Relocalization Using Virtual Keyframes For Online Environment Map Construction

Sehwan Kim* Christopher Coffin† Tobias Höllerer‡
University of California, Santa Barbara

Abstract

The acquisition of surround-view panoramas using a single hand-held or head-worn camera relies on robust real-time camera orientation tracking. In absence of robust tracking recovery methods, the complete acquisition process has to be re-started when tracking fails. This paper presents methodology for camera orientation relocalization, using virtual keyframes for online environment map construction. Instead of relying on real keyframes from incoming video, the proposed approach enables camera orientation relocalization by employing virtual keyframes which are distributed strategically within an environment map. We discuss our insights about a suitable number and distribution of virtual keyframes, as suggested by our experiments on virtual keyframe generation and orientation relocalization. After a shading correction step, we relocalize camera orientation in real-time by comparing the current camera frame to virtual keyframes. While expanding the captured environment map, we continue to simultaneously generate virtual keyframes within the completed portion of the map, as descriptors to estimate camera orientation. We implemented our camera orientation relocalizer with the help of a GPU fragment shader for real-time application, and evaluated the speed and accuracy of the proposed approach.

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

Keywords: Environment map, virtual key frame, vision-based tracking, camera pose relocalization

1 Introduction

Augmented Reality (AR) makes the physical world a part of the user interface experience, and has the potential to play a significant role in enhancing the mobile and wearable computing paradigm. Anywhere Augmentation is a conceptual extension of Mobile Augmented Reality (MAR) and its aim is to link location-specific computing services with the physical world, making them readily and directly available in any situation and location without relying on prepared environments or offline environment models [Höllerer et al. 2007]. Real-time visual tracking is used to estimate the pose of a camera relative to its surroundings. However, tracking failure is inevitable, and thus an efficient and accurate camera pose recovery is needed to provide a user with reliable tracking results.

In this paper we focus on the problem of camera orientation tracking with the goal of achieving real-time environment map acquisition, for which there are several motivating applications. Environment maps are useful as immersive representations of physical locations, e.g. as a backdrop in a tele-collaboration system, or in first-person interfaces such as QuickTime VR or Google Street View experiences. Remote presence applications [Uyttendaele et al. 2004] can use environment maps that are updated in real time as a simple way of representing and referring to dynamic remote environments. In AR systems, environment maps can be used to represent the light distribution around a single position in a compact image-based format. As such, they can be used for more seamless integration of virtual objects into the physical scene by supplying realistic image-based lighting for virtual geometry [Agusanto et al. 2003] [Grosch 2005].

Additionally, we wish to allow for a low computational cost orientation tracking solution. While our solution does not currently extend to six degrees of freedom (6DoF) position and orientation tracking, our work could be used as a component to a larger tracking system. So that, when users transition from movement, as tracked by some other system, to examining their surroundings, we can switch to Envisor as a reliable low cost method for orientation tracking while constructing environment maps at the same time. If we assume that scene geometry is relatively planar or sufficiently far away from a user, virtual keyframes can also be used for simultaneously generating and tracking over large-scale planar photo-panoramas from a panning camera in real-time. Another possible scenario is to capture an environment map of a scene while subsequently updating interesting sections with live video. For a big event, using a mobile phone with a small field of view, we could capture a panorama containing the audience and the surrounding buildings, and then capture live data being performed on the stage. It should also be possible to extend our virtual keyframe approach to 6DoF by reconstructing and then re-rendering 3D environments using image-based rendering or projective texture mapping, although this is clearly the domain of future work.

Up to now, a wide variety of tracking technologies, employing various sensors, have been investigated for AR systems. In general, relocalization has been performed by using a set of corner-like features or training a classifier with feature points. Subsampled images are also adopted as descriptors for relocalization. The methods are, however, only able to generate keyframes from video frames which have previously been acquired by a camera. Furthermore, camera pose recovery does not work well if the current camera pose is somewhat different from the camera pose at the time the keyframe was generated.

In this paper, we propose a novel camera pose relocalization approach for orientation tracking using virtual keyframes which are created from an environment map instead of incoming video frames, allowing for keyframes at novel view orientations and making it possible to provide a wider relocalization range. We use a vision-based hybrid tracking technique to create an environment map of the surrounding scene, online and largely independent of the users' viewing trace. To generate a seamless environment map, we keep exposure constant and perform a per frame correction for effects such as lens distortion and vignetting. Rotating the camera approximately around its focal point incrementally builds up an environment map of the surrounding scene while creating virtual

*e-mail: skim@cs.ucsb.edu
†e-mail: ccoffin@cs.ucsb.edu
‡e-mail: holl@cs.ucsb.edu

VRST 2009, Kyoto, Japan, November 18 – 20, 2009.
© 2009 ACM 978-1-60558-869-8/09/0011 $10.00

keyframes at the same time. Virtual keyframes are created only when all four of corners of a candidate virtual keyframe project onto the completed portion of the environment map.

We suggest a proper number of keyframes to be around 2,000 based on analysis performed in simulation and considering performance and orientation recovery. At this number of keyframes, we still manage real-time frame rates on a common laptop, and we get decent recovery performance (cf. Section 4). The ideal number of virtual keyframes is influenced by both system performance and convenience of use. Generating virtual keyframes is a relatively quick process, however generating large numbers can still affect the speed of the system, mostly due to the cost of relocalization. At the same time, a small number of keyframes may cause the user some difficulty in finding one of the keyframe locations for camera relocalization. Note that the number of keyframes needed for easy use is dependent not only on the distance between neighboring keyframe locations on a unit sphere, but also the number of virtual keyframes in the roll direction.

We present the following contributions in this paper. First, we provide an analysis for the suggested number and location of virtual keyframes for environment map generation. To achieve an acceptable visual quality, the distribution of virtual keyframes is discussed based on our simulation results. Second, we suggest how to generate virtual keyframes from the environment map on the fly using a fragment shader, and present a pose relocalization method for a camera that temporarily lost tracking (due to occlusion, motion blur, lack of visual features or other reasons) using the generated virtual keyframes. Finally, we compare the performance of the orientation tracking between an existing system [DiVerdi et al. 2008] and the proposed approach based on ground truth data which were recorded using a pan-tilt unit [DPerception 2009].

The rest of this paper is structured as follows: After a discussion of related work in Section 2, analysis of the orientation tracking module of Parallel Tracking and Mapping (PTAM) [Klein and Murray 2007] is discussed in Section 3. In Section 4, we introduce how to determine the number and distribution of virtual keyframes on the environment map, and we will describe an online virtual keyframe generation method and its use for pose recovery. Then, in Section 5, we demonstrate experimental results regarding speed and performance of the system. Finally, in Section 6, we present our conclusions and ideas for future work.

2 Related Work

In recent years, there has been steady research on camera pose relocalization. Pupilli and Calway propose a system which deals with short tracking failures in a monocular SLAM context [Pupilli and Calway 2005]. This is accomplished based on multiple hypotheses with a particle filter. The system is bootstrapped by a set of known 3D points to build up an initial particle distribution. As tracking progresses, new 3D points are introduced by identifying salient points and estimating their depths by triangulation of the camera particles. Se et al. focus on a global approach to relocalization for a moving robot [Se et al. 2005]. They use SIFT visual landmarks in unmodified environments to find matches to image features with map features, and build a 3D map of the environment by tracking the landmarks over time. These 3D landmarks are used to find the pose using RANSAC or a hough transform. Reitmayr and Drummond deal with camera pose recovery using keyframes which are saved during tracking [Reitmayr and Drummond 2006]. In the case of failure, they try to find a best-matching keyframe in the stored selection of older frames with the current video frame. They propose a statistical test to detect when the edge-based tracking system fails. They recover the camera from the proximity of a finite section of the previously traversed path. Williams et al. carry out relocalization by using a randomized list classifier to establish feature correspondences in an image [Williams et al. 2007]. Then, these correspondences are quickly detected for robust pose recovery using RANSAC when tracking fails. On the other hand, Klein and Murray employ subsampled blurry images as descriptors instead of extracting some form of interest features from keyframes [Klein and Murray 2008]. When tracking is lost, they subsample an incoming video frame, and apply a Gaussian blur. Then, they compare the incoming video frame with all of the virtual keyframes and find out the keyframe with the smallest sum-squared-difference. The final camera rotation is estimated using Efficient Second-order Method (ESM) Visual Tracking [Benhimane and Malis 2007] and best-fit 3D camera rotation estimation using virtual sample points. Most of the methods are, however, based on features in previously captured images and therefore keyframes are limited to previous camera poses. In this paper, we propose a method of generating virtual keyframes for relocalization even at camera orientations not present in the live video. We also discuss how these virtual keyframes provide a wider range of relocalization.

Our approach is different from PTAM in that our focus is to generate an environment map of the surrounding scene using a regular hand-held or head-worn camera (3DoF), while PTAM focuses on (6DoF) tracking of a camera with a wide-angle lens. Whereas PTAM generates keyframes from incoming video frames, we generate virtual keyframes from an environment map, providing more flexible generation of keyframes. With virtual keyframes, we can create a keyframe independent of the camera path. This is important in that we cannot guarantee that a user always moves his/her camera back to the exact same locations visited before. Furthermore, with PTAM, camera pose cannot be recovered if the roll of the current camera is different from that of the saved keyframe. However, using the virtual keyframe concept, we can easily generate a virtual keyframe with any camera roll at any keyframe location.

3 Analysis of Orientation Tracking of PTAM

Before we describe the generation of virtual keyframes, we first analyze the orientation recovery module of PTAM [Klein and Murray 2007] method when applied to the task of real-time panorama stitching from a fixed location. The method basically uses the ESM (Efficient Second-order Method) Visual Tracking and best-fit 3D camera rotation estimation by considering the motion of a few virtual sample points placed about the image. To evaluate the performance of the orientation recovery module in a controlled fashion, we simulate a physical environment with a computer graphics "sky box" using a set of cubemap images which are available at [Humus 2009]. A single keyframe is generated from an environment map and is used to recover the pose of the virtual camera.

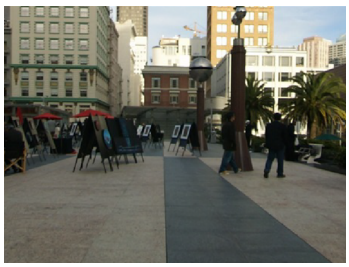


Figure 1: *Image captured from the center of a skybox*

Figure 1 is an image captured from the center of a skybox, and Figure 2 shows orientation relocalization results for changes in pure yaw and roll, respectively. For this simulation, the intrinsic parameters of a virtual camera are the same as those of the real camera

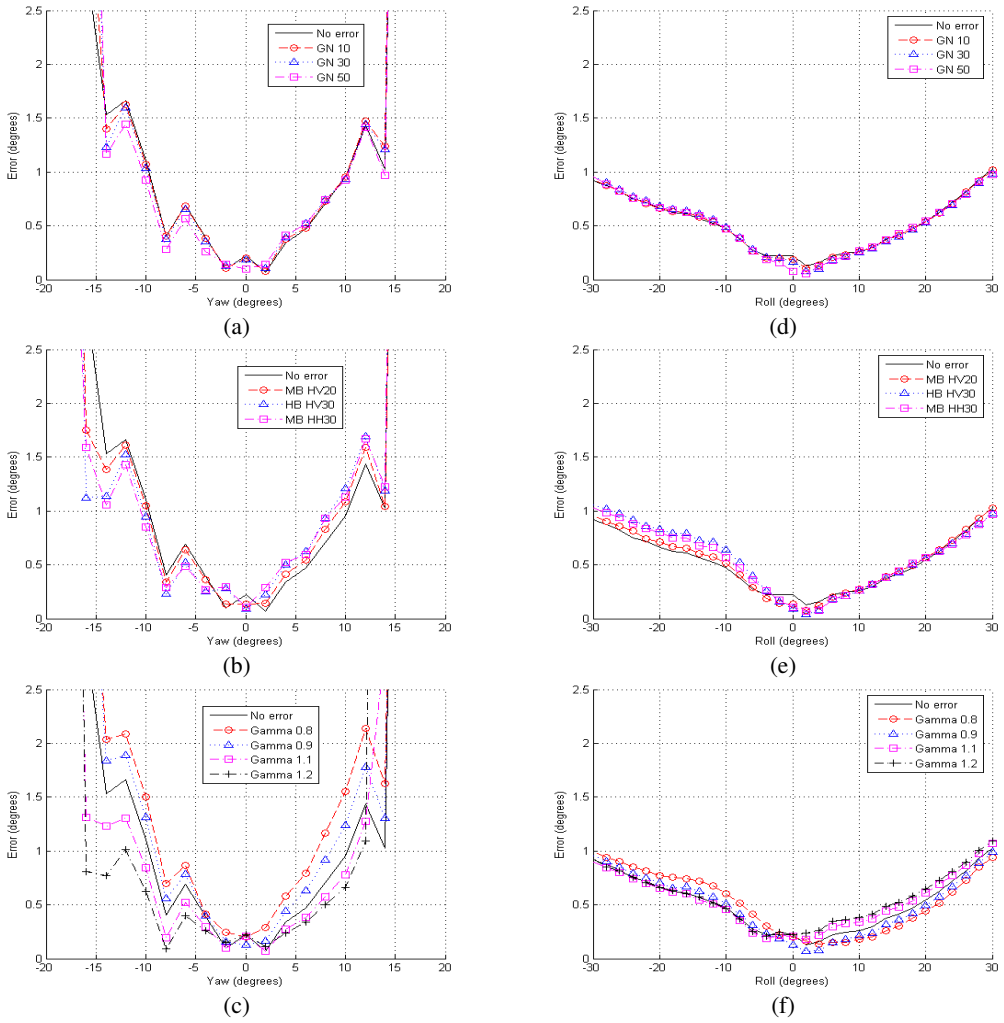


Figure 2: *Analysis of the orientation tracking of PTAM, Relocalization errors for yaw depending on (a) Gaussian noise (b) Motion blur (c) Gamma value, Relocalization errors for roll depending on (d) Gaussian noise (e) Motion blur (f) Gamma value*

used in the paper. In each figure, the vertical axis represents the absolute orientation error, which is the difference in degrees between the keyframe pose and the recovered camera pose, while the horizontal axis represents the tested deviation in yaw or roll.

Figure 2(a) and Figure 2(d) are the results of applying Gaussian noise to the keyframe and the current image with different standard deviations of 10, 30, and 50. In Figure 2(b) and Figure 2(e), 'HV20' means that we apply Gaussian motion blur of a 20 pixel radius horizontally to a keyframe and compare it with the current frame to which Gaussian motion blur of a 20 pixel radius is applied vertically. On the other hand, 'HH30' means that Gaussian motion blur of a 30 pixel radius is applied horizontally to a keyframe and the current image as well. For Figure 2(c) and Figure 2(f), different gamma values are applied to the keyframe only.

From the simulation results, we can see that Gaussian noise and motion blur have no serious influence on the relocalization performance. The main reason is that in PTAM, subsampling and blurring are applied to each keyframe and each video frame as well. However, as shown in Figure 2(c) and Figure 2(f), different lighting conditions depending on the different gamma values affect the relocalization performance considerably.

4 Virtual Keyframe-based Relocalization

We are working towards acquisition of surround-view panoramas using a single camera with robust real-time camera orientation tracking which is robust to fast smooth and abrupt orientation changes, as well as small deviations from the observer's position. In effect we want to make the acquisition of surround view panorama using a hand-held or head-worn camera as fast, convenient, and robust as possible. We propose methodology for camera orientation relocalization using virtual keyframes for online environment map construction. To this end, we present an analysis by which we determine a proper number and location of virtual keyframes. We also provide an online procedure for generating them and recovering camera orientations based on them. The flow diagram for the procedure is depicted in Figure 3.

4.1 Environment Map Construction

Unlike the conventional methods of generating keyframes for relocalization, we create virtual keyframes from an environment map which is constructed by rotating a camera. In order to construct the environment map, we use a vision-based hybrid orientation tracking mechanism which provides relatively drift-free orientation registra-

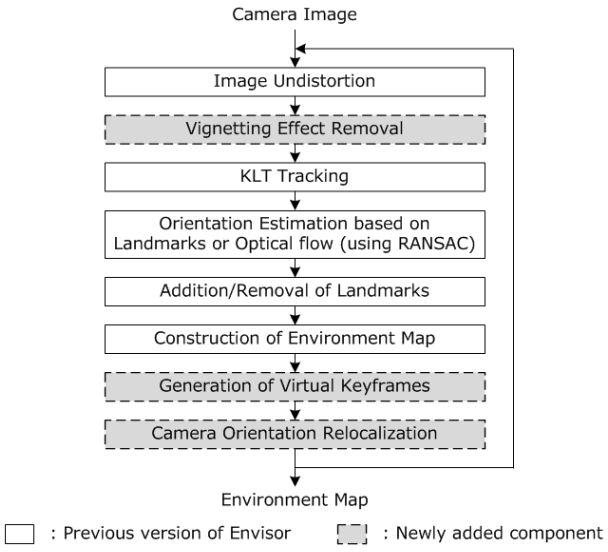


Figure 3: *An overall flow diagram*

tion [DiVerdi et al. 2008] [DiVerdi et al. 2009] [DiVerdi 2007]. The tracking mechanism is composed of two phases, one of which is frame to frame relative rotation tracking using Shi and Tomasi's feature detector and a pyramidal version of Lucas and Kanade's optical flow algorithm [Shi and Tomasi 1994] [Lucas and Kanade 1981]. The second phase is landmark-based absolute orientation tracking which adds landmarks to the frame to frame feature tracking system to combat drift during long tracking runs.

Using the orientation tracking techniques described here, we construct an environment map of the surrounding scene online and fully automatically. That is, with the help of the proposed pose relocalization, we can enable a user to generate an environment map independently of the camera path the user chooses to trace over the environment. We can recover from tracking failures due to such technical limitations as motion blur, varying lighting conditions, or insufficient textures on surrounding scenes simply by returning to a general area that was previously captured.

Firstly, each frame of a video stream is projected into a cubemap, masking out the dynamic portions of the scene. However, small gaps are likely to occur unless the user is very careful about complete coverage. Thus, we apply a texture diffusion technique to blend surrounding pixels into those gaps, reducing their visual impact.

In order to construct an evironment map for use with keyframes, the environment map should match the live video from the camera as closely as possible in terms of color and lighting. In order for this to be possible the environment map generated should be smooth. Any changes in the camera's settings such as exposure or white ballance should be avoided, or the settings locked, as sudden changes can cause discrepencies. It should be noted that these discrepencies can not be corrected by blending in new contributions from the environment map. Additionally, any distortion such as vignetting should be corrected on a per frame basis in order to avoid errors in the environment map. For all of our experiments, these settings have been locked and any distortion has been corrected.

We use Zhang's calibration technique to measure the camera's intrinsic parameters in a one-time offline calibration procedure [Zhang 2000] [Intel 2007]. In addition to the focal length and principal point, lens distortion parameters are also measured which are used to correct the position of features in the image, as well as to undistort each frame on the GPU so the image will match the pin-hole perspective model of OpenGL.

4.2 Number of Virtual Keyframes

We are ready to generate virtual keyframes during the online environment map construction. However, we have to decide how many virtual keyframes are needed for fast and accurate camera orientation relocalization. In addition, we should determine the locations of all virtual keyframes so that they can be generated while constructing an environment map.

Firstly, this question leads us to the Thomson Problem which is concerned with evenly distributing N points on a sphere. Using Thomson's solution to the problem [Thomson 1904], we generate virtual keyframes in N uniformly distributed locations. If N is very large, it takes a long time to compare the current frame with other keyframes. In addition, if the distribution of virtual keyframe locations becomes too dense, it causes delay since many virtual keyframes have to be created within a very short period of time. On the other hand, if N is very small, a user has to move his/her camera around to find relocalization locations, and the user may feel encumbered in finding the right spots to recover camera poses.

In addition to yaw and pitch, roll needs to be considered as well. As reported in [Klein and Murray 2008], relocalization is possible with a camera tilted up to about 35°. However, it is worthwhile to note that at such angular differences, accuracy suffers considerably. Relocalization accuracy is very important in our application since relocalization error greater than a threshold deteriorates visual quality drastically. In order words, even though it is important to recover the camera pose, to recover it with accuracy sufficient for rendering on the cubemap is critical.

Our experiments on the visual quality according to the orientation error showed that an error of about 0.7° does little harm to the visual appearance on the environment map. It should be noted here that the environment map we construct is a 512 by 512 pixel per face cube map. Thus, by considering the graphs in Figure 2, we conclude that about 10° difference between two adjacent virtual keyframe locations is reasonable for yaw and pitch. Figure 4 shows that we need about 400 virtual keyframe locations. On the other hand, a range from -15° to $+15^\circ$ is appropriate for roll when we consider the graphs in Figure 2. Thus, even though we can extend the range to full upside-down angle with the help of virtual keyframes, we use five different virtual keyframes at every 30° for a reliable orientation relocalization with a good visual quality, covering up to about 75°. In conclusion, the total number of virtual keyframes we need is 2,000 which are (400 virtual keyframes locations) $\times$ (5 directions in roll). Table 1 shows that about 7 ms are required in finding a best-match virtual keyframe among 2,000 virtual keyframes, which is reasonable for our real-time application.

4.3 Online Virtual Keyframe Generation

In Section 4.2, the number and locations of virtual keyframes to be generated on the fly are determined. In addition, we have to decide when we will create each virtual keyframe. For example, for the first virtual keyframe, an initial camera pose is given and the initial image is enough to create the first virtual keyframe. However, from the second virtual keyframe, it is not easy to create since the interior area of each virtual keyframe is not fully filled yet. That is, in order to generate a new virtual keyframe we have to wait until the whole area of each virtual keyframe is fully filled with valid pixel values.

Instead of checking each pixel within a virtual keyframe, we use only four corners of each virtual keyframe to decide whether a planned virtual keyframe is valid or not. We calculate its four

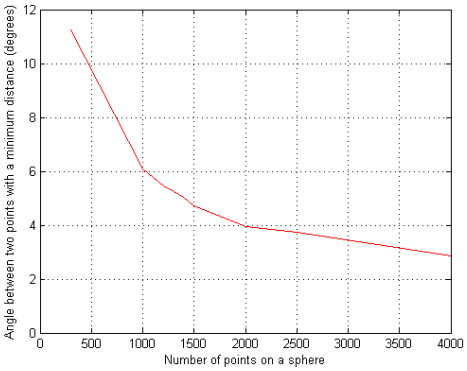


Figure 4: *Angle between two points with a minimum distance according to the number of points on a sphere*

corners by projecting the view frustum of the camera onto the cubemap. Because we already have an ideal constellation of virtual keyframes on the sphere with yaw and pitch information for each location, we can calculate four corners for all of the virtual keyframes. If the four corners become valid, the corresponding virtual keyframe image is immediately generated from the environment map and saved into the database. It is worthwhile to note that we subsample the virtual keyframe image down to 40×30 pixels, apply a Gaussian blur of $\sigma = 2.5$ pixels, and subtract the mean image intensity. This zero-mean, blurred image is saved as the virtual keyframe's descriptor, and used to calculate the sum-squared-difference (SSD) with an incoming video frame later. The corresponding absolute orientation information is saved as well, to be used for recovering the camera pose when the camera gets lost.

One possible solution is to read back the pixels of the four corners of each virtual keyframe from the cubemap directly, but this pixel readback is extremely slow as it breaks GPU pipelining by introducing a stall. It is worthwhile to note that good pipelining is critical for performance since the panorama construction already has heavy CPU and GPU use [DiVerdi et al. 2008]. Instead, the cubemap is sampled according to the direction of four corners of a virtual keyframe. Then, the four samples are combined by blending with an additive blend function, and a fragment shader is used to see if the sampled values are gaps or not. That is, using the fragment shader, we can check if all corners of each virtual keyframe have been colored on the environment map or not. Each corner of a virtual keyframe is weighted 1/4, and if all of the four corners are filled with the video input, then the output would be one. The result is M (the number of virtual keyframes) pixels in an offscreen buffer, each with an alpha value that represents the weight for the corresponding virtual keyframe.

Note that Envisor includes a technique which is very applicable to our current key frame approach. During the environment map construction, a mask is used to determine the blending of the current camera image onto the environment map. Gaussian splats are drawn into this mask around each of the outlier features and these splats are then blended together. In regions with many outliers, the entire area is masked out, and we avoid using those sections when projecting onto the cubemap. As a result the cubemap and therefore virtual keyframes we generate tend to be absent of any moving outliers such as cars or people, and therefore provide a better base for our recovery.

4.4 Pose Recovery

This section describes relocalization based on the virtual keyframes which are generated from the environment map on the fly. Basically, the procedure is following the method proposed by [Klein and Murray 2008]. That is, the subsampled and blurred images of the virtual keyframe and the incoming frame are compared constantly for camera orientation recovery. First, we subsample an incoming video frame down to 40×30 pixels, apply a Gaussian blur of $\sigma = 2.5$ pixels, and subtract the mean image intensity, which is the same procedure we apply to the virtual keyframe. Then, we compare the incoming video frame with all of the virtual keyframes and find the virtual keyframe with the smallest SSD. The final camera rotation is estimated with the help of ESM Visual Tracking and best-fit 3D camera rotation estimation by considering the motion of a few virtual sample points placed about the image.

5 Experimental results

Images are captured from a PointGrey DragonFly2 video camera equipped with a variable iris lens. The camera delivers 640×480 pixel RGB frames at 30Hz. These frames are converted to 8bpp greyscale for tracking and an RGB image for display. The intrinsic camera parameters were evaluated by Zhang's calibration procedure using the implementation of OpenCV. For testing against the orientation ground truth, we used a D46-17 pan tilt unit from Directed Perception. The step size of the device is 0.0514 degrees, with speeds over 300 degrees per second.

5.1 Accuracy

To show that the proposed orientation recovery based on virtual keyframes works well, we compared the current system with the previous version of Envisor [DiVerdi et al. 2008]. Each of our tests consisted of an initialization stage, where a camera is moved around the scene to generate virtual keyframes. We then tested various speeds of rotation for a range of 110° around the vertical axis. While it is possible to define a line or path avoiding the virtual keyframes, our tests assume that the camera comes across these points, so without loss of generality we moved the camera in the yaw direction only. In practice the time it takes a user to find a keyframe may vary, so as a control for the number of points encountered, we stationed some virtual keyframes along the horizontal line at every 10°. Since we are using 400 virtual keyframe locations and its angle between two points with a minimum distance is about 10°, our experimental setup is reasonable, if idealized. Basically, Envisor uses a hybrid vision-based tracking approach which combines a frame to frame relative rotation tracking and landmark-based absolute orientation tracking. For a fair comparison between two systems, we turned off the landmark-based tracking module when testing the virtual keyframe system. Normally, however, landmarks would still be used to estimate the camera orientation simultaneously with virtual keyframes.

Figure 5 shows the comparison results of camera orientation tracking accuracy between the conventional Envisor and the proposed system at different angular speeds of 30°/sec, (b) 60°/sec and (c) 80°/sec. As mentioned, in order to ensure a fair and accurate comparison between the tracking methods, we mounted our camera on our pan tilt unit and acquired a ground truth orientation sample for each frame of video.

Playing back the video in real-time, we confirmed the performance results mentioned in the Envisor paper, as our system was running at between 10 to 15 fps and failed at around 35°/sec. Interestingly, allowing the video to run at the full 30 fps (possible in real-time only on an ideal computer), we noticed that Envisor will break at around 90° per second movement. Note that this has to be considered offline performance, and is not used for our realtime evaluations.

As shown in Figure 5(a), the results are similar in both cases when

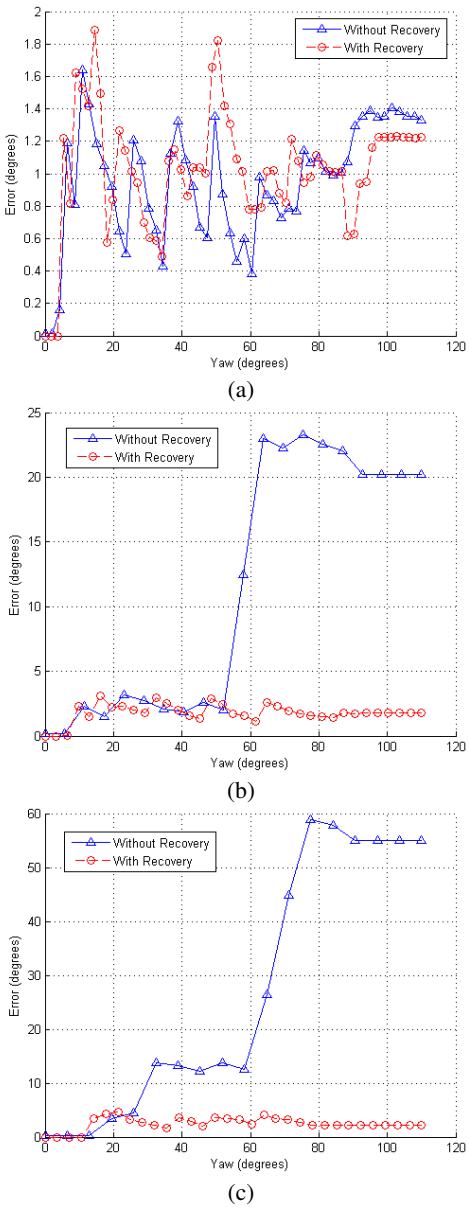


Figure 5: *Comparison of camera orientation tracking accuracy between previous version of Envisor and proposed system at the angular velocity of (a)* 30°*/sec, (b)* 60°*/sec and (c)* 80°*/sec*

the camera motion is sufficiently moderate. However, when the speed exceeds the previously mentioned threshold, the conventional Envisor always failed as shown in Figure 5(b) and Figure 5(c).

However, with the proposed orientation relocalization approach, we can recover the camera orientation and continue tracking. If we compare Figure 5(b) and Figure 5(c), we can see that the error for Envisor increases with higher angular speed.

In order to test our system in a general case, we obtained free motion data using an inertial tracker (InertiaCube2 [InterSense 2009]) which provides 3DoF information in real-time. We placed the inertial tracker on a user's head and recorded the movement of that person casually looking around. Based on the data, we again generated video frames and the corresponding ground truth data using the pan tilt unit, and compared two systems.

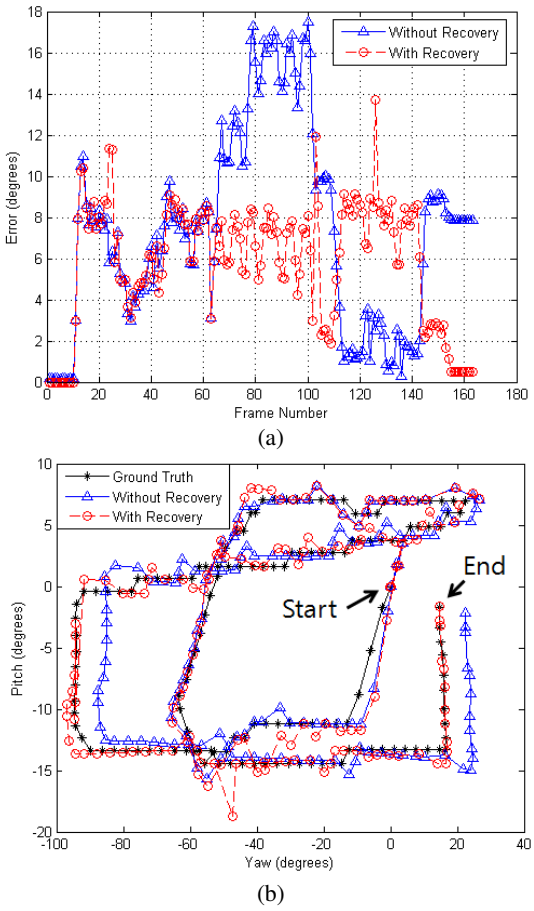
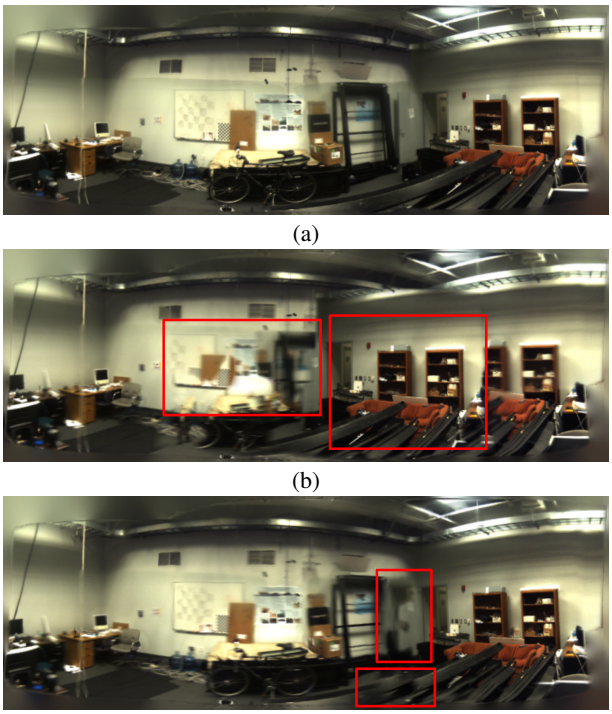


Figure 6: *Comparison of camera orientation tracking accuracy between previous version of Envisor and proposed system in case of a free motion (a) absolute orientation error (b) motion on the yaw-pitch plane: tracking starts at (0,0) and continues downward, looping around and ending at about (14.7,-1.7)*

In Figure 6 we show the comparison results of camera orientation tracking accuracy between two systems in the case of free motion. Figure 6(a) shows absolute orientation errors according to a frame number and Figure 6(b) shows the user's head motion on the yaw-pitch plane. As shown in Figure 6(a), there are large errors from the beginning and the conventional Envisor without recovery failed during a section of fast movement. The proposed system with recovery capability tries to find the best virtual keyframe and recover the orientation, generally keeping the error within 10 degrees.

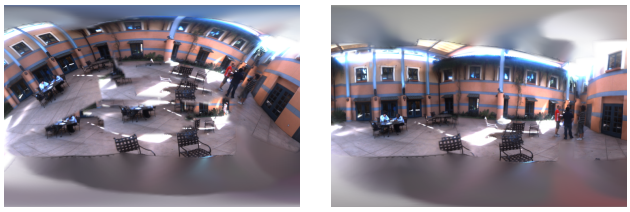
If we maintain the orientation tracking even with large errors, it provides a user with a pleasing visual appearance. That is, to maintain the tracking is very different from the case of losing the camera tracking for a user. We can see that in the end of the video sequence, the proposed system provides a much lower error than the conventional system. In Figure 6(b), we show the ground truth path along side the estimations produced using the two tracking systems. Even though the proposed system sometimes shows large errors, it maintains good tracking results, and outperforms the conventional Envisor system.

We also compare the environment maps that result from the same camera motion with the recovery ability either switched on or off, as shown in Figure 7. We first acquired Figure 7(a) as a reference environment map at a low speed, and generated virtual keyframes from the same sequence. Then we traced over the environment map using



(a) (b) (c)

Figure 7: *Comparison of environment maps between previous version of Envisor and the proposed system (a) reference (at a low speed) (b) without recovery (c) with recovery*



(a) (b)

Figure 8: *Comparison of environment maps of an outdoor scene between the previous version of Envisor and the proposed system (a) without recovery (b) with recovery*

faster and non-uniform camera rotation. As shown in Figure 7(b), without recovery capability, this results in displeasing errors within the highlighted rectangles. In addition, we can observe that in this case the camera was not able to be tracked to the correct final position. However, in Figure 7(c) we can see that recovery works well and the fast trace-over leaves an environment that is kept intact. There remain small errors due to tracking being temporarily lost but soon recovered, as well as recorded motion blur.

It is worthwhile to compare environment maps of an outdoor scene as shown in Figure 8. In this case, we did not use an image sequence for generating virtual keyframes, but instead generated them in a live fashion while moving a hand-held camera around the scene. On purpose, we moved our camera very fast five times to make tracking fail. Figure 8(a) and Figure 8(b) show tracking results without and with the relocalization capability. Whereas there are many misalignments in Figure 8(a), Figure 8(b) exhibits only small errors thanks to virtual keyframe recovery.

5.2 Performance

While our camera runs at 30 fps, Envisor runs at approximately 10 to 15 fps on our system (Dell XPS M1210, 2.0G Hz CPU), which is similar to the timing information reported in [DiVerdi et al. 2008]. On the other hand, we need to consider the processing time required to find the best-match keyframe according to the number of virtual keyframes. As shown in Table 1, the processing time increases almost linearly. Since we use 2,000 virtual keyframes, we spend about 7 ms on recovery which is not a big overhead for our system.

Table 1: *Processing time required to find the best-match keyframe according to the number of virtual keyframes.*

Number of virtual keyframes	Processing Time (ms)
400	1.6141
800	3.0532
1,200	4.3975
1,600	5.8999
2,000	7.2472
2,400	8.4199

5.3 Analysis and Discussion

In summary, we offer an improvement to the existing Envisor system by allowing it to recover from tracking errors through the use of virtual keyframes. The environment map provides a useful data structure for creating and arranging the virtual keyframes.

In order to construct a useful environment map for keyframe matching, camera distortion such as vignetting will always need to be corrected. Currently, we also require that camera exposure be locked. However, it can be difficult to maintain tracking in an outdoor environment with these restrictions. Fortunately, this restriction can be overcome if the exposure settings of the camera are known and can be compensated for when creating the environment map.

One drawback of our system is that, in order to use the keyframe recovery the user must first construct a partial environment map in the area. However, this is made easier by the fact that recovery works even during this initial step. During the construction the user is able to quickly move back over any previously covered regions to regain tracking in case of tracking loss and can now more carefully cover the area where tracking failed.

Our system needs sufficient texture information in surrounding scenes since it relies on detected features. Thus, textureless areas, e.g., homogeneous walls and skies, can make tracking fail. Another difficulty is motion blur due to fast camera motions, making feature detection difficult. However, since we use a relocalization approach based on the proposed virtual keyframes, a user can continue to generate an environment map even after tracking failure occurs if he/she moves the camera back to a valid environment map area.

The current system can be used only for generating a continuous panorama with orientation tracking only. However, if we can assume that the scene is sufficiently far from a user, then we could generate a panoramic scene while moving the camera. We are also working on speed improvements for our system in order to further raise the level of the tolerated speed and arbitrariness of camera motion and thereby the convenience and robustness of generating environment maps. One alternative is to employ a FAST detector instead of Shi and Tomasi's feature operator, and to replace SURF descriptors with image patch descriptors.

In addition, the pose recovery is vulnerable to varying lighting conditions. If different areas of a single virtual keyframe are acquired at

different lighting conditions, pose recovery will not work well. This is because the virtual keyframe will be compared to the current image which is acquired at a single lighting condition. However, if the change of lighting conditions has an effect on the whole image, the recovery will still work. The reason is that a zero-mean, blurred image is saved as a virtual keyframe's descriptor, and used to calculate the sum-squared-difference with an incoming video frame.

6 Conclusion

A robust tracking recovery method is necessary for real-time surround-view panorama acquisition to prevent the need for a system restart. We presented a relocalization method using virtual keyframes for online environment map construction. We first discussed how to arrive at a suitable number of virtual keyframes and how to distribute them to get good performance of orientation recovery. We then enabled the system to generate virtual keyframes on the fly while constructing the environment map using a fragment shader. Our results demonstrate that a user can create an environment map on the fly by rotating his/her camera and recover the camera orientation with enough accuracy to continue generating a panoramic map. There are still several remaining challenges. Currently, we distribute the locations of the virtual keyframes statically at the start of the environment map construction. While this works very well if the user intends to build a full model of the environment, an augmented reality user may be interested in only constructing a small section of the scene, or may not be interested in capturing the ground or sky. One possibility for future work would be to dynamically distribute the locations of the virtual keyframes. This would provide a higher density of keyframes and therefore better recovery at the start of the environment map construction. In addition, even if we deal with orientation tracking to construct an environment map using a stationary camera, we cannot guarantee that the optical center of the camera is kept at an exactly constant location. We will consider this translational component to enable robust tracking even in presence of considerable position deviations. Furthermore, we are experimenting with several alternatives to Surf-based feature descriptors in order to speed up the system.

Acknowledgements

This work was supported in part by a research contract with KIST through the Tangible Space Initiative Project, and NSF CAREER grant #IIS-0747520. The authors also wish to thank Stephen DiVerdi for his development of the original Envisor system and his continued help.

References

AGUSANTO, K., LI, L., CHUANGUI, Z., AND SING, N. W. 2003. Photorealistic rendering for augmented reality using environment illumination. In *ISMAR '03: Proc. 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, IEEE Computer Society, Washington, DC, USA, 208.

BENHIMANE, S., AND MALIS, E. 2007. Homography-based 2d visual tracking and servoing. *Special Joint Issue on Robotics and Vision. Journal of Robotics Research 26*, 7 (July), 661–676.

DIVERDI, S., WITHER, J., AND HÖLLERER, T. 2008. Envisor: Online environment map construction for mixed reality. In *IEEE VR*, 19–26.

DIVERDI, S., WITHER, J., AND HÖLLERER, T. 2009. All around the map: Online spherical panorama construction. *Computers and Graphics 33*, 1, 835–846.

DIVERDI, S. 2007. Towards anywhere augmentation. *Ph.D. dissertation*, 835–846.

DPERCEPTION, 2009. http://www.dperception.com, June.

GROSCH, T. 2005. PanoAR: Interactive augmentation of omnidirectional images with consistent lighting. In *Mirage 2005, Computer Vision / Computer Graphics Collaboration Techniques and Applications*, 25–34.

HÖLLERER, T., WITHER, J., AND DIVERDI, S. 2007. "Anywhere Augmentation": Towards mobile augmented reality in unprepared environments. In *Location Based Services and TeleCartography, Series: Lecture Notes in Geoinformation and Cartography*, Springer Verlag, G. Gartner, M. Peterson, and W. Cartwright, Eds., 393–416.

HUMUS, 2009. http://www.humus.name/, June.

INTEL, 2007. Open source computer vision library. http://www.intel.com/technology/computing/opencv/, May.

INTERSENSE, 2009. http://www.intersense.com/, June.

KLEIN, G., AND MURRAY, D. 2007. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*.

KLEIN, G., AND MURRAY, D. 2008. Improving the agility of keyframe-based SLAM. In *Proc. 10th European Conference on Computer Vision (ECCV'08)*, 802–815.

LUCAS, B., AND KANADE, T. 1981. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 674–679.

PUPILLI, M., AND CALWAY, A. 2005. Real-time camera tracking using a particle filter. In *British Machine Vision Conference*, 519–528.

REITMAYR, G., AND DRUMMOND, T. W. 2006. Going out: Robust tracking for outdoor augmented reality. In *Proc. Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'06)*, 109–118.

SE, S., LOWE, D. G., AND LITTLE, J. J. 2005. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics 21*, 364–375.

SHI, J., AND TOMASI, C. 1994. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, 593–600.

THOMSON, J. J. 1904. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. *Philosophical Magazine 7*, 6 (March), 237–265.

UYTTENDAELE, M., CRIMINISI, A., KANG, S. B., WINDER, S., SZELISKI, R., AND HARTLEY, R. 2004. Image-based interactive exploration of real-world environments. *IEEE Comput. Graph. Appl. 24*, 3, 52–63.

WILLIAMS, B., KLEIN, G., AND REID, I. 2007. Real-time SLAM relocalisation. In *Proc. International Conference on Computer Vision (ICCV)*, 1–8.

ZHANG, Z. 2000. A flexible new technique for camera calibration. *Transactions on PAMI 22*, 11, 1330–1334.