Efficient Computation of Absolute Pose for Gravity-Aware Augmented Reality

Chris Sweeney[1] John Flynn[2] Benjamin Nuernberger[1] Matthew Turk[1] Tobias Höllerer[1]
[1]University of California Santa Barbara [2] Google, Inc.
{cmsweeney, bnuernberger, mturk, holl}@cs.ucsb.edu jflynn@google.com

ABSTRACT

We propose a novel formulation for determining the absolute pose of a single or multi-camera system given a known vertical direction. The vertical direction may be easily obtained by detecting the vertical vanishing points with computer vision techniques, or with the aid of IMU sensor measurements from a smartphone. Our solver is general and able to compute absolute camera pose from two 2D-3D correspondences for single or multi-camera systems. We run several synthetic experiments that demonstrate our algorithm's improved robustness to image and IMU noise compared to the current state of the art. Additionally, we run an image localization experiment that demonstrates the accuracy of our algorithm in real-world scenarios. Finally, we show that our algorithm provides increased performance for real-time model-based tracking compared to solvers that do not utilize the vertical direction and show our algorithm in use with an augmented reality application running on a Google Tango tablet.

Keywords: Absolute pose, multi-camera system, gravity-aware augmented reality, inertial sensor, model-based tracking

1 INTRODUCTION

Determining a camera's position and orientation is a key requirement in augmented reality applications, a crucial step in simultaneous localization and mapping (SLAM) algorithms, and overall a major area of focus in computer vision [7, 18, 19]. Absolute pose methods utilize 2D-3D correspondences between image pixels and 3D points in a known scene to determine the full 6-degree-of-freedom (d.o.f.) camera pose. These methods are often more efficient than relative pose methods and have been demonstrated to improve accuracy and reduce camera pose jitter in SLAM [3].

The estimation of a camera's pose can benefit from motion priors provided by inertial sensors. Knowledge of the camera orientation, for instance, can reduce the number of unknown d.o.f. and thus reduce the number of unknown camera pose parameters that must be estimated. This results in a more efficient camera pose computation with improved accuracy. For applications on mobile devices with hardware constraints, developing camera pose estimation algorithms that utilize motion priors from inertial sensors is of great theoretical and practical relevance [9, 10].

We present an efficient method to estimate the absolute pose of a single or multi-camera system by utilizing partial knowledge of the camera orientation. Given the angle-axis representation of $SO(3)$ rotations, we assume knowledge of the rotation axis and solve for the unknown rotation angle about that axis. This removes two d.o.f. from the rotation and we are left to solve for the single remaining d.o.f. in the rotation and the unknown position. In applications of our methods, in which we make use of gravity sensors, the remaining degree of freedom in the rotation corresponds to a rotation angle about the vertical direction (*c.f.* Figure 1). Our formulation reduces to solving a quadratic equation to determine the remaining unknown camera pose parameters and utilizes the same formulation for single and multi-camera systems.

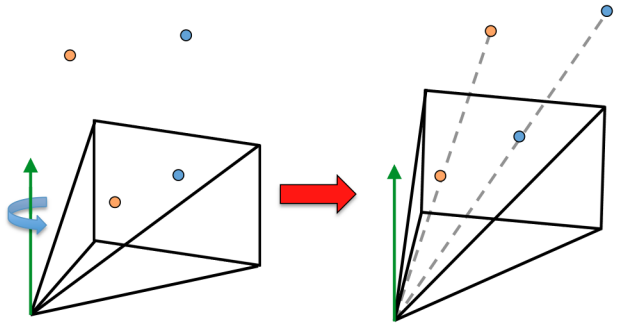

Figure 1: Our methods solve for the 6 d.o.f. pose given two 2D-3D correspondences (left) with the vertical direction (green) aligned to $[0,\ 1,\ 0]^\top$. This amounts to solving for a rotation angle about the vertical direction along with the camera translation such that the 3D points reproject to the 2D image points (right). Our solver method is general and works for single or multi-camera systems.

Our method allows for any type of partial rotation to be used. A partial rotation in the form of a rotation axis can be accurately determined through a variety of methods [2, 5, 8, 12, 13]. In this paper, we assume that the vertical direction is known, and use this as our axis of rotation after aligning it to the "up" direction (*e.g.*, $[0,\ 1,\ 0]^\top$). We use the terms "gravity direction" and "vertical direction" interchangeably since these directional vectors are parallel. The vertical direction may be conveniently obtained on most smartphone devices from the the inertial sensors and many smartphones even provide an API to directly obtain the gravity direction relative to the camera. The gravity vector may then be used to align the vertical direction to $[0,\ 1,\ 0]^\top$ (*c.f.* Figure 2). Alternatively, computer vision techniques may be used to detect vertical vanishing points [16] for determining the vertical direction. The fact that high quality motion priors may be conveniently retrieved from mobile devices places an importance on camera pose algorithms that utilize such sensor measurements, especially in the areas of computer vision, SLAM, and robotics.

In this paper we propose a novel 2-point algorithm for determining the absolute pose of a single or multi-camera system given prior knowledge of the camera orientation. While the single-camera case has been solved previously [8], our formulation presents a new formulation that is generalizes to multi-camera systems with the exact same expression. Solving for the camera pose of a multi-camera system given partial knowledge of the rotation has not been previously solved. Our formulations are simple and produce constraints that can be solved with a simple quadratic equation for the single or multi-camera case.

After discussing related work, we will describe the solution

Figure 2: The gravity vector (green) may be obtained from sensor measurements on common smartphone devices. By aligning the gravity vector to the known gravity direction $[0, -1, 0]^\top$, we remove 2 d.o.f. from the unknown rotation to create a simplified absolute pose method.

method in detail. Then, we provide a thorough analysis of the performance of our algorithms compared to the current state of the art. We perform synthetic experiments to measure the robustness of our algorithms to image noise and noise in the vertical direction. We then perform two experiments with real data. The first, a localization experiment, measures the accuracy of our method with single and multi-camera setups. Second, we measure the camera position error when using our algorithm for model-based tracking on a SLAM dataset with ground truth camera positions. Finally, we demonstrate the applicability of our algorithm with an augmented reality application running on a Google Tango tablet. Our methods are available online as open source C++ software[1].

2 Related Work

Our work builds on previous work in computer vision where inertial sensor measurements are used to enhance camera pose estimation [6, 15]. Interest in these methods has increased as the number of devices equipped with inertial sensors has increased. The limited computational power of devices such as smartphones and micro aerial vehicles places an increased importance on using as much sensory information as is available to limit the computational resources needed for visual tracking. Using knowledge of the vertical direction to compute the relative pose of two cameras from 2D-2D correspondences has been solved for single and multi-camera systems [2, 12, 22].

Computing the absolute pose of a camera determines the camera orientation and position relative to a known scene given image points and their corresponding 3D points. The Perspective 3-Point (P3P) algorithm is the method of choice, and Kneip *et al*. [7] propose a highly accurate and stable method that is an order of magnitude faster than previous methods. For multi-camera systems, Nistér and Stewénius [19] proposed a 3-point algorithm for absolute pose. When a rotation axis between camera and world coordinate systems is known, two d.o.f. are removed and the minimum number of correspondences required reduces from three to two. Kukelova *et al*. [8] derive a 2-point algorithm for perspective cameras with a known gravity vector that is able to solve for absolute camera pose from by solving a quadratic equation and a small linear system. Our 2-point algorithm has a more general formulation that is suitable for both single and multi-camera systems, is more accurate in the presence of noise, and is much more efficient, as demonstrated in our experiments.

A compact representation for multi-camera systems was first introduced by Grossberg and Nayar [4] as the "generalized camera model." The generalized camera model has since become the standard camera model for multi-camera and panoramic-camera setups

[1]Each algorithm has been incorporated as part of the Theia library for multiview geometry[21]. Documentation and source code can be found at `http://www.theia-sfm.org`

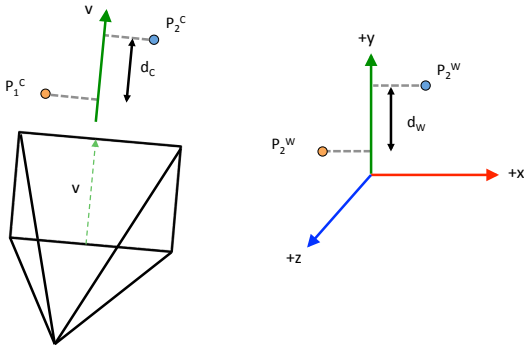

Figure 3: We form a constraint for the absolute pose problem by projecting the 3D points in the camera coordinate system (left) and world coordinate system (right) onto the gravity vector (green). The distance between the 3D points is the same in both coordinate systems when projected onto the gravity vector, thus $d_c = d_w$ as explained in Eq. (3).

[6, 20]. The generalized camera model gives highly accurate and stable motion estimation because of the potentially wide visual coverage. To our knowledge, the 2 point algorithm presented here is the first algorithm to compute absolute pose for generalized cameras (*i.e*., multi-camera systems) given partial knowledge of the rotation.

In SLAM, Kurz and Benhimane [9] introduce and discuss the use of Gravity-aligned and Gravity-rectified feature detectors in Augmented Reality applications. This work is later extended to a more detailed evaluation of gravity-aware methods and a full gravity-aware SLAM pipeline [10]. They present methods for using the gravity vector for feature extraction and matching in addition to image rectification. The authors show that having knowledge of the gravity vector (from sensor measurements or computer vision) can greatly enhance augmentations by allowing them to display affects of gravity (*e.g*., the surface of water remains parallel to the ground even as its container is moved). The tracking method that they present, however, is a planar template tracking method and the authors indeed acknowledge the need for 3D tracking that utilizes the gravity vector. The methods presented in this paper aim to fill that need. Our focus is on the estimation of the absolute pose of a camera rather than frame-to-frame relative tracking, though our method is efficient enough to enable direct model-based tracking if feature matches can be retrieved quickly with an efficient search structure (Secton 4.5). In the following section, we derive a solution to the absolute pose problem for generalized cameras and show that the case of perspective cameras is a special case of this general formulation.

3 Estimating Absolute Pose with a Known Axis of Rotation

Our goal is to compute the camera rotation and translation given two 3D points P_1^W, P_2^W in some unknown (world) coordinate system and their corresponding image rays with unit directions p_1 and p_2 with ray origins at q_1 and q_2 in some generalized camera. Assuming that the depths of the points in the camera coordinate system are λ_1 and λ_2, then the 3D points in the camera coordinate system can be written as $\lambda_1 p_1 + q_1$ and $\lambda_2 p_2 + q_2$. Our goal is to solve for the unknown rotation angle α about the rotation axis v and the unknown translation t such that,

$$\lambda_i p_i + q_i = R(\alpha, v) P_i^W + t \qquad i = 1, 2. \tag{1}$$

We will first solve for the unknown depths λ_1 and λ_2. The transformation of the 3D points from camera coordinate system to world

coordinate system can then be solved to determine α and t.

To solve for the unknown depths, we utilize two constraints based on the relationship between the two 3D points. The first constraint is given by the requirement that the distance between the two points must be the same in both camera and world coordinate systems:

$$||P_1^W - P_2^W|| = ||(\lambda_1 p_1 + q_1) - (\lambda_2 p_2 + q_2)||. \quad (2)$$

Given that the rotation is only around the axis v, the second constraint arises by noting that vector between the two points will be the same in each coordinate system when projected on to the axis v:

$$(P_1^W - P_2^W) \cdot v = [(\lambda_1 p_1 + q_1) - (\lambda_2 p_2 + q_2)] \cdot v. \quad (3)$$

This constraint is illustrated in Figure 3 and can be simplified so that λ_1 is expressed in terms of λ_2 in the form:

$$\lambda_1 = m + \lambda_2 n, \quad (4)$$

where,

$$m = \frac{[(P_1^W - q_1) - (P_2^W - q_2)] \cdot v}{p_1 \cdot v},$$
$$n = \frac{p_2 \cdot v}{p_1 \cdot v}.$$

Eq. (4) can then be substituted into Eq. (2), leading to an easily solvable quadratic in λ_2. For the special case of perspective cameras $q_1 = q_2$ and m is simply:

$$m = \frac{(P_1^W - P_2^W) \cdot v}{p_1 \cdot v}.$$

Once the depths are known we simply need to align the 3D points from the camera coordinate system to the points in the world coordinate system. Let us denote the 3D points in the camera coordinate system by

$$P_i^C = \lambda_i p_i + q_i \qquad i = 1, 2. \quad (5)$$

We compute the rotation angle α such that

$$R(\alpha, v)\overline{(P_1^C - P_2^C)} = \overline{(P_1^W - P_2^W)}, \quad (6)$$

where, given a vector x the unit-norm vector is denoted as $\bar{x}$. To solve for α, we project the vector between the two points on to the plane that is normal to the axis (*c.f.* Figure 4). The angle between the two projected vectors is the angle α which a corresponds to the rotation angle about the axis v. After applying the rotation to the points P_i^W, the translation t is simply:

$$t = P_1^C - R(\alpha, v)P_1^W = P_2^C - R(\alpha, v)P_2^W \quad (7)$$

It should be noted, however, that our algorithm is degenerate when the two 3D points project to the same location on the vertical axis. That is, when the line $P_1^W - P_2^W$ is orthogonal to the vertical axis. In this scenario, it is clear to see that Eq. 3 deduces to an amiguous $0 = 0$ constraint. This is an extremely rare case when using real data and we did not find this degeneracy to affect performance in our experiments.

4 Experiments

We conducted several experiments using synthetic and real data to measure the performance of our algorithm. First, we measure the robustness to image and IMU noise on synthetic experiments. Then we perform a localization experiment to showcase the accuracy of our method on real test data for single and multi-camera systems. Next, we use our algorithm for model-based tracking in a SLAM sequence where ground truth camera poses were captured. Our method provides stable, efficient tracking throughout the sequence. Finally, we demonstrate the applicability of our method with an augmented reality application running on Google Tango.

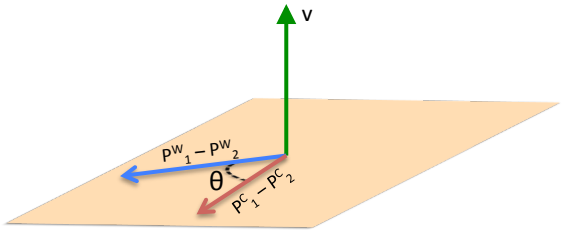

Figure 4: To solve for the unknown rotation angle α around the gravity vector (green) we project the lines connecting the 3D points on to the plane formed by gravity vector. The angle between the projected lines is the unknown rotation angle we seek to solve.

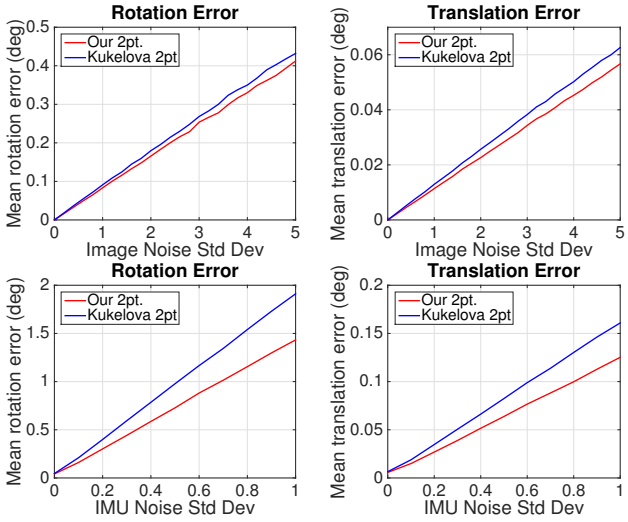

Figure 5: We compare our algorithm with Kukelova *et al*. [8] on synthetic experiments. **Top:** We increase the level of gaussian pixel noise and compute the camera pose estimation error. Our algorithm exhibits similar robustness to image pixel noise as [8]. **Bottom:** As synthetic noise in the vertical direction increases, our algorithm exhibits better performance than [8]. This makes it a good candidate for real-world applications where noise in the vertical direction is expected to be less than 1 degree.

4.1 Robustness to image noise

We performed synthetic experiments to evaluate the performance of our algorithms as image noise increases. The synthetic data consists of 1,000 3D points that are randomly distributed uniformly in a cube. The camera position is fixed at $(0,0,6)$ looking toward the origin with 3D points distributed uniformly in a $4 \times 4 \times 4$ cube centered around the origin. All experiments were performed on a Macbook Pro with a 2GHz processor and 8GB of RAM. We compare our algorithms to the state of the art solution of Kukelova *et al*. [8]. Note that we only include the results of experiments with a single camera, as we found the multi-camera method to have exactly the same performance since it is derived from the same formulation.

To measure the accuracy and robustness of our methods, we performed an experiment on synthetic data with Gaussian pixel noise ranging from 0 to 5 pixels. We performed 1,000 trials at increasing levels of pixel noise and measured the camera rotation and translation errors. The results are shown in Figure 5. Our algorithm performs slightly better than [8]. This is because utilizing the axis of rotation has been proven to be slightly more robust to noise [22].

Figure 6: Example images from the Metaio Outdoor Localization dataset [11]. Images have varying lighting, position and rotation.

4.2 Robustness to rotation axis noise

In order for our method to be of practical use, it must be robust to noise in the vertical direction (*i.e.*, the axis of rotation). To measure the robustness to the vertical direction, we simulate an IMU and add increasing amounts of noise within the the range of expected operating noise of sensors found on common devices today (less than 0.5 degrees [8]). We perform 1000 trials at each level of IMU noise and 0.5 pixels of image noise for all trials. We use the same camera and point configuration as the image noise experiment. Similar to the image noise experiments, our algorithms are slightly more robust to IMU noise than [8]. This indicates that our algorithms are well-suited for mobile devices and other common cameras that are equipped with IMU sensors that can expect to have less than 1 degree of IMU noise.

4.3 Computational complexity

Our algorithm is extremely efficient and only requires solving a single quadratic equation. The algorithm of Kukelova *et al*. [8] also solves a quadratic equation; however, their method also requires solving a 4x4 linear system and as a result is much slower. We ran 10^5 trials with the described configuration for each algorithm using C++ versions of all algorithms for optimal run-time efficiency over all trials. The algorithm of [8] executed with a mean time of 6 μs while our algorithm ran in just 5 ns – a speedup of over 3 orders of magnitude. While both methods are fast, this speedup is significant in the context of mobile devices where hardware resources are more constrained. Further, the extreme efficiency speaks to the overall simplicity of our algorithm.

4.4 SfM Localization with an Apple iPhone 4

We demonstrate that our algorithms are suitable for current mobile devices by conducting experiments with real image and sensory data from the Metaio Outdoor Dataset[2] [11]. This dataset consists of images and corresponding sensor measurements from an iPhone 4 with 6 degree of freedom ground truth poses. We obtain the gravity vector with measurements taken directly from the iPhone 4, and align that vector to $[0,\ -1,\ 0]^\top$. After aligning each image such that the gravity direction corresponds to $[0,\ -1,\ 0]^\top$, we use this vector as the axis of known rotation for our algorithms. We created an SfM reconstruction from the dataset, excluding 100 images that serve as our query images. We localize the remaining 100 images using approximate nearest neighbor correspondences from SIFT descriptors to establish 2D-3D correspondences. To test a multi-camera system with our method, we chose 100 unique pairs of images from the query image set and treated them as a rigid multi-camera rig. To establish correspondences for the multi-camera case we used SIFT matches from both images. All localization was performed in a standard RANSAC [1] scheme.

[2]http://www.metaio.com/research

Table 1: The table below shows results for our localization experiment. We give the mean position error, timing, and the number of RANSAC iterations required to compute a solution with 99.9% confidence. The reported timing only considers the time of RANSAC estimation and does not include the approximate nearest neighbor search. Our algorithms give the best performance for all metrics The position error of localization is smallest when using a multi-camera rig.

Algorithm	Pos. Error (cm)	Time (s)	RANSAC Its
Ours	5.2	0.006	154
Ours (multi-cam)	3.7	0.006	131
Kukelova [8]	6.6	0.014	172
P3P [7]	8.2	0.021	437

We were able to verify each of our algorithms against the ground truth poses provided in the dataset. As shown in Table 1, our algorithm outperforms the current state-of-the-art method of Kukelova [8] in terms of accuracy and efficiency. Our algorithm has a 20-45% increase in accuracy and a speedup of more than $2\times$ compared to [8]. This indicates that our algorithm is an excellent candidate for use on mobile devices. We include results for P3P [7] simply as a baseline though, as expected, it does not perform as well as the methods that utilize additional information from the gravity sensor.

4.5 Model-Based Tracking

To demonstrate the viability of our algorithm for using in augmented reality, we used our algorithm for "direct" model-based tracking. We call this method "direct" because we localize each frame to a reference 3D model in real-time. We captured a video sequence with a Google Tango Tablet while recording the ground truth camera positions with a highly accurate Vicon tracker [3]. We chose to only demonstrate our algorithm on a single-camera system since the multi-camera method was shown to have equivalent or better performance on synthetic and real data (*c.f*. Section 4.1 - 4.4). For each video frame, we extract the features and match them with 3D points in a previously constructed structure-from-motion model. We use use FLANN [17] for fast approximate nearest neighbor searches. We use sensor measurements directly from the Tango device to align the gravity direction of each frame to $[0,\ -1,\ 0]^\top$. The 2D-3D matches are used in a RANSAC loop to estimate the absolute pose of each video frame with respect to the 3D model.

The direct model-based tracking requires 26 milliseconds per frame (or roughly 35-40 frames per second) on average for pose estimation (*i.e.*, timing does not include feature extraction) when run on a 2011 Macbook Pro with a 2GHz processor. Typically the approximate nearest neighbor search is too slow for real-time methods; however, we are able to achieve real-time performance by sacrificing accuracy for speed in the nearest neighbor search. Lower accuracy in the matching will results in a lower ratio of inliers in the matches which will require more RANSAC iterations to compute a good pose. Given the exceptional speed of our method (3 orders of magnitude faster than [8]. *c.f*. Section 4.3), this is an acceptable trade-off.

[3]The reported position accuracy of the Vicon tracker is sub-millimeter.

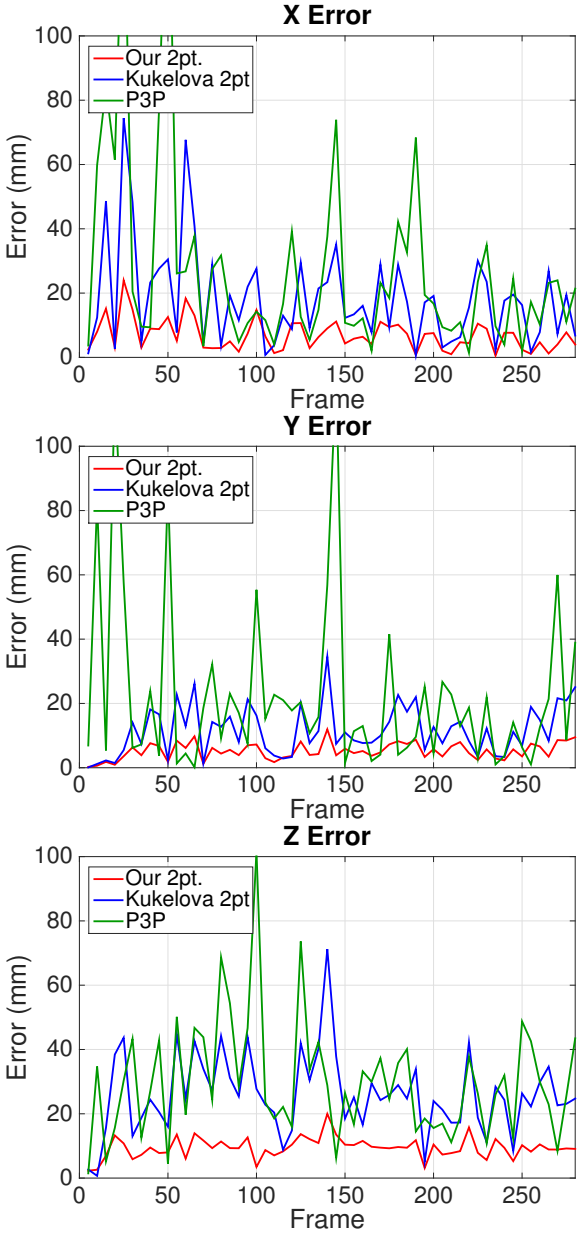

Figure 7: Position errors from direct model-based tracking with SLAM. We used a Vicon tracking system to obtain ground truth positions for each video frame, and plot the position errors in the x, y, and z-dimensions. Our method is more accurate than the current state-of-the-art method of [8] and P3P [7]. Our method has significantly less jitter than other methods, making it a feasible options for use in SLAM.

We compare the accuracy of using our algorithm for direct model-based tracking to using the state-of-the-art algorithm of [8] (which also utilizes a gravity vector) as well as the standard P3P algorithm [7]. We use the same 2D-3D matches for each algorithm to ensure a fair comparison. We plot the errors in the x, y, and z axes in Figure 7. Our algorithm is clearly more accurate than either alternative algorithm. Further, our algorithm is noticeably more stable and has less jitter in the pose error. Finally, the alternative algorithms are too slow to be able to localize every video frame in real-time.

The performance of our algorithm for localizing every frame in a video sequence in real-time indicates makes it a viable option for model-based tracking. With a more intelligent matching and tracking scheme (*e.g.*, the method presented in [14]) it is likely that our method could be used to provide localization for every frame in real-time for model-based tracking on a mobile device. In the following section, we further show the feasibility of our algorithm with an augmented reality application.

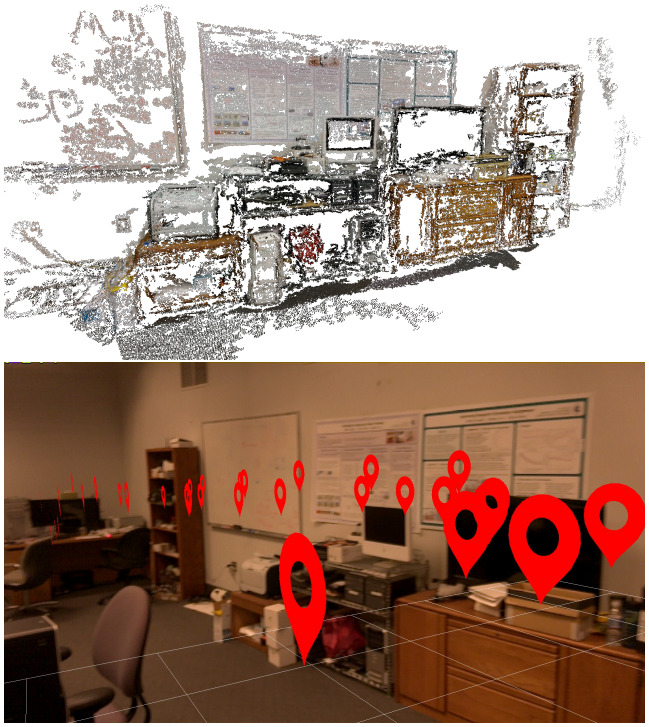

Figure 8: **Top:** We use 3D point cloud (semi-dense rendering is shown) computed offline to serve as a reference for localization. **Bottom:** We created a simple augmented reality application that uses our new absolute pose solver to localize a Google Tango device to the point cloud shown.

4.6 Augmented Reality on Google Tango

Utilizing the vertical or gravity direction for augmented reality applications can improve motion tracking and efficiency [10]. In many cases it is especially convenient since mobile devices often provide an API to directly obtain the direction of gravity. Further, computer vision techniques can easily detect the vertical or gravity direction or use the normal of the detected ground plane as the vertical direction.

We have created an augmented realty application with a Google Tango tablet that uses our new absolute pose method to localize to a reference 3D point cloud. Our algorithm is used to perform an initial localization with respect to a known 3D point cloud. After localization, we utilize Tango's pose tracking API for frame-to-frame tracking and perform localization in a background thread if tracking is lost. This allows the user to freely move the tablet at all times, enabling a seamless user experience while the feature extraction and nearest neighbor search are being performed in the background.

For our augmented reality application, we have created a navigation tool that helps users explore areas that have been identified as "points of interest" in a pre-built reference 3D model. An example screenshot of our application and a semi-dense rendering of an example reference 3D model are shown in Figure 8. For our experiments, we used a 1st generation Google Tango Tablet devel-

opment kit which has a quad-core NVIDIA Tegra K1 processor and 4 GB RAM. In our localization experiments we only used Tango's color camera for pose estimation and the monochrome fisheye camera for frame-to-frame tracking. We do not currently make use of the depth sensor. We tested our application on a sparse point cloud of 2,927 points using the mean SIFT descriptors for each 3D point. Our application runs in real time with an average frame rate of 41.8 frames per second with little to no jitter. For localization in the background thread, we extract SIFT features (keeping the top 500 keypoints with the strongest response) and perform a KD-tree based nearest neighbor search into the 3D point cloud using the FLANN library [17]. The absolute pose is then estimated with a RANSAC scheme using our novel method. The approximate nearest neighbor search and pose estimation takes just over 1 second on average on an unoptimized implementation. A video demonstration of our application is included in the supplemental material. We plan to investigate using binary features (cf. [14]) to enable faster localization in future work.

5 Conclusion

In this paper, we have presented a new algorithm for using knowledge of the vertical direction (or alternatively, the gravity direction) to compute the absolute pose of a camera. Our method is general and works for single or multi-camera systems. Algorithms that utilize inertial sensors that are increasingly important as these sensors become more common on imaging devices. Utilizing motion priors reduces the number of unknown degrees of freedom to solve for and reduces computational complexity. This is important for augmented reality applications where computational resources are often limited.

Our algorithm utilizes an angle-axis formulation which has been proven previously to be very robust to noise [22]. Indeed, our algorithm has slightly better robustness to image and IMU noise compare to the current state-of-the-art. We also show that our algorithm gives better accuracy in a localization experiment with an iPhone 4. Perhaps most importantly, our algorithm only involves solving a quadratic equation and is 3 orders of magnitude faster than the current state-of-the-art. We show that this increased efficiency allows our algorithm to be used for localizing frames in a video sequence directly from a reference 3D model in real time. Finally, we created an augmented reality application using a Google Tango tablet to show the real-world feasibility of our algorithm. We are eager to use our new method on other devices, and have already begun development of our algorithm for a Google Glass device.

6 Acknowledgements

We would like the thank Lucas Buckland for his help with the ART tracker. This work was supported in part by NSF Grant IIS-1219261,NSF Grant IIS-1423676 and NSF Graduate Research Fellowship Grant DGE-1144085.

References

[1] 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*, 24(6):381–395, 1981.

[2] F. Fraundorfer, P. Tanskanen, and M. Pollefeys. A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles. In *Proc. of the European Conference on Computer Vision*, pages 269–282. Springer, 2010.

[3] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Hollerer. Model estimation and selection towards unconstrained real-time tracking and mapping. *Visualization and Computer Graphics, IEEE Transactions on*, 20(6):825–838, 2014.

[4] M. D. Grossberg and S. K. Nayar. A general imaging model and a method for finding its parameters. In *Proc. of IEEE Intn'l. Conf. on Computer Vision*, 2001.

[5] M. Kalantari, A. Hashemi, F. Jung, and J.-P. Guedon. A new solution to the relative orientation problem using only 3 points and the vertical direction. *Journal of Mathematical Imaging and Vision*, 39(3):259–268, 2011.

[6] J.-H. Kim, H. Li, and R. Hartley. Motion estimation for nonoverlapping multicamera rigs: Linear algebraic and l geometric solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1044–1059, 2010.

[7] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2011.

[8] Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-form solutions to minimal absolute pose problems with known vertical direction. In *Proc. of Asian Conference on Computer Vision*, pages 216–229. Springer, 2011.

[9] D. Kurz and S. Benhimane. Gravity-aware handheld augmented reality. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 111–120. IEEE, 2011.

[10] D. Kurz and S. Benhimane. Handheld augmented reality involving gravity measurements. *Computers & Graphics*, 36(7):866–883, 2012.

[11] D. Kurz, P. G. Meier, A. Plopski, and G. Klinker. An Outdoor Ground Truth Evaluation Dataset for Sensor-Aided Visual Handheld Camera Localization. In *Proc. of the Intn'l. Symposium on Mixed and Augmented Reality*, 2013.

[12] G. H. Lee, F. Fraundorfer, M. Pollefeys, P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Muhlfellner, S. Wonneberger, et al. Motion Estimation for Self-Driving Cars With a Generalized Camera. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2013.

[13] B. Li, L. Heng, G. H. Lee, and M. Pollefeys. A 4-point Algorithm for Relative Pose Estimation of a Calibrated Camera with a Known Relative Rotation Angle. In *Proc. of IEEE/RSJ Intn'l. Conf. on Intelligent Robots and Systems*, 2013.

[14] H. Lim, S. N. Sinha, M. F. Cohen, and M. Uyttendaele. Real-time image-based 6-dof localization in large-scale environments. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2012)*, June 2012.

[15] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, 33(1-2):21–39, 2012.

[16] B. Micusik and H. Wildenauer. Minimal Solution for Uncalibrated Absolute Pose Problem with a Known Vanishing Point. In *Proc. of IEEE Intn'l. Conf. on 3DTV*, 2013.

[17] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2, 2009.

[18] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[19] D. Nistér and H. Stewénius. A minimal solution to the generalised 3-point pose problem. *Journal of Mathematical Imaging and Vision*, 27(1):67–79, 2007.

[20] R. Pless. Using many cameras as one. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.

[21] C. Sweeney. *Theia Multiview Geometry Library: Tutorial & Reference*. University of California Santa Barbara.

[22] C. Sweeney, J. Flynn, and M. Turk. Solving for relative pose with a partially known rotation is a quadratic eigenvalue problem. In *Proc. of International Conference on 3D Vision*, 2014.