Single View Pose Estimation of Mobile Devices in Urban Environments

Aaron Hallquist
University of California, Berkeley
aaronh@eecs.berkeley.edu

Avideh Zakhor
University of California, Berkeley
avz@eecs.berkeley.edu
http://www-video.eecs.berkeley.edu/~avz/

Abstract

Pose estimation of mobile devices is useful for a wide variety of applications, including augmented reality and geo-tagging. Even though most of today's cell phones are equipped with sensors such as GPS, accelerometers, and gyros, the pose estimated via these is often inaccurate, particularly in urban environments. In this paper, we describe an image based localization algorithm for estimating the pose of cell phones in urban environments. Our proposed approach solves for a homography transformation matrix between the cell phone image and a matching database image, constrained by knowledge of the change in orientation obtained from the cell phone gyro, and augmented with 3D information from the database to achieve an estimate of pose which improves upon readings from the GPS and compass. We characterize the performance of this approach for a dataset in Oakland, CA and show that for a query set of 92 images, our computed location (yaw) is within 10 meters (degrees) for 92% (96%) of queries as compared to 31% (26%) for the GPS (compass) on the cell phone.

Introduction

Position and orientation, or pose of a mobile device is useful in many online applications such as gaming, entertainment and augmented reality as well as offline applications such as geo-tagging. In order to determine pose, most modern smart-phones are equipped with full sensor packages, including an accelerometer, gyros, GPS, and a compass. Unfortunately, in urban environments, tall buildings block satellite view and introduce multi-path effects, while power lines and trains add significant magnetic interference, making both GPS and compass readings error prone. Furthermore, the barometer, the only source of altitude information, is extremely unreliable. Since most of today's mobile devices are equipped with camera sensors, it is conceivable to use camera imagery to compensate for the above deficiencies in recovering full position and orientation of the mobile device.

In this paper, we propose a sensor fusion approach for determining the pose of a mobile device equipped with typical sensors in a smart-phone. Our approach uses readings from the GPS, accelerometer, and compass to guide a two-step image based localization system. The first step, discussed extensively in [10], uses a city-scale image database to find an image that matches the scene in the query image captured in the cell phone's viewfinder. The approach in [10] achieves 90% image retrieval accuracy for datasets in Berkeley and Oakland, CA, made of tens of thousands of images.

The second step, which is the focus of this paper, uses the matching database image recovered in the first step and its associated pose in the global world coordinates to recover the pose of the query image by taking into account the input from the accelerometer and compass on the mobile device. Our city-scale database contains three dimensional (3D) information, including the full pose of each database image and 3D point clouds corresponding to the depth map of each image. For certain regions such as the city of Oakland, CA, our image database also contains a plane map associated with each image, which fits clusters in the 3D point clouds to large planes such as building faces and the ground. Such databases are becoming increasingly more common as witnessed in Street View by Google and Bing's Streetside by Microsoft.

Our proposed algorithm uses pitch and roll from the cell phone within a vanishing point framework to detect planes corresponding to building faces in both the query and the matched database image. The building faces are then aligned between the query and the matched database image in order to solve for the cell phone's yaw. Using the full orientation and the planes detected, we then estimate a homography matrix using point feature matches between the query and database image. The typical algorithm for such a homography solution is modified to use the 3D information from our database, allowing us to compute the full translation vector including the scale factor, rather than the direction of translation only.

In Section 0, we review related work in this area and the connections to our proposed method. We go over our approach in detail in Section 0. Section 0 shows the results applied to datasets in Berkeley and Oakland CA; we

present conclusions and discuss future work in Section 0.

Related Work

A number of papers deal with pose estimation for urban scenes, with a wide variety of motivations and tools at their disposal. The two major related categories are (a) those solving the single view pose recovery problem, but not necessarily targeting a mobile device, and (b) those targeting mobile platforms, but tracking pose over a video sequence rather than single view pose estimation.

In the former category, the most related work is by Zhang and Kosecka [11], where pose is recovered for a single query image using a similar database matching step and homography estimation. A key difference is that Zhang and Kosecka work with a database containing only the absolute pose of images, while our database contains 3D point cloud associated with imagery as well as the absolute pose of the imagery. In order to compute the scale of translation, [11] must retrieve two images from the database and use the absolute distance between them to triangulate the query position. A similar triangulation approach is taken in [4], estimating the rotation and translation between three views, two of which have known locations. However, they use an essential matrix and also optimize for speed on a mobile device, both of which sacrifice accuracy. Using two database views is potentially unreliable for city-scale databases because it might not be possible to retrieve two matching images in the database. Since we have access to 3D depth information in the database imagery, we opt to use it to compute translation scale without the use of multiple database matches.

Rather than using point features to recover pose, other papers have experimented with matching line features [2] or 2D patterns from a 3D database [6]. The latter is particularly interesting because it takes advantage of the fact that many urban scenes contain repetitive patterns, even though for point features such repetitions are problematic [6].

The majority of work targeting mobile platforms focuses on the tracking problem attempting to create an online algorithm to retrieve images. This is demonstrated in [7], which deals with the accuracy of retrieving database images within X meters of the query image. Others solve the tracking problem, but provide no quantitative performance [8,9]. In contrast, we focus on recovering the pose for a scene from a single camera image captured by a mobile device and evaluate the system performance relative to ground truth. In this paper we do not deal with real time aspects of our approach leaving it as future work.

System Overview

As described in the introduction, our proposed approach estimates the pose of the cell phone by retrieving a matching image from a city-scale database and combining the camera imagery with cell phone sensors.

We assume image retrieval has already been performed and that the input to our algorithm is a correct matching database image. In addition, we assume a specific type of query image taken, one which (a) is mostly upright, i.e. with small roll close to zero, (b) views an urban scene from a reasonable distance and reasonable angles, generally across the street rather than down the street, (c) is taken at roughly human height e.g. 1 to 3 meters above ground, and (d) stays steady for the duration of the capture. Each of these assumptions play their roles in various parts of our algorithm.

The database we work with contains the following information:

- A dense set of black and white image panoramas covering the city, spaced about 10 meters apart.
- The pose for each panorama.
- A depth map for each panorama.
- A set of six "street-facing" rectilinear images and depth maps extracted from each panorama.

The rectilinear images extracted from each panorama are used for image retrieval and pose recovery, and their camera parameters are assumed to be known. Similarly, the camera parameters extracted from the cell phone are assumed to be known. We perform no camera calibration and work in the 3D world rather than with pixel coordinates. That is, for every pixel location (p_x, p_y) in an image, we pre-multiply by the inverse of the camera matrix $\boldsymbol{K}$ to attain its in-world ray w, as in

$$w = \boldsymbol{K}^{-1}[p_x, p_y, 1]^T \quad (1)$$

The skew parameter for $\boldsymbol{K}$ is assumed to be $\gamma = 0$, and its aspect ratio is assumed to be $\alpha = 1$. These conventions and assumptions about image coordinates and camera parameters apply to both the query and database images.

Since city scenes are dominated by building faces, we assume that many point features lie in a plane and compute the cell phone pose by solving for a homography transformation from the database image D to the query image Q. However, the simultaneous solution of position, orientation, and plane normal is ill-conditioned in a noisy environment with respect to the minimization of reprojection error associated with a homography matrix [3]. To address this, we strive to solve for as few of these parameters as possible in a given homography estimation.

There are two main parts to our approach: vanishing point alignment and constrained homography estimation via point feature correspondences. In the system layout in Figure 1, this is visually shown by separating them into a top and bottom flow. The respective goals of the two subsystems are as follows. Using vanishing points we find plane normals in the scenes and align them between the query and database image in order to find the yaw

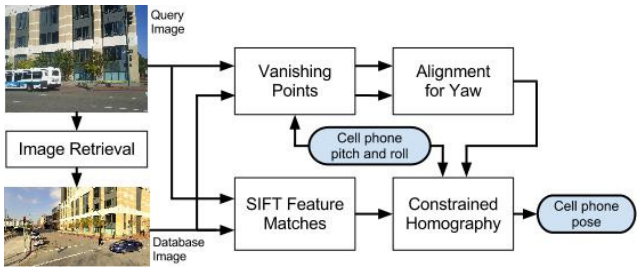

Figure 1: Layout of the pose estimation system.

orientation of the query view. Using feature matches between the query and database image, we solve for the query position by estimating a constrained homography transformation. The extraction of vanishing points and their application are described in Sections 1.2-1.3, while point features and homography estimation are described in Sections 1.4-1.7.

Note that we do not explicitly solve for pitch and roll. Rather, assuming a steady cell phone, they are estimated by using the direction of gravity as measured by a low pass filter of the accelerometer.

1.1. Notation

We refer to the query image taken from the cell phone as Q and to its matched image retrieved from the database as D. Furthermore we distinguish between a "full 3D vector" and a "3D ray": a full 3D vector has a known scale and 3 independent parameters, while a 3D ray is always normalized and has an unknown scale, i.e. 2 independent parameters. In an effort to maintain clarity, the following notation is used:

- Capital letters denote matrices, sets, and objects while lowercase letters denote vectors and scalars.
- A hat is used to denote a normalized vector or a 3D ray, e.g. $\hat{t}$ as the direction of translation.
- The vector symbol to denote a full 3D vector with scale, e.g. $\vec{t}$ to denote 3D translation.
- Unmodified lower-case letters denote a scalar.
- For capital letters, scripts denote a set, bolds denote a matrix, while normal capitals denote an object.

1.2. Vanishing Point Extraction

To obtain the orientation of Q, we use the cell phone accelerometer and magnetometer to extract yaw, pitch, and roll. This extraction is trivial and relies on the direction of gravity from the accelerometer to solve for pitch, roll, and the direction of north from the compass to solve for yaw. Magnetic interference in urban environments typically result in the reported yaw by the mobile device, yaw_{cell}, to be error prone.

To recover a more accurate yaw value for the cell phone and also to compute plane normals for our scene, we turn to vanishing points. Under the assumption that our scenes have planar and upright building faces, we obtain the plane normals of these building faces by computing horizontal vanishing points for both Q and D. This provides a set of planes in each image which we then align to compute a more accurate yaw value for the cell phone.

This section describes our approach to computing the horizontal vanishing points for Q and D. We extract lines using the algorithm proposed in [1]. Since we only need horizontal vanishing points and we assume our images are near upright, we remove lines within 10 degrees of vertical on the image plane. After multiplying by the inverse camera matrix followed by the orientation matrix, we obtain a set of lines $\mathcal{L} = \{L_i\}$ with associated lengths, $\text{length}(L_i)$, in the world frame. Seeds of vanishing points are created, densely sampling all vanishing points perpendicular to gravity. Lines are then assigned to each seed in order to find the most likely vanishing points of the scene's streets and buildings.

In practice, not all vanishing points are perfectly perpendicular to gravity. In our datasets, the angle between gravity and vanishing points is always greater than 85 degrees; therefore, we account for that small difference by applying a guided matching algorithm [3] to the computed set of vanishing points obtained from the seeds mentioned above. This steers each vanishing point toward a nearby location, not necessarily perpendicular to gravity, which simultaneously maximizes the number of contributing vanishing lines and minimizes the vanishing point deviation from all contributing vanishing lines.

The output of guided matching yields a refined set of vanishing points in the world frame and a set of contributing lines $\mathcal{L}_k$. In practice, we output between one and four horizontal vanishing points per image. For any computed vanishing point $\hat{v}$, we compute a confidence value based on the total length of all contributing lines. This is defined as

$$c = \frac{\sum_{i \in \mathcal{L}(\hat{v})} \text{length}(L_i)}{\sum_{j \in \mathcal{L}} \text{length}(L_j)} \quad (2)$$

where $\mathcal{L}$ is the set of all lines detected in the image and $\mathcal{L}(\hat{v})$ are those lines associated with the vanishing point $\hat{v}$. This confidence is used in Section 1.3 for alignment.

1.3. Plane and Yaw Estimation

In this section we describe the way vanishing points computed for Q and D are used to extract planes for the scene and to recover yaw orientation for Q. Using the assumption that all building faces are upright, we compute the plane normals from the vanishing points by taking the cross product of each vanishing point with the gravity vector. We use scene geometry to ensure the dot product between the plane normal and the camera vector[1] is negative, thus specifying the sign of the plane normal.

[1] The camera vector is defined as the world vector representing the direction the viewfinder of the cell phone is facing.

This reduces each plane normal to an associated plane yaw between 0 and 360 degrees, indicating the direction each plane faces.

So far, we have estimated vanishing points, plane yaws, and their associated confidences as computed in Equation 2 for Q and D. We label the plane yaws for Q and D as ϕ^{Q} and ϕ^{D}, respectively, where the superscripts here denote that the plane yaw source is from Q and D, not that they are represented in the query and database coordinate frames. Similarly the confidences are labeled c^{Q} and c^{D}.

Note that the plane yaws and vanishing points for D are considerably more accurate than those for Q since the latter are based on the yaw reported by the cell phone. For a known pitch and roll, and a yaw error of $\Delta\mathrm{yaw} = \mathrm{yaw}_{\mathrm{actual}} - \mathrm{yaw}_{\mathrm{cell}}$, the transformations to be applied to the query vanishing points and plane yaws to correct for this error are given by

$$\bar{\phi}^{Q} = \phi^{Q} + \Delta yaw \tag{3}$$

where the bar marks a corrected plane yaw.

Our objective is to find the optimal alignment of the vanishing points from Q and D in the world frame in order to estimate the yaw error from the cell phone. To proceed, we find the yaw error $\Delta\mathrm{yaw}$, under the constraint $|\Delta\mathrm{yaw}| < 50°$, which maximizes alignment confidence:

$$\mathrm{conf}_{\Delta yaw} = \sum_{(i,j)\in\mathcal{A}(\Delta yaw)} c_i^Q \, c_j^D \tag{4a}$$

$$\mathcal{A}(\Delta yaw) = \{\, (i,j) \mid \bar{\phi}_i^Q - \phi_j^D < 5° \,\} \tag{4b}$$

In Equation 4(b), the set $\mathcal{A}(\Delta yaw)$ represents every pair of plane yaws which, when correcting for yaw error, are aligned to within 5 degrees of each other. In this way, maximizing the confidence in Equation 4(a) computes a yaw error $\Delta\mathrm{yaw}$ which generates the best alignment of planes between Q and D. With this optimal alignment $\mathcal{A}$, we refine the estimate of cell phone yaw error as a weighted average of each aligned plane yaw. That is,

$$\Delta yaw' = \frac{\sum_{(i,j)\in\mathcal{A}} c_i^Q \, c_j^D \left(\phi_j^D - \phi_i^Q\right)}{\sum_{(i,j)\in\mathcal{A}} c_i^Q \, c_j^D} \tag{5}$$

where this represents a weighted average of the yaw errors generated from each aligned pair. Similarly, we estimate the yaws for our scene's planar normals as $\phi_i = \phi_i^Q + \Delta yaw' \;\forall\; i \in \mathcal{A}$, generating one plane for every aligned pair in the optimal set. In practice, we output one to three planes from this alignment process.

1.4. Feature Correspondences

In order to estimate a homography matrix, we need point correspondences between Q and D. To do so, we extract high resolution SIFT features from the two images and perform a nearest neighbor search between the two sets of features to find matches [5]. We apply a series of pre-homography filters to reduce the number of correspondences, and in particular reduce the number of incorrect feature matches, i.e. outliers.

Since we expect both Q and D to be upright, i.e. roll around 0, we can filter out feature correspondences by their gradient orientation, a parameter we have access to from the SIFT decomposition. The first filter we apply requires feature correspondence to have orientations within 60 degrees of each other. We further apply a relative distance filter, or ratio test, to keep only feature correspondences that are unique [5].

Finally, we use the 3D database to remove all database correspondences that lack depth data. This is because all such features are either too near to the database camera or are part of the sky, neither of which is desirable to keep. In practice this removes very few features.

1.5. Constrained Homography

Given yaw, plane normals, and feature matches, we now estimate a homography matrix to solve for translation. To do so, we must take into account our known parameters, utilize 3D information to recover the scale of translation, and account for outliers with robust estimation. In the remainder of the paper, all vectors are assumed to be represented in the coordinate frame for Q, except when the vector has a superscript d, which denotes that the vector is in the coordinate frame for D.

This section outlines homography estimation using the constraints imposed by the known planes and orientations. The homography transformation maps a plane from a starting view D, which in our set up has a known pose, to a new view Q, containing the same plane [3]. The homography transformation is defined as the mapping from any point feature $\hat{\mathrm{x}}^{\mathrm{d}}$ on D to its corresponding feature $\hat{\mathrm{y}}$ on Q using the following relation:

$$\hat{y} \propto \boldsymbol{H}\hat{x}^{d} \quad \text{where} \quad \boldsymbol{H} = \boldsymbol{R}_d^q + r\,\hat{t}\,(\hat{n}^d)^T \tag{6}$$

The symbol $\propto$ represents proportionality and the superscript T denotes a vector transpose. $\mathbf{H}$ is the canonical homography matrix, representing a rotation and translation of $\hat{\mathrm{x}}^{\mathrm{d}}$ to map to the feature location $\hat{\mathrm{y}}$. $\mathrm{R}_{\mathrm{d}}^{\mathrm{q}}$ and $\vec{\mathrm{t}}$ are the rotation and translation parameters that map a 3D point from D to Q, i.e. $\vec{\mathrm{y}} = \mathbf{R}_{\mathrm{d}}^{\mathrm{q}}\vec{\mathrm{x}}^{\mathrm{d}} + \vec{\mathrm{t}}$, where $\vec{\mathrm{x}}^{\mathrm{d}}$ and $\vec{\mathrm{y}}$ represent the true 3D locations, including depth, of the feature points $\hat{\mathrm{x}}^{\mathrm{d}}$ and $\hat{\mathrm{y}}$ respectively. In this way, $\hat{\mathrm{t}}$ is the direction of translation. The parameter $\hat{\mathrm{n}}^{\mathrm{d}}$ is the normal vector of the scene plane, while r is a unitless scale factor defined as the ratio $\mathrm{r} = |\vec{\mathrm{t}}|/\mathrm{p}$, where p is the projected distance from the plane to the center of the camera for D, i.e. $\mathrm{p} = (\hat{\mathrm{n}}^{\mathrm{d}})^{\mathrm{T}}\vec{\mathrm{x}}^{\mathrm{d}}$ for all points $\vec{\mathrm{x}}^{\mathrm{d}}$ on the plane. Note that the homography matrix traditionally requires solving for eight parameters.

Since we know the absolute orientation of D, and have computed the same for Q using vanishing points, we know $\mathrm{R}_{\mathrm{d}}^{\mathrm{q}}$. Therefore we can remove orientation from Equation 6 by mapping $\hat{\mathrm{x}}^{\mathrm{d}}$, $\vec{\mathrm{x}}^{\mathrm{d}}$, and $\hat{\mathrm{n}}^{\mathrm{d}}$ to the query coordinate frame, i.e. $\hat{\mathrm{x}} = \mathbf{R}_{\mathrm{d}}^{\mathrm{q}}\hat{\mathrm{x}}^{\mathrm{d}}$, $\vec{\mathrm{x}} = \mathbf{R}_{\mathrm{d}}^{\mathrm{q}}\vec{\mathrm{x}}^{\mathrm{d}}$, and $\hat{\mathrm{n}} = \mathbf{R}_{\mathrm{d}}^{\mathrm{q}}\hat{\mathrm{n}}^{\mathrm{d}}$. In doing so, we

obtain the simplified homography relation with all variables in the query coordinate frame:

$$\hat{y} \propto \boldsymbol{J}\hat{x} \quad \text{where} \quad \boldsymbol{J} = \boldsymbol{I} + r\,\hat{t}\,\hat{n}^T \tag{7}$$

We now have a simple expression containing a transformed homography matrix with only five unknown parameters: two in each of $\hat{\mathrm{t}}$ and $\hat{\mathrm{n}}$ and one in r. Using the known plane yaws from Section 1.3 and the assumption that all viewed building faces are upright, $\hat{n}$ is known, which eliminates both parameters from the plane normal and reduces the number of unknown parameters to three.

In order to find an optimal solution minimizing error, we define an error metric with Equation 7. We note that Equation 7 contains three proportionality constraints using the same proportionality factor, and so we can divide by one of them to obtain two equality constraints. The two equality constraints can be formulated as a reprojection error metric for a given feature correspondence pair $(\hat{\mathrm{x}}, \hat{\mathrm{y}})$ and parameter solution set $(\mathrm{r}, \hat{\mathrm{t}}, \hat{\mathrm{n}})$

$$e[0] = \hat{y}[0]/\hat{y}[2] - \vec{z}[0]/\vec{z}[2] \tag{8a}$$
$$e[1] = \hat{y}[1]/\hat{y}[2] - \vec{z}[1]/\vec{z}[2] \tag{8b}$$
$$\text{with} \quad \vec{z} \triangleq \hat{x} + r\,\hat{t}\,\hat{n}^T\hat{x} \tag{8c}$$

The first terms $\hat{\mathrm{y}}[0]/\hat{\mathrm{y}}[2]$ and $\hat{\mathrm{y}}[1]/\hat{\mathrm{y}}[2]$ of Equations 8(a) and 8(b) respectively represent the left side of Equation 7, while the second terms $\vec{\mathrm{z}}[0]/\vec{\mathrm{z}}[2]$ and $\vec{\mathrm{z}}[1]/\vec{\mathrm{z}}[2]$ represent the right side. Equation 8 is the canonical homography reprojection error under the constraints we have imposed. Since the error constraint is two-dimensional and we solve for three parameters, we can find a unique solution which minimizes error using only two independent correspondence pairs, rather than the four that would have been required had we not constrained the problem as above. This reduced complexity better conditions the homography estimation and improves runtime for our robust estimation algorithm.

1.6. Scale Recovery

This section describes how we solve for the scale of translation. The error metric described in Section 1.5 does not solve for the absolute distance $|\vec{\mathrm{t}}|$ between the query and database views, but only the direction $\hat{\mathrm{t}}$. To recover the distance, note that the homography solution obtained does solve for $\mathrm{r} = |\vec{\mathrm{t}}|/\mathrm{p}$. Using a database feature's depth $|\vec{\mathrm{x}}|$, we can compute p with only one database feature as

$$p = \hat{n}^T\vec{x} = |\vec{x}|\,\hat{n}^T\hat{x} \tag{9}$$

where $\hat{\mathrm{n}}$ and $\hat{\mathrm{x}}$ above are in the same coordinate system. Once we have the orthogonal distance p from the center of the camera for D to the plane, we also have the translation scale as $|\vec{\mathrm{t}}| = \mathrm{rp}$, resulting in the absolute position of Q.

We incorporate Equation 9 into the error metric of Equation 8 to find an optimal solution by defining an expanded three-dimensional error metric for a given feature pair and depth $(\hat{\mathrm{x}}, \hat{\mathrm{y}}, |\vec{\mathrm{x}}|)$ and expanded solution set $(\mathrm{r}, \mathrm{p}, \hat{\mathrm{t}}, \hat{\mathrm{n}})$

$$e[0] = \hat{y}[0]/\hat{y}[2] - \vec{z}[0]/\vec{z}[2] \tag{10a}$$
$$e[1] = \hat{y}[1]/\hat{y}[2] - \vec{z}[1]/\vec{z}[2] \tag{10b}$$
$$e[2] = \alpha\,(\,1 - |\vec{x}|\,\hat{n}^T\hat{x}\,/\,p\,) \tag{10c}$$

We have deliberately chosen the error metric e[2] in Equation 10(c) not to be $\mathrm{p} - |\vec{\mathrm{x}}|\,\hat{\mathrm{n}}^{\mathrm{T}}\hat{\mathrm{x}}$, because this would result in an error metric with units of meters. Rather, we choose the form in Equation 10(c) to match the unitless reprojection errors e[0] and e[1]. The constant α in e[2] is a factor which determines the relative importance of the plane depth error with respect to lateral reprojection error. Since our 3D database tends to have greater depth error than lateral error, we set $\alpha = 0.2$, which equates 5% depth error to 1% lateral deviation error. The addition of scale to the error metric has introduced one more unknown parameter, namely p, and one more constraint, namely Equation 9. With this change, we still require only two correspondence pairs to find a unique solution minimizing the error function e.

1.7. Robust Estimation

We estimate the homography parameters using RANSAC [3]. Since we have constrained our homography estimation problem, we only need two random correspondences in each iteration to find an initial solution set to test. However, this is offset by a decrease in the fraction of inliers, since we add an additional depth constraint to the error metric in Equation 10(c) and do not optimize orientation for reprojection error. These changes reject some feature correspondences which would be inliers in a standard homography. In practice, the percentage of inliers for our system is between 2% and 20%, and as such, we generally run between 1000 and 100,000 iterations, with the higher number of iterations occurring when a resulting solution appears to be physically infeasible.

For each potential RANSAC solution, we apply three filters which weed out poor solutions and allow us to detect when RANSAC has failed. These three validity checks are:

$$|\mathcal{M}| \geq m \tag{11a}$$
$$|\vec{t}| < 75 \text{ meters} \tag{11b}$$
$$|2 + \Delta vert| < 3 \text{ meters} \tag{11c}$$

where $\mathcal{M}$ is the inlier subset of all features correspondences and Δvert is the computed vertical displacement between the center of the cameras from D to Q. Equation 11(a) requires the number of inliers must be greater than or equal to a fixed parameter m chosen ahead of time; Equation 11(b) requires the distance between Q and D must be less than 75 meters; finally Equation 11(c) requires vertical displacement in the world frame must lie in a bounded region. The third validity check in Equation 11(c) uses an assumption about the constraints of our problem. In practice we have found that query images taken by a human user are consistently around 2 meters

below their respective database images taken by a van. We therefore add this check to reject solutions that do not fit in a bounded region around that vertical displacement. When all solutions fail these validity checks, the homography estimation has failed and we output a null solution.

The validity checks outlined in Equation 11 allow for failure cases, where no solutions are found. We opt to take advantage of this by being stringent on our definition of a valid solution, then relaxing the validity checks and inputs for queries that cannot find a solution under such strict constraints. As such, we run up to three execution cases, one primary and two fail-safe. Two relaxation parameters are varied over these cases: the error threshold ε for defining an inlier and the minimum number of feature inliers m.

In the first execution case, we set $\varepsilon = 0.01$ and $m = 15$. If the result of this execution case returns no valid solution, then the second execution case relaxes the parameters to $\varepsilon = 0.03$ and $m = 10$. If there is no solution after the second case, we relax the third case by letting RANSAC optimize the plane yaw in $\hat{n}$ in addition to r and $\vec{t}$. The relaxation parameters remain the same, i.e. $\varepsilon = 0.03$ and $m = 10$. Note that this increases the complexity to four unknowns, but still requires only two feature correspondences to find an initial solution.

If the final execution case fails to retrieve a valid solution, then we naively set the 3D translation vector $\vec{t} = 0$, implying that Q and D are taken from the same spot. This mirrors the approach taken by [11] when homography solutions fail. In practice, most queries find a valid solution in the first execution case, while only a few fail in all execution cases.

Experimental Results

We have tested our algorithm on city-scale databases of Berkeley and Oakland, CA provided by Earthmine Inc. The database images provided are geo-tagged panoramas taken from a van with a corresponding depth map and, in the case of the Oakland database, a plane map. From the panoramas, we extract 6 rectilinear building-facing views, corresponding to 60, 90, 120, 240, 270, and 300 degrees relative yaw, defining 0 degrees relative yaw as the direction the van is facing and 90 degrees as the street-side view to the right of the van. Table 1 contains more of the specifics of the Berkeley and Oakland databases.

The database images are used as potential matching views to the query dataset, where the matched views are obtained from the image retrieval algorithm presented in [10]. Since the algorithm does not have perfect performance, we eliminate those matched views which cannot be usefully applied to our pose estimation system. Those eliminated include the following cases:

(a) The matched view is of the wrong building.
(b) The focal scene is too far from either camera.
(c) The matched view is of the wrong building face on the correct building.
(d) The matched view has too little in common, i.e. the overlap between both views is less than 5%.
(e) The matched view is at too oblique an angle with respect to the relevant building face; generally < 15 degrees.

Those query images not removed by case (a) represent the "Correct retrieval" query images in Table 1, while those not removed by any case above are considered the "Useful retrievals" in Table 1.

In order to evaluate the performance of our system, we use Google Earth, query imagery, and memory knowledge of the query path to record a ground truth latitude, longitude, and yaw orientation for each query image in both the Berkeley and Oakland query sets. We acknowledge that these ground truth values may be error prone, both because they were estimated by hand and eye and because there may be a discrepancy between Google Earth coordinates and Earthmine coordinates. However, our performance plots indicate that the errors in our ground truth locations and yaw are within 5 meters and 5 degrees respectively. For this reason, we focus on 5 and 10 meter performance values, and we do not attempt to analyze performance within a couple of meters.

1.8. Yaw Performance

Figure 2 shows the performance of the yaw estimation method discussed in Section 1.3, as compared to the compass on the mobile device for the Oakland and Berkeley datasets. In Oakland, we obtain a yaw within 5 degrees of ground truth 76% of the time, compared to 10% for the compass, and within 10 degrees of ground truth 96% of the time, compared to 26% for the compass. Similar gains are found in Berkeley, where 62% of queries are within 5 degrees, compared to 13% for the compass, and 90% of queries are within 10 degrees, compared to 36% for the compass.

One limitation of our approach is the reliance on the compass reading to ensure that orthogonal building faces

Table 1: Dataset properties.

Dataset	Berkeley	Oakland
Database area	~1 km^2	~4 km^2
Db image size	2048x1371	2500x1200
Database size	~12000 images	~29000 images
Plane fitting	no	yes
Query image size	2592x1952	2560x1920
Query size	91 images	112 images
Correct retrievals	83 images	102 images
Useful retrievals	73 images	92 images

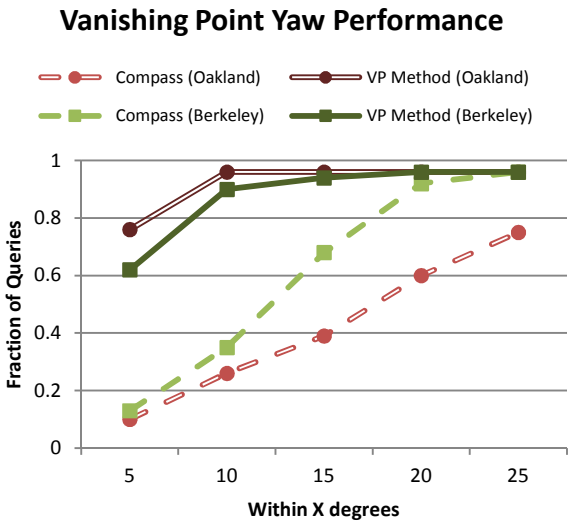

Figure 2: Yaw orientation error in degrees using manual ground truth yaw value from Google Earth for both the Berkeley and Oakland datasets. The plots compare our vanishing point method to compass values from the cell phone.

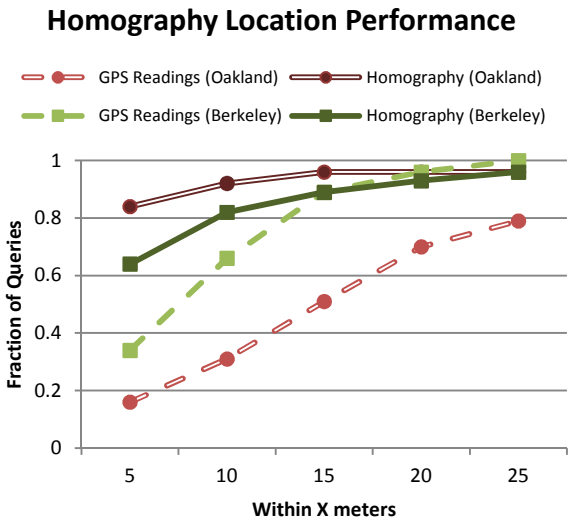

Figure 3: Location performance within X meters of the ground truth location for both the Berkeley and Oakland datasets. The plots compare our homography estimation method to GPS readings from the cell phone.

are not accidentally aligned. Since compass error is generally within 50 degrees and most planes in the city are separated by 90 degrees, there are not too many misalignments. However, not all city planes lie on a rectangular grid. As such, the algorithm struggles when plane normals are separated by less than 50 degrees, as the compass cannot distinguish between multiple distinct plane alignments when they are all within the bounds of compass error. When there are multiple plane alignments, we depend on the plane weights and alignment confidence to choose the alignment, which is susceptible to error.

1.9. Location Performance

The results for location from homography estimation, discussed in Section 1.7, are presented in Figure 3, which plots location performance for the Berkeley and Oakland datasets using both our system and the raw GPS reading from the cell phone. Our algorithm improves location estimates substantially, especially with the Oakland dataset, where it is within 5 meters of ground truth 84% of the time compared to only 16% for GPS, and within 10 meters 92% of the time compared to only 31% of the time for GPS. Berkeley performance does not improve as much over GPS, both because our system performs more poorly and GPS performs better. Using the homography location estimate in Berkeley, 64% of queries are localized within 5 meters, while 82% are localized within 10 meters. This is still better than GPS, which performs at 34% within 5 meters and 66% within 10 meters. We speculate that Berkeley performs more poorly than Oakland because the accuracy of the Berkeley 3D database from Earthmine is lower than that of Oakland, and improvements in the 3D point clouds result in more accurate pose estimation.

Table 2 shows the breakdown of various execution cases for Oakland queries with less than 5 and 10 meter location error. As seen, the majority of queries find a solution in the first execution case, and succeed in being well localized when they do.

Homography fail cases generally can be explained by problems with the scenery. For example, when the database and query images are separated by a large distance or angle, the homography estimate struggles to find inliers. Additionally, feature repetition can cause homography matches to exhibit a displacement that produces an incorrect position estimate. More generally, the algorithm can fail because of scenes that make feature matching difficult, such as extreme occlusion, drastic lighting differences, or changes in scenery in the time between query and database snapshots. Since we work with a homography transformation, the algorithm also struggles in the absence of dominant planes.

Furthermore, since homography estimation leaves yaw orientation fixed, its performance is inherently linked to the performance of our yaw estimate. In our datasets, this

Table 2: Oakland dataset run characteristics, including execution cases and the use of planes from vanishing points.

Execution Case	***1***	***2***	***3***	***t=0***	***Total***
Total number of queries	83	7	1	1	92
Number within 10 meters	79	4	1	1	85
Number within 5 meters	73	3	1	0	80

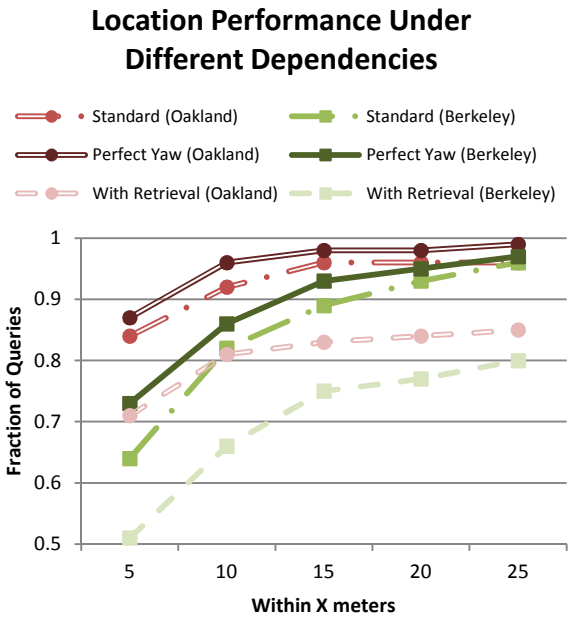

Figure 4: Location performance within X meters of ground truth for both the Berkeley and Oakland datasets. The plots compare how the homography estimation performs when incorporating the results of image retrieval as well as when removing the dependence on a yaw estimate from vanishing points to the standard performance obtained from Figure 3.

accounts for a portion of the incorrect homography estimates. Finally, we have implicitly assumed that our overall system has correctly retrieved a usable database image to match the query. Thus the overall system performance depends on three subsystems: retrieval, vanishing points, and homography estimation.

The following three location performance cases are plotted in Figure 4 for both Oakland and Berkeley: dependency on both image retrieval and yaw recovery, dependency on yaw recovery but assuming a successful retrieval, and assuming both a successful retrieval and a perfect yaw using the ground truth value. This analysis shows we perform extremely well given the correct yaw, localizing within 5 meters 86% of the time and within 10 meters 96% of the time in Oakland. However, if we incorporate image retrieval performance into our system, we only localize within 10 meters 81% of the time in Oakland and 66% of the time in Berkeley. This is identical to GPS in Berkeley but still greatly outperforms GPS in Oakland.

Future Work

For real time application areas such as augmented reality, computational complexity is an important factor. In this paper, we have mostly ignored all real time issues as we deal with single image pose recovery. To improve the homography estimation runtime, it is possible to examine robust estimation schemes which scale better when the fraction of inliers goes down. We can analyze the performance of the system with lower resolution images, which would reduce the number of lines, number of features, and RANSAC runtime.

Acknowledgements

Left blank for anonymous submission.

References

[1] R. Grompone, J. Jakubowicz, J. Morel, G. Randall. LSD: A Fast Line Segment Detector with a False Detection Control. *Pattern Analysis and Machine Intelligence,* 32, 4 (Apr. 2010), 722-732.

[2] W. Guan, L. Wang, J. Mooser, S. You, U. Neumann. Robust Pose Estimation in Untextured Environments for Augmented Reality Applications. In *Mixed and Augmented Reality* (Orlando, Florida, USA, October 19-22, 2011).

[3] R. Hartley, A. Zisserman. *Multiple View Geometry*. Cambridge University Press, 2006.

[4] H. Hile, R. Grzeszczuk, A. Liu, R. Vedantham, J. Kosecka, G. Borriello. Landmark-Based Pedestrian Navigation with Enhanced Spatial Reasoning. In *Proceedings of Pervasive* (Nara, Japan, May 11-14, 2009).

[5] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision,* 60, 2 (Jan. 2004), 91-110.

[6] G. Schindler, P. Krishnamurthy, R. Lublinerman, Y. Liu, F. Dellaert. Detecting and Matching Repeated Patterns for Automatic Geo-tagging in Urban Environments. In *Computer Vision and Pattern Recognition* (Anchorage, AK, June 23-28, 2008).

[7] G. Schroth, R. Huitl, D. Chen. Mobile Visual Location Recognition. *Signal Processing Magazine,* 28, 4 (Jul. 2011), 77-89.

[8] G. Takacs, M. Choubassi, Y. Wu, I. Kozintsev. 3D Mobile Augmented Reality in Urban Scenes. In *International Conference on Multimedia and Expo* (Barcelona, Spain, July 11-15, 2011).

[9] Y. Wu, M. Choubassi, I. Kozintsev. Augmenting 3D Urban Environment Using Mobile Devices. In *Mixed and Augmented Reality* (Basel, Switzerland, October 26-29, 2011).

[10] J. Zhang, A. Hallquist, E. Liang, A. Zakhor. Location-based Image Retrieval for Urban Environments. In *International Conference on Image Processing* (Brussells, Belgium, September 11-14, 2011).

[11] W. Zhang, J. Kosecka. Image Based Localization in Urban Environments. In *3D Data Processing, Visualization, and Transmission* (Chapel Hill, NC, June 14-16, 2006).