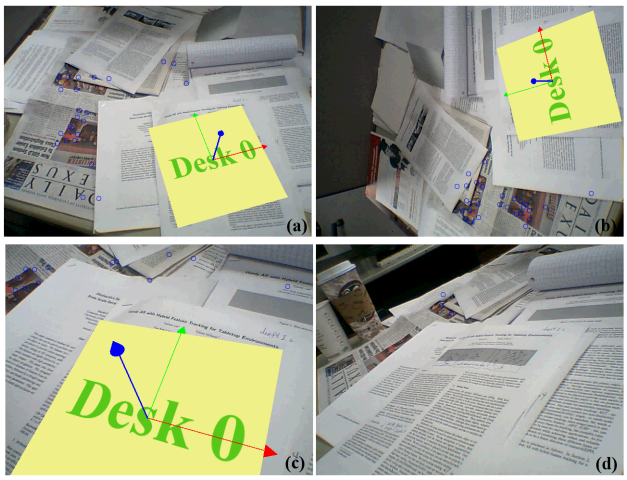


Figure 8: Various angles (a)(b)(c) of camera pose, estimated from the features in the scene. (d) Exceeding a certain viewing angle, features are no longer detected.

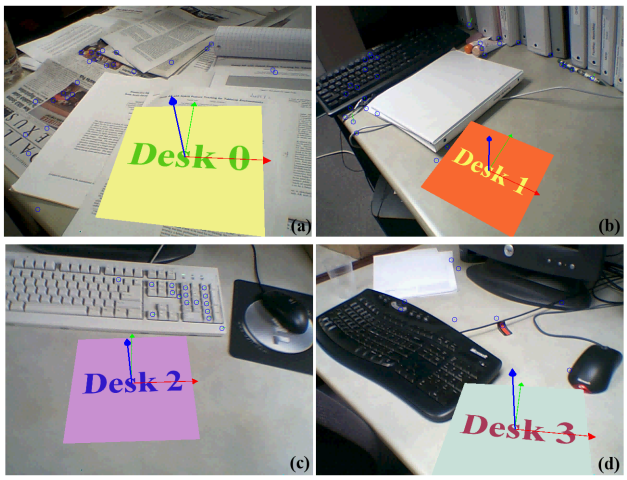


Figure 9: Snapshots of recognizing a scene. Each desk is indicated with an annotation.

as a background process, showing a steady frame rate at around 2.1fps. The Handy AR and the feature tracking threads become idle when they are not required to be processed as shown in Table 1, and run at real-time speed when one of them is running (for the expansion or selection step). When a user places a virtual object to the environment, Handy AR maintains real-time speed, while feature tracking runs at interactive speed.

We also measured the behavior of the SIFT detection thread and the feature tracking thread as shown in Figure 7b. The status of each thread is plotted indicating whether it is idle (bottom level), working asynchronously (middle level), or working synchronized (top level). The two threads are synchronized when the SIFT detection thread adds new features to global variables. This synchronization shows in the plot, as the highest steps of the SIFT thread's plot and the lowest steps of the feature thread's plot meet periodically in Figure 7b.

We examined the robustness of the hybrid feature tracking approach while moving a camera in the scene in several ways, exhibiting stable location and orientation, slightly shaky or jittery motion, smooth movement with constant speed, abrupt quick motion, constant fast motion, looking at a completely new space, and returning back to the original space. The tracking method showed successful feature detection in the scene and tracking was mostly continuous. Under fast motion, the system loses track of the features, but quickly recovers to track them whenever the distinctive features are re-detected. With our tabletop AR environment scenarios, this robust recovery is very useful to register augmentations again whenever the tracking is lost. Dead reckoning could be used to avoid disruption of AR annotations during frames that cause tracking failure.

The performance of camera pose estimation was tested while looking at the scene from various angles. As shown in Figure 8, the viewing angle may vary drastically from the initial camera pose, thanks to the distinctive features. Under shallow angles like Figure 8d, however, the SIFT features are not easily detected because they are not perspectively invariant. In the case of tracking the features over time, those viewing angles will still be covered by using the tracked features and newly detected features while expanding the tracking region as described in Section 2.5. Only when the user arrives at such angles through rapid motions will tracking fail until stored SIFT features are re-discovered.

3.2 Interaction UI and Application Prototype

We have implemented a proof-of-concept application for tabletop augmented reality environments using Handy AR and our hybrid feature tracking method. A user establishes coordinate systems for several desks separately, as an initial step for each space. The distinctive features are stored persistently, noting their world coordinates. When the user sits at one of the desks and starts the AR application, the features are detected and matched to recognize the scene the user is viewing. Figure 9 shows that the scene is recognized successfully and the annotations are registered according to the coordinate system that the user established previously. Through the same framework that recognizes each space, the system can also be used for just setting up one AR coordinate system, so that a user can bring an individual AR environment to several locations. This enables users to personalize their tabletop AR system while moving into new locations, as long as their coordinate systems are set up once with Handy AR on the new tabletop environment.

When a coordinate system is established for a tabletop AR environment, a user may place virtual objects in the workspace. By using the Handy AR system, a user can select an object and then place it into the established coordinate system. In absence of a trigger event simulating a mouse button, we cycle through a set of selectable objects when a Handy AR pose is detected in the initial (base interaction) mode. Objects are rendered on top of the hand in a rotating sequence of 1-second intervals. Breaking Handy AR tracking (e.g. by making a fist for "grabbing" the displayed object) selects the currently displayed object, and the interaction mode switches to placement mode. Objects can be placed using either hand, so it is entirely possible to select an object with your non-dominant hand and place it into the scene with your dominant hand since Handy AR detects either one.

As shown in Figure 10a and 10b, when a camera pose is estimated from both the user's hand and hybrid feature tracking in placement mode, the object's transform matrix in the world coordinate system is computed by multiplying the *hand-to-camera* and the *camera-to-world* transform matrices so that the hand's transform is used to place the object in the environment's coordinate system; The *hand-to-camera* matrix is retrieved from the camera pose estimated by Handy AR, and the *camera-to-world* is the inverse of the camera pose from our markerless tracking system. When a hand gesture is triggered to place the object (i.e. again making a fist to break fingertip tracking), the transform of the object is stored to keep it stabilized in the environment, as shown in Figure 10a

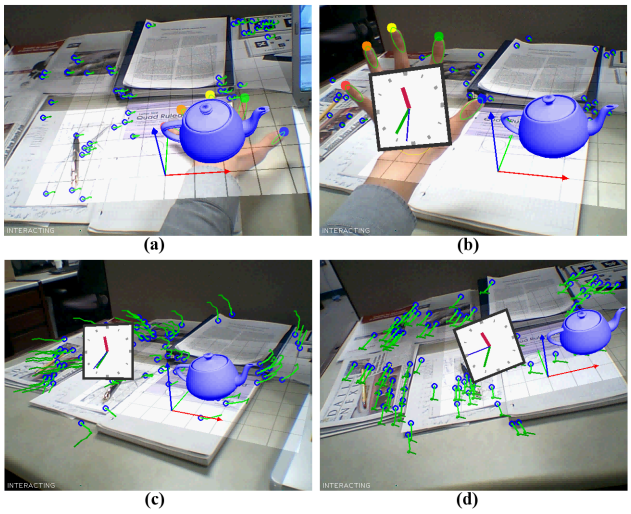


Figure 10: Snapshots of interacting with the tabletop AR environment. Handy AR is used for placing virtual objects: (a) teapot and (b) clock. (c)&(d) The tabletop AR environment is tracked with its virtual objects stabilized.

and 10b. Then, the stabilized objects can be viewed from different angles as shown in Figure 10c and 10d. Note that the trails of features are drawn as green lines in these figures for debug purposes only.

4 Discussion

Since Handy AR is using a skin-color-based hand segmentation algorithm, other objects that have similar color will be also detected as candidate regions for initial detection. Once the hand is detected, blob tracking ensures that it isn't lost even in presence of other skin-color objects unless there is considerable overlap. In case of initial hand detection failure, the user still needs to be able to let the system work properly. A simple workaround is to use a masking object of different color (e.g. a sheet of paper) as a temporary background during detection. After setting up the AR coordinate system by detecting features in the scene, the user may remove the hand and also the masking object. Since the features in or around the hand will not be used as further features to track as shown in Figure 3c, the subsequent camera tracking can detect new features in the scene, expanding the tracking region appropriately.

Our current method for propagating the established coordinate system makes the assumption that the workspace is a flat surface, providing at least a dominant plane. The first reference frame should provide reasonable registration for tabletop environments and the tracking region is further expanded while detecting more features over a larger tabletop space. Since common objects that lie on tables like sheets of paper, books, notes or pens are often flat or of low height, this planar assumption is effective. The features on non-planar objects will be discarded when applying the RANSAC algorithm to get rid of outliers for estimating a camera pose based on the planar assumption.

More sophisticated algorithms could enhance the camera pose tracking accuracy and relieve the planar assumption. Moving a camera around the scene while establishing the AR environment, the structure of the 3D scene can be reconstructed, either on-line or off-line. A practical on-line approach of building a 3D landmark map is presented by [3], where a probabilistic framework enables a reliable single camera tracking system. In such an approach, our hybrid feature tracking algorithm can be used for detecting and tracking landmarks. On the other hand, off-line approaches [24] usually process the input data as a batch job. Although these off-line approaches are not feasible yet to be used in real-time applications, combining on-line and off-line algorithms seems to be beneficial to enhance AR tracking systems.

Our multi-threading approach enables the system to balance the work load among the different tasks while running them simultaneously, letting the operating system control the threads to share the available processing power. In this way, our hybrid feature tracking mechanism works smoothly with a heavy-weight SIFT detection thread as a likely background process, assisting the light-weight fast tracking method to be more robust to drift. This idea is similar to combining multiple sensor modalities in a system, such as inertial sensors, GPS, optical devices, and vision-based systems. A similar approach of fusing different data sources, presented in the more formal framework of an Extended Kalman Filter [29], can be used for processing multiple tracking sources. However, we need to pay more attention on designing such systems running with multiple threads, because sharing data among several threads makes it hard to predict complex runtime behavior.

As for the behavior of the hybrid feature tracking, the mechanism described in the Section 2.4 works well not only for continuous tracking, but also for the starting state and the recovering state. In the starting state, there are no features to track yet. Therefore, all detected SIFT features are added as interest points to track. However, because of the direct optical flow matching between $frame(t_0)$ and $frame(t_{now})$ as described in Section 2.4, the features could be dropped according to large difference between the views. Thus, the camera should stay relatively stable in the beginning stage in order to successfully start to track features. Similarly, after feature tracking is completely lost, the recovery to resume tracking works in the same fashion as the starting state. The condition that the camera should not move much during the beginning or resuming period is easily satisfied in common user interaction situations. For example, people tend to stay focused on target objects when they start interactions with them, therefore naturally keeping the camera stable enough to begin tracking.

One limitation of tracking natural features in the environment is that the scene is considered to be stationary. In tabletop systems, however, users may move objects around while interacting with them. In such cases, the pose estimation may be less accurate when the features on these moved objects are taken into account for feature tracking (instead of being discarded by RANSAC). It may break down completely when the majority of surface texture is changed (e.g. by shuffling papers around). This problem can be addressed by adding new features constantly while interacting with the tabletop AR system. This is in contrast to our implementation, which simply expands the tracked space, adding new features more sparsely. If the displacement of objects are minor changes in the environment (e.g. removing a book), the major remaining features will be used to track the camera pose.

5 Conclusions and Future Work

We have introduced a method for 6 degree-of-freedom camera tracking that includes a hybrid approach detecting distinctive image features and tracking them with optical flow computations. The markerless tracking is initialized by a simple hand gesture using the Handy AR system, which estimates a camera pose from a user's outstretched hand. The established coordinate system is propagated to the scene, and then the tabletop workspace for AR is continuously expanded to cover a larger area than a single reference frame. The system uses distinctive image features in order to recognize the scene and to correct for accumulated tracking errors. Our proposed hybrid tracking approach proves to be useful for real-time camera pose estimation without using fiducial markers in a tabletop AR environment.

For future work, we want to tackle full 3D scene reconstruction.

It is a challenging topic, which receives significant attention in the computer vision and robotics communities, with approaches such as structure from motion (SfM) or simultaneous localization and mapping (SLAM). The real-time constraint and user interaction required by AR systems make the problem even more interesting. As for more robust and reliable tracking, we are experimenting with several different types of features using edges and partial models.

ACKNOWLEDGEMENTS

This research was supported in part by the Korea Science and Engineering Foundation Grant (#2005-215-D00316), and by a research contract with the Korea Institute of Science and Technology (KIST) through the Tangible Space Initiative project.

REFERENCES

[1] H. Bay, T. Tuytelaars, and L. J. V. Gool. SURF: Speeded up robust features. In *ECCV (1)*, pages 404–417, 2006.

[2] A. Butz, T. Höllerer, S. Feiner, B. MacIntyre, and C. Beshers. Enveloping users and computers in a collaborative 3D augmented reality. In *Proc. IWAR '99 (Int. Workshop on Augmented Reality)*, pages 35–44, San Francisco, CA, Oct. 20–21 1999.

[3] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.

[4] P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM Press.

[5] S. DiVerdi, D. Nurmi, and T. Höllerer. ARWin - a desktop augmented reality window manager. In *ISMAR*, pages 298–299, October 2003.

[6] V. Ferrari, T. Tuytelaars, and L. J. V. Gool. Markerless augmented reality with a real-time affine region tracker. In *ISAR*, pages 87–96. IEEE Computer Society, 2001.

[7] M. Fiala. ARTag, a fiducial marker system using digital techniques. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 590–596. IEEE Computer Society, 2005.

[8] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[9] J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM Press.

[10] R. Hess. SIFT detector. http://web.engr.oregonstate.edu/ hess/ June, 2007.

[11] T. Höllerer, J. Wither, and S. DiVerdi. *Anywhere Augmentation: Towards Mobile Augmented Reality in Unprepared Environments*. Lecture Notes in Geoinformation and Cartography. Springer Verlag, 2007.

[12] Intel Corporation. Open Source Computer Vision Library reference manual. December 2000.

[13] H. Ishii and B. Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proc. Conference on Human Factors in Computing Systems (CHI '97), (Atlanta, March 1997)*, pages 234–241. ACM Press, 1997.

[14] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. In *CVPR*, pages 1274–1280. IEEE Computer Society, 1999.

[15] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, Oct. 1999.

[16] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana. Virtual object manipulation on a table-top AR environment. In *Proceedings of the International Symposium on Augmented Reality ISAR 2000*, pages 111–119. IEEE Computer Society Press, Oct. 5–6 2000.

[17] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[18] M. Kölsch and M. Turk. Fast 2D hand tracking with flocks of features and multi-cue integration. In *IEEE Workshop on Real-Time Vision for Human-Computer Interaction*, page 158, 2004.

[19] T. Lee and T. Höllerer. Viewpoint stabilization for live collaborative video augmentations. In *IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 241–242, 2006.

[20] T. Lee and T. Höllerer. Handy AR: Markerless inspection of augmented reality objects using fingertip tracking. In *International Symposium on Wearable Computers*, October 2007.

[21] V. Lepetit and P. Fua. Monocular model-based 3d tracking of rigid objects. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2006.

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

[23] M. L. Maher and M. J. Kim. Studying designers using a tabletop system for 3d design with a focus on the impact on spatial cognition. In *TABLETOP '06: Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 105–112, Washington, DC, USA, 2006. IEEE Computer Society.

[24] M. Pollefeys, R. Koch, and L. J. V. Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *ICCV*, pages 90–95, 1998.

[25] J. Rekimoto and M. Saitoh. Augmented surfaces: A spatially continuous work space for hybrid computing environments. In *Proc. ACM CHI '99*, Pittsburgh, PA, May 15–20 1999. ACM Press.

[26] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[27] G. Simon, A. W. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *IEEE/ACM International Symposium on Augmented Reality*, pages 120–128, 2000.

[28] I. Skrypnyk and D. G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *ISMAR*, pages 110–119. IEEE Computer Society, 2004.

[29] G. Welch and G. Bishop. An introduction to the kalman filter. In *Technical Report*. University of North Carolina at Chapel Hill, 1995.

[30] P. Wellner. Interacting with paper on the digitaldesk. *Commun. ACM*, 36(7):87–96, 1993.

[31] Z. Y. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov. 2000.