International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

An Elaborated Survey on Mobile based Augmented Reality System

Shaik Abdul Gaffar [1], Shaik Nyamathulla[2]

[1]Computer Science Engineering, Guntur Engineering College, Guntur, India

Abstract- ***Augmented reality (AR) is a live and combines direct or indirect view of a physical, real-world environment it's real time data whose elements are augmented by computer generated virtual content such as sound, video, graphics or GPS data. An AR system poses unique challenges including requiring a high quality camera pose estimate and operating on resource limited platforms. There is several hybrid approach using ORB binary features and optic flow that is able to real time performance result is possible with platform specific optimizations, improve speed and extend the usable tracking range.***

Keywords: Augmented Reality; Markerless; Mobile; Real Time; Feature Detectors; ORB; Hybrid Tracking

1. Introduction

Augmented reality is a concept of supplementing the real world with the virtual world. Although it uses a virtual environment created by computer graphics, its main playground is the real environment. Computer graphics serve the function of adding necessary information into the real environment. In so doing, it makes up for the weak point of unreality which can occur in the environment providing only the virtual world. Augmented reality is to improve the recognition tools for the real world and thus to efficiently interact between humans and computers.

In order to perform the object tracking, markerless augmented reality systems rely in natural features instead of fiducial marks. Therefore, there are no ambient intrusive markers which are not really part of the environment. Furthermore, markerless augmented reality counts on specialized and robust trackers already available. Another advantage of the markerless systems is the possibility of extracting from the environment characteristics and information that may later be used by them. However, among the disadvantages we can consider for markerless augmented reality systems is that tracking and registration techniques become more complex.

Old AR systems like ARTag [1] utilize fiducial markers or “tags”, over which virtual objects are rendered. Marker-based systems are ideal for applications involving a static or fixed environment, or situations where the desired virtual defined marker tag does not extend beyond the tag itself. In the performing and using of application, users must print and carry the tags in order to use the AR application. For more complex, large scale applications, significant infrastructure is required in the form of huge number of or hundreds of markers placed throughout the environment [2].

Figure 1: AR pipeline overview

In this paper, we present an end-to-end markerless AR pipeline based on the binary ORB descriptor by Rublee et al [3]. The system is capable of real time performance on current generation consumer grade mobile devices. The system requires nothing more than a calibrated monocular camera; every other process is performed online and in real time, including map creation. The computational restrictions of mobile devices are addressed at each stage in the pipeline, and an Android implementation is used to evaluate the system as a whole.

2. Previous Work

There has been myriad work done in the field of AR over the past decade. In this paper we focus mainly on work related to markerless AR, mobile AR, and their relevant computer vision algorithms.

The best known work in monocular camera localization and mapping is the MonoSLAM system developed by Davison et al [4]. The authors successfully applied SLAM methodologies to the vision domain using a monocular camera as the sole input device. MonoSLAM maintains a sparse map of oriented planar textures (~12 for any given camera view) centered on Shi-Tomasicorners. The system is initialized using a planar target containing four known features before dynamically updating the map. They present results in the AR domain, and achieve real-time operation at 30Hz while rendering augmentations.

The Parallel Tracking and Mapping (PTAM) system, introduced by Klein and Murray in 2007 [5], uses a multi-threaded approach in order to simultaneously track interest points by morphing and matching image patches, and maintain a map of these patches. PTAM uses a stereo pair of images of a planar scene for initialization, and uses bundle adjustment to determine and update the 3D locations of interest points.

Taehee and Hollerer also propose a multi-threaded approach in [6]. They use a hybrid tracking method which extracts SIFT features [7] (instead of image patches) and then tracks them using optic flow. They achieved real-time performance by only extracting and matching SIFT features periodically in a thread separate from the tracker. They also perform scene recognition by recognizing previously recorded SIFT features.

On the mobile front, attempts have been made to adapt SIFT and its speedier counterpart SURF [8] to mobile devices. Wagner et al proposed a hybrid between FAST corners [9] and a reduced version of the SIFT descriptor in [10]. Ratherthan compute descriptors every frame, they also adopt a hybrid approach and track existing features using SAD patch correlation. This method achieves upwards of 20Hz while extracting ~150 features per frame (320x240). Chen et al propose a modified version of SURF in [11] that achieves a roughly 30% speed-up over the original SURF algorithm, but nevertheless falls far short of real-time operation on mobile devices.

The first self-contained AR system to run on a consumer-grade cell phone was presented in 2004 by Mohring et al [12]. Their system recognizes different markers via circular bar codes and detecting gradient changes in the red, green, and blue color channels. Their entire pipeline achieved a frame rate of 4-5fps, at a camera resolution of 160x120.

A reduced version of the PTAM system has been adapted to the iPhone [13], but results showed severely reduced accuracy and execution speed. PTAM is intended for use in small AR workspaces, and suffers reduced performance as the map gets bigger and bigger due to the bundle adjustment process being cubic with respect to the number of features in the map (O(n3)).

As a continuation of their work in [10], Wagner et al propose a more complete version of their feature detection and matching system in [14]. They use a similarly modified version of SIFT and outlier rejection that runs at approximately 26Hz. However, when AR-related overhead (image retrieval from camera, rendering, etc.) is taken into consideration, the speed drops to 15Hz. They combine this with patch tracking to greatly improve speed to 8ms per frame (not including AR overhead) instead of using SIFT on every frame. Their system runs on 320x240 imagery during the SIFT phase, and down-samples further to 160x120 while tracking. In addition, a maximum of 100 features are tracked at any given time.

To the best of our knowledge, there exists no complete AR system that incorporates all the elements presented in this paper while achieving real-time operation on mobile devices. Most notably absent from existing systems are online map creation and multiple map support.

3. Technical Approach

This section describes our system. The stages of the AR pipeline shown in Figure 1 are discussed. For each stage we discuss which algorithms are used, and any modifications that were made to improve efficiency.

3.1 Map Overview

We define a ‘map’ as a list of feature descriptors and their corresponding 3D world coordinates. In our system, matching features in an input video frame to a map is accelerated by using a tree structure. When a map is created, the descriptors are placed into a clustering tree with eight branches as done by Muja and Lowe in [15]. Each map has an associated 3D bounding box consisting of four world coordinates. This bounding box can be projected into the image frame once camera pose has been estimated. Additionally, the system maintains a 100-word descriptor vocabulary. Each map is matched to this vocabulary to create unique response histograms.

3.2 Platform Specific Acceleration

To achieve real time speeds on a resource limited mobile devices, platform specific adaptations are used. This system makes use of the NEON instruction set, which utilizes the SIMD engine present in the ARM Cortex-A series of processors2. This is achieved through the use of ‘intrinsic’ functions added to the C/C++ code.

3.3 Processing Stages

Our system is a 'hybrid' system, in that it either uses feature detection and matching or optic flow tracking to find corresponding points between each input frame and the stored maps. Which method it uses depends on if the system is 'lost' or not. The stages depicted in Figure 1 are described below.

3.4 Image Decoding

On mobile devices it is often necessary to convert the image formats for use in AR. The input camera format and output screen format are usually incompatible and require a conversion step. The vast majority of smart phone and tablet cameras return images in some variation of the YUV format. The common YUV420 format is convenient for vision algorithms as the luminance data block can be used directly as a gray scale image. However, in order to display a colour image on the screen (using the Android API) or upload it as an OpenGL texture, it must be converted to an RGB format. A resolution of 640x480 is used throughout the system, which the camera is able to return directly without the need for software down-sampling. See Appendix A for conversion formulae

3.5 Feature Detection (performed in 'lost' hybrid mode)

Our system is based on the ORB descriptor, which is an oriented version of the BRIEF descriptor [17]. Both ORB and BRIEF are binary descriptors, meaning the descriptor itself is a binary string rather than a floating-point vector as is the case with the SIFT and SURF algorithms. To build the ORB descriptor, interest points (typically FAST corners) are detected in the image and a series of pixel intensity comparisons are carried out between the interest point and some distribution of nearby pixels (256 comparisons in the case of the 32-byte descriptor). A single bit is required to store the result of each comparison, and each comparison is very fast to compute. This makes ORB very well suited for applications where memory and computation resources are at a premium.

The most notable downside to the ORB descriptor is that it is not scale invariant. This can be compensated for by extracting features at different image scales in the input image and/or in the stored map but this was found to reduce execution speed too greatly. Instead, the system requires that maps be detected at or near their home position.

There are two general cases considered when extracting features from the new camera frame. First, if the system is not already localized to a specific map (the 'lost' state,) features are extracted from the entire image and then used to determine which map, if any, the camera is looking at second case occurs when the system is already localized to a map, but existing feature inliers are not being tracked (conditions for tracking may not have been met). In this case, features are only extracted from within the bounding box of the map as it appeared in the previous frame. This bounding box is updated each frame during the map update process.

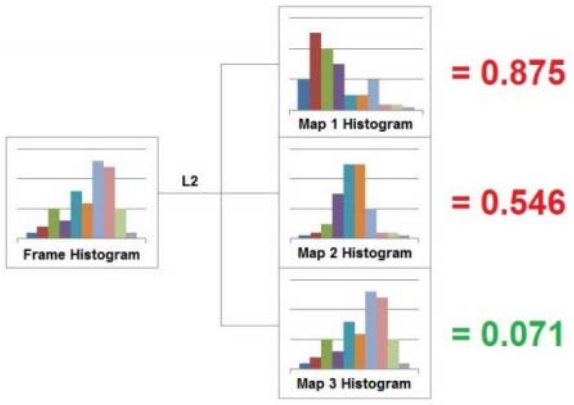

Figure 2: Histogram comparison for map recognition

3.6 Frame Lookup (performed in 'lost' hybrid mode)

Newly detected features in the camera frame are matched to the map set using a bag of words scheme. Normalized response histograms are created for both frame and map by matching their descriptors to a descriptor vocabulary. These histograms are compared using their L2 distance, and the map whose histogram gives the lowest distance is returned as the most likely to be present in the frame (Figure 2).

One notable shortcoming is that we do not apply a hard threshold on whether or not a map is present at all. Even if there is no map present in the scene, the system will always return the most likely candidate. The reason for this is that we could not find a value that reliably separated the two cases. The system relies on the descriptor matching phase described in the next section to determine if the map is actually present. Once the most suitable map is found, it is matched to the camera frame.

3.7 Descriptor Matching Behaviour

Comparing binary descriptors is done by calculating their Hamming distance. This consists of an exclusive-or operation followed by a population count of the result. This can be done quickly using built in XOR and POPCNT instructions, but we achieve a further speed increase by once again using the SIMD engine. Rather than iterating through the 256-bit descriptor one integer at a time (8 integers total), they are stored across two 128-bit registers and processed in only two parts. This yields an approximate threefold increase in speed, even over the POPCNT instruction.

Frame descriptors are first matched to the roots of the candidate map, and then to the branch whose root yielded Figure 2–Histogram comparison for map recognition the best score. Matches are filtered using two thresholds: an absolute threshold of 75 for the best Hamming score (this value was picked based on the distribution of true positive scores in [17]), and a relative threshold of 0.75 for the ratio between the best and second best matches. Additionally, a threshold of 20 is set as the minimum number of good matches that must be found for the map to be deemed detected. This number was determined empirically, based on the observed average number of false matches that occur when there is no map present in the frame. Using this threshold the system

ignores frames that do not contain a map, regardless of the best map returned in the lookup phase.

3.8 Pose Estimation Behaviour

If a suitable number of good matches are found, the system attempts to estimate the pose of the camera using the known 3D coordinates of the map and their corresponding 2D frame locations. We have found that improved performance can be achieved by using two different pose estimation algorithms, one for each of the hybrid modes ('lost' or 'tracking').When determining pose, two cases are considered, one for each mode of the hybrid system: pose from descriptor matches, and pose from tracked inliers (discussed below). Despite the various limits and thresholds applied during the matching process, it is still possible for false positives to be present. These bad matches can comprise a significant portion of the correspondences, and, if not rejected, degrade the accuracy of the recovered camera pose. The goal is to use only correct matches in the Calculation of the pose. To do this, the EPnP [18] technique is used inside a RANdom SAmple and Consensus (RANSAC) loop [19].

The maximum number of iterations was determined empirically. Since the RANSAC algorithm is not deterministic, the integrity of the camera pose depends largely on the probability of selecting four accurate correspondences within the maximum number of iterations. 5000 iterations may have an infinitesimal chance of failing to find inliers, but the running time would be far too high. Therefore, this probability is weighed against the iteration speed in order to find a middle ground between accuracy and efficiency. In practice, when the correct map is present in the camera image, the number of outliers was typically below 25%. However, under extreme conditions this number was observed to climb to 50% (or higher), so the maximum number of iterations was calculated based on a 50% outlier ratio. Additionally, to determine the number of iterations, the desired probability of selecting four good inliers was arbitrarily set to 99%. The number of iterations was set to 72 using Equation (3) in appendix B. There is one main downside to pose estimation using feature matches: The same features are not always detected in every frame, which means the set of inliers is always different. This causes the camera pose to differ slightly each frame, which manifests itself as jitter in the displayed augmentation. To combat this, an inlier tracking method is used.

3.9 Inlier Tracking Behaviour

In our system we utilize an optic flow [20] tracking mode, both for speed performance and to allow for a greater range of motion without losing pose. Inlier tracking begins when the number of inliers reaches 50% of the total map size or higher. This value was obtained by observing that the average number of matches under ideal conditions is typically two thirds of the total map size (66%), and the average number of inliers under ideal conditions is 75% or more, which combine for 50% or more of the total map size. This requires the camera to be moderately close to its original position when the map was created, but when tracking it is desirable to begin with as many inliers as possible.

When tracking begins, instead of extracting and matching new features at the start of the next frame, inliers from the previous frame are tracked in the new frame using optic flow. This provides a number of benefits. First, tracking existing features is computationally faster than extracting new features and matching them. Second, because these are inliers, they are already known to be accurate matches. Rather than performing RANSAC-based pose estimation on the tracked inliers, the system skips straight to least-squares pose estimation. This time, however, an iterative pose estimation method based on Levenberg-Marquardt optimization [21] present in OpenCV is used instead of EPnP. The reason for using the iterative method is that it produces more consistently robust results at extreme viewing angles. Although the iterative method is slower in general, performing least-squares iterative pose estimation is still much faster than using RANSAC based EPnP. Using optic flow and iterative pose estimation, the system is able to track the map plane until it is nearly perpendicular to the camera plane and still achieve a robust, stable pose.

This represents one of the contributions of this paper, the observation that a RANSAC loop not needed in the tracking mode if properly initialized, and that furthermore the iterative pose mode provides better visual stability for AR.

Ideally, every inlier in the previous frame will be tracked into the new frame. This is not realistic, however. Each new frame will result in fewer and fewer successful tracks due to occlusion; difficult viewing conditions, motion blur, or even camera noise. Eventually there will be too few tracks remaining to obtain a reliable pose. When this occurs, the system assumes the map has been lost, extracts new features, and goes back to the frame lookup step. The lower limit on the number of tracks is set to 20. This number is based solely on observation of augmentation stability.

3.10 Map Updation

The final process in the pipeline is to dynamically update the map. First, the projection matrix is computed from the pose and camera parameters. It is used to re-project the 3D bounding corners of the map into 2D image space to provide the region of interest used in feature detection.

Secondly, if features were extracted from the current frame, inlier descriptors are added to the map. In this way, each 3D world point in the map can have more than one descriptor associated with it. In order to prevent the map from growing overly large, the number of descriptors that can be associated with each point is limited. When the limit is reached, older descriptors are removed in favour of the newer ones with the exception of the original descriptor, which is never removed. This allows map features to be more consistently detected from different viewpoints.

3.11 Augment Portrayal

Once the map has been updated, the final task is to pass the recovered pose back to OpenGL to be used to create a model view matrix (Appendix C). With a model view matrix obtained via an accurate camera pose, any 3D augmentation can be drawn. With sufficient 3D graphics experience, one

could design complex augmentations that integrate seamlessly with their real world surroundings.

4. Conclusion

This paper presents an elaborated survey on mobile based augmented reality system designed to operate in real environment current generation mobile devices. Many past research issues have been highlighted and directions for future work have been suggested. Many open issues have been highlighted by the researchers such as dealing with optimisations, speed with dynamic scene.

References

[1] M. Fiala, "ARTag, a fiducial marker system using digital techniques," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 2005, pp. 590-596 vol. 2.
[2] M. Fiala and G. Roth, "Magic Lens Augmented Reality: Table-top and Augmentorium," presented at the ACM SIGGRAPH 2007 posters, San Diego, California, 2007.
[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in Computer Vision (ICCV), 2011 IEEE International Conference on, 2011, pp. 2564-2571.
[4] J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 29, pp. 1052-1067, 2007.
[5] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on, 2007, pp. 225-234.
[6] L. Taehee and T. Hollerer, "Hybrid Feature Tracking and User Interaction for Markerless Augmented Reality," in Virtual Reality Conference, 2008. VR '08. IEEE, 2008, pp. 145-152.
[7] D. G. Lowe, "Object recognition from local scale-invariant features," in Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, 1999, pp. 1150-1157 vol.2.
[8] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," Computer Vision–ECCV 2006, pp. 404-417, 2006.
[9] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," presented at the European Conference on Computer Vision, 2006.
[10] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose tracking from natural features on mobile phones," presented at the Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, 2008.
[11] W.-C. Chen, Y. Xiong, J. Gao, N. Gelfand, and R. Grzeszczuk, "Efficient Extraction of Robust Image Features on Mobile Devices," presented at the Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007.
[12] M. Mohring, C. Lessig, and O. Bimber, "Video See-Through AR on Consumer Cell-Phones," presented at the Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality, 2004.
[13] G. Klein and D. Murray, "Parallel Tracking and Mapping on a camera phone," in Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on, 2009, pp. 83-86.
[14] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Real-Time Detection and Tracking for Augmented Reality on Mobile Phones," Visualization and Computer Graphics, IEEE Transactions on, vol. 16, pp. 355-368, 2010.
[15] M. Muja and D. G. Lowe, "Fast Matching of Binary Features," in Computer and Robot Vision (CRV), 2012 Ninth Conference on,2012, pp. 404-410.
[16] T. P. Morgan. (2012). ARM Snags 95 Percent Of Smartphone Market, Eyes New Areas For Growth. Available: http://www.crn.com/news/components-peripherals/240003811/arm-snags-95-percent-of-smartphone-market-eyes-new-areas-for-growth.htm
[17] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," Computer Vision–ECCV 2010,pp. 778-792, 2010.
[18] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," International Journal of Computer Vision, vol. 81, pp. 155-166, 2009.
[19] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Commun. ACM, vol. 24, pp. 381-395, 1981.
[20] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," Carnegie Mellon University1991.
[21] K. Levenberg, "A Method for the Solution of Certain Non-Linear Problems in Least Squares," Quarterly of Applied Mathematics, pp. 164-168, 1944.
[22] W. Garage. (2012). OpenCV Change Log. Available: http://code.opencv.org/projects/opencv/wiki/ChangeLog
[23] J. Hruska. (2013). Nvidia's Tegra 4 demystified: 28nm, 72-core GPU, integrated LTE, and questionable power consumption.Available: http://www.extremetech.com/computing/144942-nvidias-tegra-4-demystified-28nm-72-core-gpu-integrated-lte-and-questionable-power-consumption

Author Profile

Shaik Abdul Gaffar Obtained the B.Tech degree in Computer Science and Engineering (CSE) from **Chalapathi Institute of Technology**, Mothadaka, Guntur District. At present I am persuing the M.Tech in Computer Science and Engineering (CSE) Department at **Guntur Engineering College**, Guntur.

Shaik Nyamathulla obtained the B.Tech. degree in Computer Science and Information Technology (CSIT) from **Hi-Point college of Engg and Tech,** Hyderabad in 2009 and M.Tech from Guntur Engineering College in 2013. He has 5 years of teaching experience and working in Computer Science and Engineering (CSE) Department at **Guntur Engineering College**, Guntur.