Global Localization and Tracking for Wearable Augmented Reality in Urban Environments

Thomas Calloway, Dalila B. Megherbi
Department of Electrical and Computer Engineering
CMINDS Research Center, UMASS Lowell
Lowell MA, USA

Hongsheng Zhang
Thales Visionix, Incorporated
InterSense Motion Tracking Department
Billerica MA, USA

***Abstract*—Reliably localizing and tracking displays moving relative to content in the physical world is one of the primary technical challenges facing all augmented reality systems. While significant progress has been made in recent years, all approaches remain limited to functioning only in certain environments and situations. Attempts to improve solution generality via additional sensors (e.g., depth sensors, multiple cameras) add significant size, weight and power to wearable solutions sensitive to these attributes. In this work, we propose an approach to tracking and localization using a single camera and inertial chip. Through a combination of visual-inertial navigation, point cloud mapping and dynamically correlating building faces and edges with sparse OpenStreetMap datasets, we achieved a typical global localization precision of less than 0.25 meters and 1 degree heading relative to the map. All motion tracking calculations are performed on a local mobile device with less than 10 milliseconds of latency while global localization and drift correction is performed remotely.**

Index Terms*—augmented reality, virtual reality, localization, motion tracking, SLAM, VINS, geo-referencing, wearable computing

I. Introduction

A. Background

Compelling visual augmented reality (AR) experiences require precision alignment of virtual information onto the physical world. This spatial and temporal alignment, called registration, is a key attribute of any AR system. Whether the mechanism of augmentation is a simple video feed or high end see-through display, poor registration breaks the mixed reality illusion and detracts from the quality and utility of any solution. Problems with registration can even lead to dizziness, nausea and other symptoms for users [1].

Augmented reality registration is typically achieved through a combination of tracking and localization. While these two terms are sometimes used interchangeably, they are often considered separate phenomenon that serve different purposes. Measuring a device's position and orientation (i.e., pose) relative to relevant objects in an environment is called tracking. It is typically a high rate process designed to minimize latency, jitter and other dynamic errors. The term localization commonly refers to the act of determining pose in a global reference frame, rather than a local or relative reference frame [2]. Localization can be important not only for establishing a global frame of reference, but also for correcting pose drift inherent in many motion tracking solutions like visual-inertial navigation or inertial dead reckoning.

Achieving AR registration that is accurate enough to fool the human brain and eyes in all situations and environments is a long way from being solved, particularly for see-through displays where latency and update rates are critical. Pose tracking accuracy specifications will continue driving towards sub-millimeter and sub-milliradian levels. While similar localization accuracies are desirable in many applications, another major challenge is simply achieving any pose localization at all in environments where detailed three-dimensional mapping has not been previously performed. With over half of the world's population currently living in cities, where GPS signals are easily occluded [3], urban environments may be a good place to focus research.

Tracking accurately in a shared global coordinate system could create tremendous opportunities for developers of registered augmented reality content. Not only will users be able to share AR experiences in public spaces, but the location of road intersections, relevant shops, people, smart devices and much more can be visible in a well registered augmented world.

B. Related Work

Early applications of precision motion tracking and localization to AR necessitated the installation of costly support infrastructure into controlled environments. Examples of professional systems include the installation of room mounted IR cameras [4], magnetic tracking sources [5] and ultrasonic emitters [6]. Though such systems remain relevant in high end training, visualization and motion capture markets, there is a clear trend towards the untethering of augmented reality systems from environments with motion tracking infrastructure [7] [8].

Perhaps the most popular method of providing low latency tracking in new, infrastructure-free environments is visual-inertial navigation (also called visual inertial odometry) [9] [10]. These approaches typically employ Kalman filters such as the extended Kalman filter (EKF) [11] and multi-state constraint Kalman filter (MSCKF) [12]. The EKF based solution from ETH Zurich described in [13] is particularly robust against rapid rotations. These Kalman filter based approaches are attractive due to their low computational complexity, while providing the low latency and jitter required in augmented reality applications. The primary weaknesses of visual-inertial navigation systems (VINS) are that they accumulate error over time and provide relative position and heading only. To address these deficiencies, some form of absolute localization to an external reference frame is needed.

The best known augmented reality systems incorporating untethered tracking and localization today are probably the Microsoft HoloLens and Google Tango devices. Both systems incorporate bleeding edge computer vision and sensor fusion techniques that merge information from cameras, inertial sensors and structured light depth sensors. While cameras and inertial sensor units (IMUs) handle the low latency tracking, IR projecting depth sensors are used to create three-dimensional maps of the surrounding environment. This mapping from inexpensive structured light depth sensors was pioneered at Microsoft in their seminal work on Kinect Fusion [14]. This work has been extensively studied, refined and extended over the past several years [15] [16] [17] [18].

The ability to create three-dimensional point cloud maps using only passive cameras is an attractive proposition, as the same wearable monocular camera used for visual-inertial navigation can used for (re)localization. Methods of creating point clouds from monocular video sequences can include direct methods such as those described in LSD-SLAM [19] or feature based methods such as those described in PTAM [20] and ORB-SLAM [21]. Some approaches to global localization have had success matching image features to three-dimensional models of nearby buildings [22].

C. Limitations of Related Work

All previous work on tracking and localization have noteworthy limitations. Depth sensors are bulky, power hungry, have limited range and don't work in direct sunlight. GPS data is inaccurate and drops out completely when indoors or in urban canyons. Finally, pose recovery from a database of three-dimensional models of geo-referenced content does not work if such data bases do not exist, are not up to date or are not freely available.

We achieved accurate AR registration to geo-referenced content in our previous work by tracking head motion inside of globally localized moving vehicles [23]. Unfortunately, such systems only function while the user is in a vehicle that is accurately tracking and reporting its global pose. Accurate vehicle tracking was only achieved with an expensive GPS/INS system installed on the dashboard.

D. Main Objectives & Contributions

The objective of this paper is to contribute to the state of the art in motion tracking and localization for pedestrian AR content registration. We use a new monocular vision-inertial motion tracking sensor from Thales Visionix to develop and present an algorithm for accurately determining the latitude, longitude and heading of an AR display in outdoor urban environments.

The sensor and tracking software from Thales Visionix provides a wearable sensor with minimal mobile computing power to track head (or tablet) pose at 200Hz with negligible latency. We use this system to create a sparse three-dimensional point cloud in a relative coordinate system. Our primary contribution is to then transform the tracking reference frame and local point clouds into a global (latitude, longitude and heading) coordinate space. This is done through a combination of scene recognition, building edge detection, point cloud analysis and correlation with two-dimensional map data downloaded in real time from OpenStreetMap (OSM).

II. METHOD

A. System Architecture

For algorithm development and testing we used the InertiaCam sensor from Thales Visionix. The device, shown in Figure 1 below, consists of a single monochrome global shutter camera and NavChip inertial measurement unit. The camera provides a 130^0 wide angle field of view image configured to run at 20Hz. The NavChip is pre-calibrated over temperature with an in-run gyro bias stability of 5^0/hr. The sensor is mounted to the back of a Microsoft Surface Pro tablet for data collection and experimentation.

Figure 1: InertiaCam monocular vision-inertial sensor module

Figure 2 below shows the top-level system architecture of our approach. A high rate visual-inertial navigation system runs locally on the tablet, computing position and orientation at 200Hz with less than 10 milliseconds of tracking latency. This pose tracking data feeds into a mapping module along with lower rate timestamped images. The mapping module is responsible for creating three-dimensional point clouds from this data, which are useful for (re)localization and tracker drift correction. Finally, a global localization module uses these point clouds and locally tracked keyframe images to estimate global pose in terms of latitude, longitude and absolute heading. All this information is needed to facilitate the accurate registration of globally localized augmented reality content.

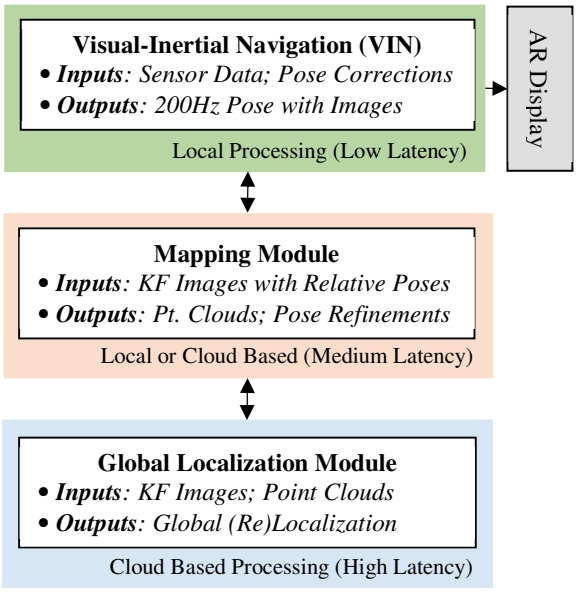

Figure 2: Tracking & localization system architecture

B. Front End Tracking & Mapping

The visual-inertial navigation system depicted at the top of Figure 2 implements a tightly coupled Extended Kalman Filter (EKF). The filter was tuned for and tested with the InertiaCam, but could be extended to any pre-calibrated monocular vision-inertial sensor in which the IMU and camera are synchronized.

The NavChip IMU outputs 3 axis angular rates $\tilde{\omega}_{ws}$ and accelerations $\tilde{a}_s$ in the inertial sensor frame *S* relative to the world frame *W*. The measurements are directly impacted by the sensor biases *b* and additive Gaussian noise *n* as described in (1) and (2) below. The orientation offset between the sensor and world frames is R_{ws} and the vector representing gravity in the world frame is g_w.

$$\tilde{\omega}_{ws}(t) = \omega_{ws}(t) + b_g(t) + n_g(t) \quad (1)$$

$$\tilde{a}_s(t) = R_{ws}^T(t)(a_w(t) - g_w) + b_a(t) + n_a(t) \quad (2)$$

Several other sources of measurement error are such as the scale factors and internal misalignments are already compensated in the NavChip, where factory calibration is applied. Important for augmented reality applications is the fact that the NavChip angular rate outputs do not saturate until reaching 2000^0/s. Integration of the orientation, velocity and position information are described in (3), where Δt is generally 5 milliseconds.

$$R(t+\Delta t) = R(t)\, Exp\left(\left(\tilde{\omega}(t) - b_g(t) - n_{gd(t)}\right)\Delta t\right)$$

$$v(t+\Delta t) = v(t) + g\Delta t + R(t)\left(\tilde{a}(t) - b_a(t) - n_{ad}(t)\right)\Delta t$$

$$P(t+\Delta t) = p(t) + v(t)\Delta t + \tfrac{1}{2}g\Delta t^2 + \tfrac{1}{2}R(t)\left(\tilde{a}(t) - b_a(t) + n_{ad}(t)\right)\Delta t^2 \quad (3)$$

While the primary motion tracking thread processes the full EKF cycle upon receipt of new images, a second thread propagates only the primary navigation states (i.e., no optical structure) through numerical integration with minimal computational complexity. This allows full six degree-of-freedom pose tracking to be calculated at 200Hz with under 10 milliseconds of total system latency. An additional layer of inertial prediction estimates future pose to reduce the perceived latency from 10 milliseconds to near zero.

Visual SLAM (VSLAM) is infamously fragile when a camera is subjected to quick motions, especially for fast rotations. Constant velocity motion models have been introduced to alleviate this issue, however rapid accelerations and decelerations violate the constant velocity model assumptions [21]. We therefore use VINS to fill the role of a real time true motion model. The benefits of this approach are three-fold. First, the mapping module is significantly more robust against rapid motions. Second, the computational cost resulting from an iterative motion search based on a poor motion model has been minimized. Third, correct global scale and orientation information is used.

Together with the covariance estimate, VINS poses are effectively combined with traditional bundle adjustment for optimal point cloud mapping. Immediately prior to analyzing the relevant region of the point cloud in the global localization module, we perform a full global bundle adjustment on all relevant points and keyframe locations using the g2o software framework [27].

C. Triggering Global Localization

With robust pose tracking and point cloud generation working reliably, we move onto the challenge of attempting to estimate the global pose of the system. The global localization module from Figure 2 receives as input keyframe images with local pose at a rate of about 1-2 Hz. It also receives the portion of the point cloud directly in front of the sensor. Because the images are monochrome 640 x 480 resolution and the point clouds are sparse, the amount of data needed to be sent over the network for real-time localization is already possible with today's 4G networks.

Figure 3 shows the top-level algorithm for the global localization module. Before any localization tasks are attempted, the image is first sent to Google's image analysis service. Only if the scene recognition AI determines that the keyframe is a picture of a "building" or "architecture" does the process proceed to the next steps. Future work should also take advantage of previous keyframe content and pose when making this determination. Also, color images would likely allow for more accurate scene identification.

Whenever building recognition succeeds on an incoming image, independent point cloud and map analysis modules are run. The point cloud analysis function is tasked with determining whether a vertical plane consistent with a building face is present. The street map analysis module fetches local building edge coordinates and determines the most likely building face that was detected in the image.

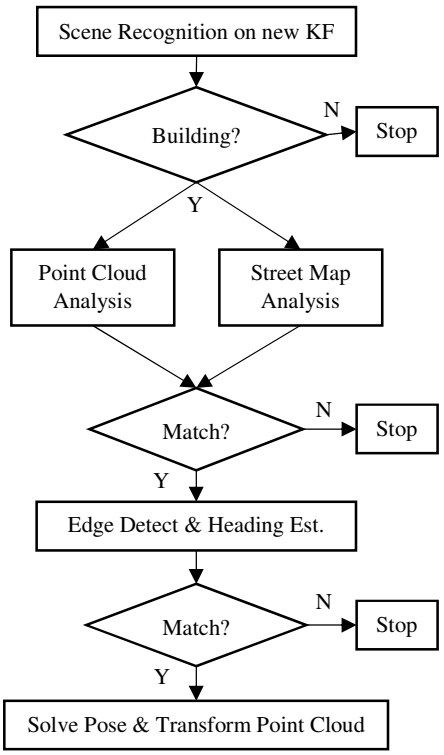

Figure 3: Top level global localization algorithm

As can be seen in Figure 4, the original point clouds generated by our mapping module are quite noisy to the human eye. To extract meaningful information from this point cloud, we employ the random sample consensus (RANSAC) algorithm to find the best fit plane located within

the data [24]. Through experimentation, we found that a rejection threshold of 0.3 meters provided the most accurate and reliable results, but this threshold may be adapted in future work depending on several factors. The right side of Figure 4 shows the result of removing outliers from the point cloud (the green dot on the far right represents the sensor location relative to the plane).

We now add the constraint that the building's external wall should be perpendicular with gravity. With n representing the normal vector of the plane and the vector pointing directly down with gravity known to be [0 0 1], we directly compute the perpendicularity of the plane in (4). If the plane is not perpendicular to within a given threshold (typically 3^0) then the keyframe is immediately rejected and the localization process halted.

$$\Theta = \left| cos^{-1} \frac{[0\ 0\ 1] \cdot n}{||[0\ 0\ 1]|| \cdot |n|} \right| - \frac{\pi}{2} \quad (4)$$

We realized during testing that the normal vector of the plane could either be directed into the building or out of the building. We compensate for this with the added constraint that the user is looking at the building from the outside. We then choose the normal vector for the plane that faces the same hemisphere as the user. The heading offset between the relative sensor pose and plane normal vector is then just the difference between the two.

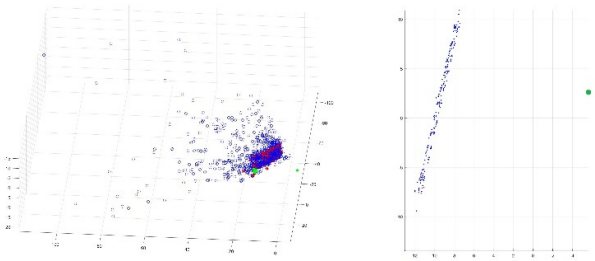

Figure 4: Original point cloud (left); processed and localized (right)

The street map analysis module uses the OpenStreetMap API to automatically fetch a small XML file containing the content immediately surrounding the current location. Figure 5 shows one such file displayed using the Web Mercator projection. The green circle represents the predicted position uncertainty resulting from the use of consumer GPS devices adjacent to a three-story building. The currently estimated location is not in the exact center of the image due to the limited resolution of the OpenStreetMap API region of interest.

After downloading the relevant street map data, we programmatically extract all building faces as a series of top-down line segments as shown in Figure 7. We then limit analysis to only those building faces which are not occluded. Correlating the remaining building line segments with the calculated point cloud plane and pose successfully eliminated all irrelevant building faces in our test dataset.

The output of the street map analysis module is a single line segment representing the building face most prevalent in the current keyframe image. Importantly, this single line segment contains all the information we need from a georeferenced database to compute the six degree of freedom pose of our sensor. Each end of the line segment gives us the latitude and longitude of the building edges while the direction of the line provides us with accurate heading information relative to true north.

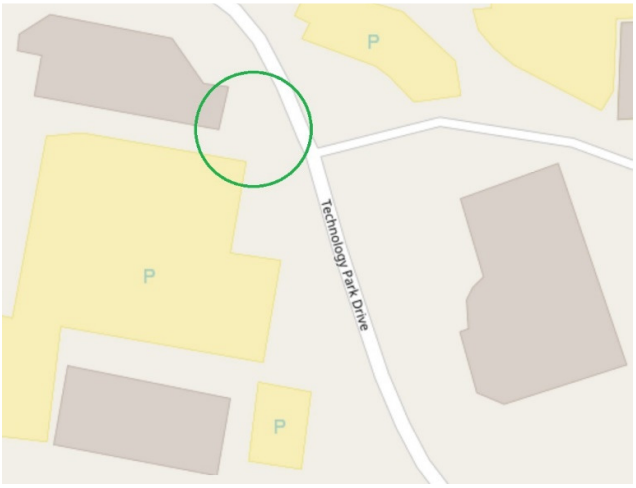

Figure 5: OpenStreetMap XML data dynamically fetched by our localization module; buildings are later extracted as line segments

Due to the noisy nature of the point cloud data, the plane resulting from RANSAC does not have well defined edges to correlate with the street map data. For this reason, we perform robust edge detection on the keyframe image to identify the edge of the building. After much experimentation, structured forest edge detection from Microsoft [26] was found to perform best. Figure 6 shows the results of this edge detection on a sample keyframe image from our test dataset.

Before performing edge detection, we first roll the image so that it is level with gravity using sensor orientation data provided by the VINS module. Once the image is level, we predict that the longest and cleanest vertical edge will correspond to a building corner as shown in Figure 6. We then calculate the angle from the center of the image to the edge of the building. This is done by transforming the average horizontal pixel coordinate of the edge to radians using the lens calibration parameters provided with the InertiaCam sensor module. If the predicted edge of the building found in the image roughly lines up with one edge of the point cloud plane, then we have a match. All that is left is to use the output of this and the previous modules to calculate the global pose of the sensor.

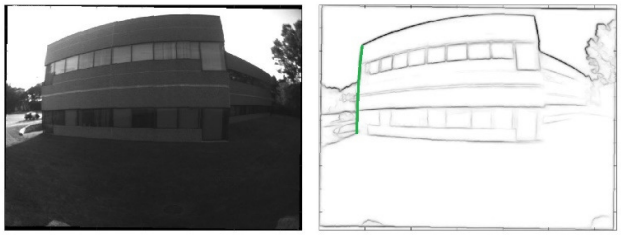

Figure 6: Roll-adjusted keyframe image (left) and edge detection (right)

D. Estimating Global Pose

The local point cloud and sensor pose data is stored and reported in units of meters with arbitrary origin. Until a global localization has occurred, the heading of the local coordinate system is also effectively arbitrary. Our first task is to rotate the point cloud so that local north lines up with true north. Since we already have the angle of the building face from our

street map analysis, finding the angle by which to rotate the point cloud coordinate system is trivial. Figure 4 show the results of this transformation. When we rotate the local point cloud coordinate system by the heading offset, we also adjust the sensor heading by the same mount. We have now solved for the heading of our augmented reality system relative to true north.

Now that we have solved for global heading of the sensor and rotated our local point cloud coordinate system to ground truth, our final task is to find the latitude and longitude of the sensor. Given that the original point cloud is noisy, we ignore individual points and instead use the output of our previous calculations. We draw a ray directed out from our sensor offset by the angle determined by our edge detection module. The intersection of our robust plane and this new ray is determined to be the edge of the building.

The latitude and longitude of the building edge is known, as are the distance and direction of the sensor with respect to the building edge. With this information, we can directly compute the latitude and longitude of the augmented reality sensor in (5) and (6). The horizontal and vertical distances from the sensor to the building edge are given as Δx and Δy, and the radius of the Earth is r. Because the relative distances involved are so small, errors resulting from treating linear distances as arc lengths are negligible. We note that for larger distances, the Haversine formula would be preferable due to the improved spherical approximations.

$$Lat_s = Lat_{bldg} + \frac{\Delta x}{r} \quad (5)$$

$$Lon_s = Lon_{bldg} + \frac{\Delta y}{r\cos(Lat_{bldg})} \quad (6)$$

For altitude, we simply assume that we are walking on the ground and interpolate subsampled altitude data to put ourselves above the ground by the height of the user's eyes as was done in [23]. Future work could detect the distance from points on the ground to get height more precisely.

With this, we have solved for all six degrees of freedom using the following methods:

1) VINS provides us with the ***pitch*** and ***roll*** values
2) Alignment of the RANSAC estimated plane to the building face solves for the absolute ***heading***
3) The intersection of the RANSAC calculated plane with building edge and ray provided by the 2D image allows us to calculate ***latitude*** and ***longitude***
4) Interpolating public altitude data from online sources and adding nominal display height above the ground provides us with ***altitude***

E. Test Methodology

To test our algorithm, we collected and recorded a fresh set of data using our tracking and mapping "front end". Our dataset consisted of a three-dimensional point cloud and sequence of 50 keyframe images that were extracted from the front end at about 1 Hz. Each keyframe image was timestamped and tagged with a corresponding relative pose obtained from the tracking module.

We tested our approach by running the global localization algorithm offline in the MATLAB development environment. In the next section, we present and discuss the results of applying the algorithms to each of the 50 keyframe images.

III. RESULTS

Recall from Figure 3 that there were many points at which the global localization algorithms could fail to confidently solve for pose. TABLE I shows that the first major decision point (scene recognition) failed 50% of the keyframe images. While many of these images were genuinely inadequate, many of them clearly showed a nearby building with a distinct edge. We hypothesize that a transition to color images could greatly improve the scene recognition stage.

One positive outcome of the first module failing many of the images was that the remaining images were all very clean. Thus, all the RANSAC estimations resulted in surfaces that were highly perpendicular to gravity. It is likely that occlusions such as vehicles or pedestrians would cause additional failures here, as would a failure to "strafe" the building for several feet while creating an adequate point cloud.

Detecting edges that correlated properly with the edge of the building plane failed in almost 10% of the keyframe images. In future work, we might first analyze the plane and then search for edges specifically around a horizontal region of interest. Our approach to extracting the largest gradient from the edge detected images may also be improved.

Because the tracking front end only drifts approximately 1 meter for every 100 meters of travel, not every keyframe needs to succeed. In fact, with the local point clouds correcting for tracking drift whenever a pedestrian lingers in a scene, we need only correct for pose every 50 meters of travel or so to maintain optimal accuracy.

TABLE I: 42% of keyframe images successfully localized

	KF Image Count
Total Image Count	***50***
Recognition Failure	25 (50%)
RANSAC Failure	0 (0%)
Edge Detection Failure	4 (8%)
Success	21 (42%)

The global localization error was dominated by RANSAC, which was a result of non-Gaussian noise in the local point cloud. TABLE II below shows this noise reflected in the pose output. We ran the same keyframe through the localization algorithm for 100 trials and obtained a typical error of less than 1^0 and 0.25 meters. While the accuracy of the OpenStreetMap latitude and longitude information is not precisely known, subjective registration can be improved by using the same data source to define content and localize.

TABLE II: Estimated pose errors (100 trials)

	1σ Error	**Max Error**
Heading	0.85^0	1.77^0
Position	0.23 meters	0.44 meters

Note that due to the nature of our approach, errors in heading directly translate into position error. The further a pedestrian is from a building edge, the more this error will grow. Fortunately, plane estimation and image edge detection provided far more reliable and accurate information than matching single three-dimensional points.

Figure 7 shows an example map of dynamically downloaded and extracted building edges, simulated GPS uncertainty, and the calculated global pose with heading. Even with a more accurate GPS receiver, the occlusion problems prevalent in urban canyons [3] will not allow satellite GPS systems to reach the position accuracy that we achieved in this work. Heading is an even greater challenge using consumer hardware, with magnetometers being highly susceptible to external electromagnetic interference.

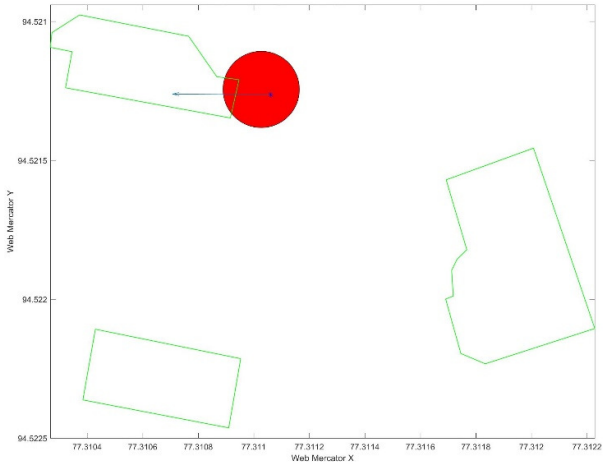

Figure 7: Map of nearby buildings as line segments with simulated GPS uncertainty (red) and estimated pose output (blue asterisk and arrow)

IV. Conclusions

In this work, we presented a method for tracking, mapping and globally localizing an augmented reality system using a single wearable camera and inertial chip. In our test dataset, we calculated position to within 0.25 meters and heading to within 1^0. These results are much better than today's GPS / magnetometer approaches can achieve, particularly in urban environments where nearby buildings may occlude GPS satellites.

Although our approach was successful, tracking and localization for "augmented reality anywhere" will require many such algorithms and approaches working in tandem to be as robust and universal as possible. Our own future work will likely involve more thorough experimentation and generalization of the localization algorithms. Augmented reality systems are continuing to improve at a rapid pace, but much more research will be needed before the technology can be considered perfected.

References

[1] Drascic, David, and Paul Milgram. "Perceptual issues in augmented reality." Electronic Imaging: Science & Technology. International Society for Optics and Photonics, 1996.

[2] Gervautz, Michael, and Dieter Schmalstieg. "Anywhere interfaces using handheld augmented reality." Computer 45.7 (2012): 26-31.

[3] Cui, Youjing, and Shuzhi Sam Ge. "Autonomous vehicle positioning with GPS in urban canyon environments." IEEE transactions on robotics and automation 19.1 (2003): 15-25.

[4] Ribo, Miguel, Axel Pinz, and Anton L. Fuhrmann. "A new optical tracking system for virtual and augmented reality applications." Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE. Vol. 3. IEEE, 2001.

[5] Livingston, Mark A. "Magnetic tracker calibration for improved augmented reality registration." Presence: Teleoperators and Virtual Environments 6.5 (1997): 532-546.

[6] Welch, Greg, and Eric Foxlin. "Motion tracking: No silver bullet, but a respectable arsenal." IEEE Computer graphics and Applications 22.6 (2002): 24-38.

[7] Billinghurst, Mark, Adrian Clark, and Gun Lee. "A survey of augmented reality." Foundations and Trends® Human–Computer Interaction 8.2-3 (2015): 73-272.

[8] Van Krevelen, D., and R. Poelman. "Augmented Reality: Technologies, Applications, and Limitations." (2007).

[9] Jones, Eagle S., and Stefano Soatto. "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach." The International Journal of Robotics Research 30.4 (2011): 407-430.

[10] Li, Mingyang, and Anastasios I. Mourikis. "High-precision, consistent EKF-based visual-inertial odometry." The International Journal of Robotics Research 32.6 (2013): 690-711.

[11] Fujii, Keisuke. "Extended kalman filter." Refernce Manual (2013).

[12] Mourikis, Anastasios I., and Stergios I. Roumeliotis. "A multi-state constraint Kalman filter for vision-aided inertial navigation." Robotics and automation, 2007 IEEE international conference on. IEEE, 2007.

[13] Bloesch, Michael, et al. "Robust visual inertial odometry using a direct EKF-based approach." Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. IEEE, 2015.

[14] Newcombe, Richard A., et al. "KinectFusion: Real-time dense surface mapping and tracking." Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on. IEEE, 2011.

[15] Whelan, Thomas, et al. "Kintinuous: Spatially extended kinectfusion." (2012).

[16] Pagliari, D., et al. "Kinect Fusion improvement using depth camera calibration." The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 40.5 (2014): 479.

[17] Nakayama, Yusuke, et al. "Accurate camera pose estimation for kinectfusion based on line segment matching by LEHF." Pattern Recognition (ICPR), 2014 22nd International Conference on. IEEE, 2014.

[18] Nguyen, Chuong V., Shahram Izadi, and David Lovell. "Modeling kinect sensor noise for improved 3d reconstruction and tracking." 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on. IEEE, 2012.

[19] Engel, Jakob, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-scale direct monocular SLAM." European Conference on Computer Vision. Springer International Publishing, 2014.

[20] Klein, Georg, and David Murray. "Parallel tracking and mapping for small AR workspaces." Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on. IEEE, 2007.

[21] Mur-Artal, Raul, Jose Maria Martinez Montiel, and Juan D. Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system." IEEE Transactions on Robotics 31.5 (2015): 1147-1163.

[22] Karlekar, Jayashree, et al. "Model-based localization and drift-free user tracking for outdoor augmented reality." Multimedia and Expo (ICME), 2010 IEEE International Conference on. IEEE, 2010.

[23] Foxlin, Eric, Thomas Calloway, and Hongsheng Zhang. "Improved registration for vehicular AR using auto-harmonization." Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on. IEEE, 2014.

[24] Qian, Xiangfei, and Cang Ye. "NCC-RANSAC: a fast plane extraction method for 3-D range data segmentation." IEEE transactions on cybernetics 44.12 (2014): 2771-2783.

[25] Haklay, Mordechai, and Patrick Weber. "Openstreetmap: User-generated street maps." IEEE Pervasive Computing 7.4 (2008): 12-18.

[26] Dollár, Piotr, and C. Lawrence Zitnick. "Structured forests for fast edge detection." Proceedings of the IEEE International Conference on Computer Vision. 2013.

[27] Kümmerle, Rainer, et al. "g 2 o: A general framework for graph optimization." Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011.