Outdoor Mobile Localization from Panoramic Imagery

Jonathan Ventura
Department of Computer Science
University of California, Santa Barbara

Tobias Höllerer
Department of Computer Science
University of California, Santa Barbara

Abstract

We describe an end-to-end system for mobile, vision-based localization and tracking in urban environments. Our system uses panoramic imagery which is processed and indexed to provide localization coverage over a large area using few capture points. We utilize a client-server model which allows for remote computation and data storage while maintaining real-time tracking performance. Previous search results are cached and re-used by the mobile client to minimize communication overhead. We evaluate the use of the system for flexible real-time camera tracking in large outdoor spaces.

Index Terms: I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—3D/stereo scene analysis; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking

1 Introduction

In this paper we consider how outdoor mobile localization and tracking can be realized using panoramic environment capture and a client-server model for image retrieval. Recently, researchers have investigated how to adapt geometric image retrieval to the mobile phone platform, achieving 2-3 Hz tracking [1]. However, a remote server or a computation cloud offers the possibility of using much greater computation and storage resources than available on a mobile phone, as well as greater parallelism. In this work we consider pairing a lightweight localization method, performed on the client, with a more capable image search procedure, performed on the server. In contrast to previous work on client-server based tracking [3], we demonstrate wide-area 6DOF tracking.

2 Panoramic Modeling

Our system produces a model from panoramas captured in the outdoor environment to be tracked. To prepare input for our modeling procedure, we first extract perspective views from the source panoramas. We use four orthogonal views to capture the entire horizontal field of view. Unlike a traditional cube map, we use an 'extended cube map' which has increased horizontal field of view for each face. We also perform vertical and horizontal vanishing point alignment to maximize image resolution of the building facades.

Structure from motion proceeds in a linear fashion, since we assume that the panoramas were taken by moving along a path. The panoramas are first organized into triplets, independently reconstructed, and then merged into a common scale. We use the upright constraint to improve pose estimation robustness [9]. After complete reconstruction using SIFT features [7], we re-triangulate points and perform bundle adjustment using the pyramidal search methods of the PTAM system [6]. We then extract SIFT descriptors for the triangulated FAST corners, to be used for localization.

3 Localization and Tracking

Our system uses the patch-based tracker from the PTAM system [6], although the online mapping capability is not used in our system. When the tracker is in a 'lost' state, such that there is no prior estimate of the camera pose, a suitable localization technique must be used to recover the camera pose. We use two localization techniques in tandem.

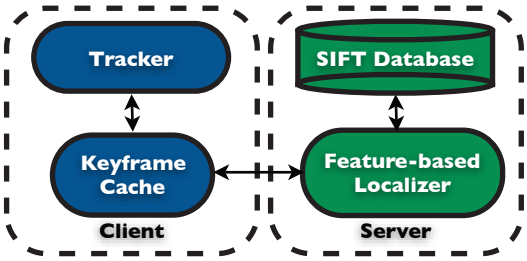

Figure 1: Client / server system outline for tracking and localization. When the local image cache fails to find a match, localization queries are enqueued on the remote server. When ready, the result is added back to the image cache for later use.

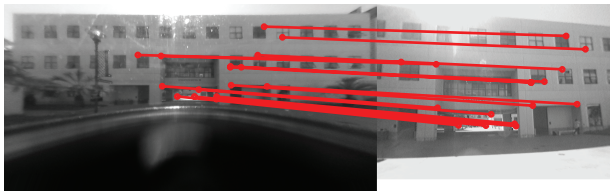

Figure 2: Matching between panorama keyframe (*left*) and a mobile camera image (*right*). The lines indicate 3D-2D correspondences determined to be inliers by the geometric verification process.

The system first queries a local image cache containing recently seen images together with their known pose. The closest matching image, after downsampling and blurring, is chosen as a potential match [2]. The pose of the cached image is taken as the current pose, after a rotational adjustment. If the tracker succeeds in updating the pose using this prior, then the localization is determined to be correct. The image-based search is fast, taking on average 10 ms on our test machine. However, it is not very robust to changes in rotation or scale (see Section 3.1).

If the image-based localization fails, the system enqueues the image to be processed using feature-based localization. We use a vocabulary tree to classify SIFT descriptors, and *tf-idf* weighted matching to identify potentially matching images [8]. The top-ranked images are geometrically verified using the three-point pose algorithm [4], and then we run the tracker to determine if the pose prior leads to successful tracking. If the best two poses are consistent up to a small rotation, we accept the best pose as correct. This rigorous procedure is too slow to be real-time, taking on average 400 ms for SIFT computation and 50 ms per document for geometric verification. It also requires storage of high-dimensional descriptors and their indexing structures. Because of these issues, we relegate this processing to either a background thread, or a separate server on the network. If successful, the localized query image is then added to the tracker's image cache.

3.1 Localization Method Comparison

We compared the image- and vocabulary-based localization methods on a simulated urban environment in our lab. We used the City of Sights scale building models, which are made using printed paper [5]. This setup allowed us to simulate images of an urban environment while allowing more control of the capture environment compared to the outdoors. PTAM was used to reconstruct the scene. In the next section we will discuss our outdoor experiments using our reconstruction methods.

We ran both localization methods on image sequences taken with increasing distance from the scene. The feature-based localization method outperformed the image-based method in all cases. The image-based method was unsuccessful in all image sequences beyond the first, which was taken at roughly the same distance as the keyframes in the map. This shows that feature-based localization offers much greater range in comparison to the image-based method. From our experiment, we conclude that the feature-based localization can be applied successfully when the distance to the tracked objects is at most five times that of the map keyframes.

4 System Evaluation

We sampled panoramas from three different building courtyards on our campus, using the Sony Bloggie catadioptric camera mounted on a wheeled cart. Each panoramic sequence was reconstructed using our pipeline.

We also captured a twenty second video in each outdoor scene. Each video contains a combination of panning the camera from one location, and moving between locations. For our tests we used a 2.4 GHz dual-core laptop with an outward-facing Unibrain Fire-i camera attached. The camera produces 640×480 pixel images, and we used a 2.1-mm lens which gives a horizontal field of view of about 80 degrees. In all tests we ran the the tracking client and localization server in separate threads. Running each video at half speed (15 frames per second) to help cope with the slowness of the feature-based localizer, we achieved an average of 52% frames tracked.

One important aspect of the system to evaluate is the relationship between localization latency and tracker performance. This latency could be due to both computation time and network communication time, in the case of a remote localization server. During the time in which the client waits for server result, the user might move the camera to a different viewpoint, from which the result cannot be used. As the server latency increases, the likelihood of the camera moving away from the query frame also increases. The optimal system would finish localization before the user gives up and directs the camera somewhere else.

To test the effect of latency in our system, we evaluated tracking performance for each video using a controlled localization latency. For each video, we pre-computed the feature-based localization result for each frame, and stored the results. This allowed us to control the localization latency by adding a delay to the system before loading a pre-computed query result. We ran each video through the tracker at 30 frames per second using a range of delays, from 0.25 to 8 seconds. We also included a zero-latency condition, to test the ideal case where localization results are immediately available to the tracker.

Figure 3 plots the fraction of frames tracked for all videos. Increasing latency generally reduces the number of tracked frames, as is expected. Higher latency means that the system has fewer chances to attempt tracking recovery, because it must wait for the previous result to be computed. This results in the tracker spending more time in a lost state. However, the tracker performance is also highly dependent on the camera movements made. With less camera movement, the system has a greater chance of recognizing and localizing to a previously seen view, since there is less difference between views.

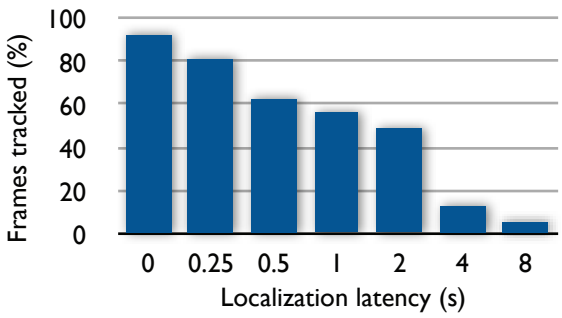

Figure 3: Average tracking performance with increasing localization latency – the amount of delay before the system returned a localization result during periods of lost tracking. We tested the system on a 20 second video in three locations.

Our results show that even with 500 ms of localization latency, good tracking performance is possible, with greater than 60% of frames tracked in all videos. Because localization results are prepared in the background, and cached for later use, the system can tolerate a localization method that does not have real-time performance. With reasonable camera movement, our results suggest it is sufficient to optimize the localization algorithm to return a result within a half a second.

5 Conclusion

In this work we proposed a pipeline for processing sequences of panoramas taken in urban environments, considered how to prepare usable keyframes for a mobile tracking system, and evaluated the system in terms of latency due to image search computation and network communication. Our system offers a compromise between fast image-based localization, and a feature-based method which has greater range but requires more computation.

Acknowledgements

This work was partially supported by NSF CAREER grant IIS-0747520.

References

[1] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide area localization on mobile phones. *International Symposium on Mixed and Augmented Reality*, 2009.

[2] R. O. Castle, G. Klein, and D. W. Murray. Video-rate localization in multiple maps for wearable augmented reality. *International Symposium on Wearable Computing*, 2008.

[3] D. M. Chen, S. S. Tsai, R. Vedantham, R. Grzeszczuk, and B. Girod. Streaming mobile augmented reality on mobile phones. *International Symposium on Mixed and Augmented Reality*, 2009.

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.

[5] L. Gruber, S. Gauglitz, J. Ventura, S. Zollmann, M. Huber, M. Schlegel, G. Klinker, D. Schmalstieg, and T. Höllerer. The city of sights: Design, construction and measurement of an augmented reality stage set. *International Symposium on Mixed and Augmented Reality*, 2010.

[6] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. *International Symposium on Mixed and Augmented Reality*, 2007.

[7] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[8] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. *Computer Vision and Pattern Recognition*, 2007.

[9] J. Ventura and T. Höllerer. Structure and motion in urban environments using upright panoramas. *Virtual Reality (*under review*)*, 2011.