Multisensory Embedded Pose Estimation

Eyrun Eyjolfsdottir Matthew Turk
Computer Science Department, UC Santa Barbara
eyrun@cs.ucsb.edu, mturk@cs.ucsb.edu

Abstract

We present a multisensory method for estimating the transformation of a mobile phone between two images taken from its camera. Pose estimation is a necessary step for applications such as 3D reconstruction and panorama construction, but detecting and matching robust features can be computationally expensive. In this paper we propose a method for combining the inertial sensors (accelerometers and gyroscopes) of a mobile phone with its camera to provide a fast and accurate pose estimation.We use the inertial based pose to warp two images into the same perspective frame. We then employ an adaptive FAST feature detector and image patches, normalized with respect to illumination, as feature descriptors. After the warping the images are approximately aligned with each other so the search for matching key-points also becomes faster and in certain cases more reliable. Our results show that by incorporating the inertial sensors we can considerably speed up the process of detecting and matching key-points between two images, which is the most time consuming step of the pose estimation.

1. Introduction and related work

The mobile phone has become an ideal platform for augmented reality applications, with its growing computational power and camera quality as well as the availability of sensors such as accelerometers, gyroscopes, magnetometers and GPS receivers. Many augmented reality applications rely on computer vision algorithms, some of which can be extremely computationally expensive. The processors in the mobile phones are not as powerful as those of PCs, however the benefit of having the additional sensors can be greater than the disadvantage of having less processing power. We aim to demonstrate that in this work.

We focus on the problem of estimating the position and orientation (pose) of a phone at the time images are taken on its camera. This involves finding the transformation of the phone as it moves from its pose corresponding to one image, to its pose corresponding to a second image. It is the core of applications such as image stitching and 3D reconstruction. We show how the combination of accelerometers and gyroscopes, can simplify and speed up the process of finding this transformation.

We use accelerometers and gyroscopes to track the position of the phone as it moves in space. Using such sensors to track an object is not new: inertial navigation systems (INS) were originally designed for rockets and have since been used in vehicles such as aircraft, submarines, ships and automobiles. Now that very small and affordable accelerometers and gyroscopes have become available, the same techniques can be used for mobile devices. These sensors have been used for gesture and handwriting recognition [1], as well as for motion capture [2]. They have also been used in combination with computer vision; Aron- $et\ al$ [3] use a combination of inertial sensors and camera for motion tracking in an AR application. They use vision as their primary method, but when it does not give reliable results they fall back on they inertial based estimation. This provides them with more freedom and more accuracy than using only one technique or the other. Other kinds of fusions have also been explored, such as using the inertial sensors for short time pose prediction [4], or full integration [5]. Bleser and Stricker [6] provide a comparison of the different models with special investigation of the effects of using accelerometers on the tracking performance. Their results show that fully exploiting all information given by the accelerometer measurements has clear benefits: better tracking quality and reduced demand for vision measurements.

The inertial tracker can be used to estimate the transformation of the phone as it is rotated and translated between taking images. The estimate can then be used to warp the images such that they appear to have been taken from the same position. This allows the use of less computationally heavy feature descriptors. A similar idea has been implemented by Pulli- $et\ al$ [7] for their Panorama application. They track the phone's orientation using low resolution images captured in real time, and then use that estimate as input to an algorithm which uses higher resolution images to find a more precise rotation estimation.

To detect features we have implemented an algorithm that uses FAST [8, 9] and is adaptive such that the number of features detected lie in a desired range and are well distributed across the image. Brown- $et\ al$ [10] introduced an adaptive non-maximal suppression algorithm to ensure the same properties and they stress the importance of having those for fast, yet robust keypoint matching. Once we have detected the keypoints we use very low cost descriptors, namely image patches, to match the keypoints between two images.

In our evaluation we compare our method with another which uses computationally heavy, robust features, and our results show that our approach produces very similar results, when successful, but is more than ten times faster.

The paper is organized in the following manner. First we give a brief overview of system components and the tools we used to build the system. In Section 3 we discuss the process of tracking the position and orientation of the phone using the inertial sensors; in Section 4 we describe our keypoint detector and descriptor and discuss how we use the inertial based pose estimation to assist our vision based approach. Finally we describe our experiments and results.

2. System overview

Our system consists of the following main components: (1) an interface thread which streams the camera output to the screen as well as thumbnails of the images taken, (2) a camera thread which processes the raw camera output, (3) a sensor thread which continuously reads data from the sensors and marks the time of each image taken, and finally (4) a computation thread which as soon as two images have been processed by the camera thread, starts estimating the pose. At the end of this pipeline there could be a viewer for 3D models, or for panoramas. We implemented a viewer which lets us visualize the sparse 3D point cloud extracted from the image pairs during the process.

For this project we used a Nokia N900 phone which runs on Maemo, a Linux-based operating system. It has 256 MB RAM and 32 GB storage, an ARM Cortex A8 600 MHz processor as well as a PowerVR SGZ530 graphics card and a 5 MP camera. The phone has built in accelerometers, but in order to make use of gyroscopes as well we have attached an external sensor box to the phone. The sensor box includes three gyroscopes that measure the angular velocity around each axis, and three accelerometers that measure the acceleration along each axis. To access the data from the sensor box, the phone connects to it via Bluetooth.

We implemented the system in C++. We use Qt for the graphical user interface and to handle the multiple threads of our system, OpenGL ES 2 for 3D visualization, and for some of the computer vision algorithms we have used OpenCV. To access and control the camera we used FCam, which is an API that was recently developed by Stanford and Nokia [11].

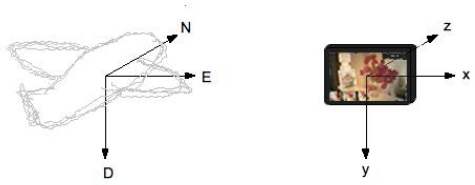

Figure 1. The inertial coordinate frame, N-E-D, where N stands for north, E for east and D for down, and the local coordinate frame of the phone.

3. Inertial based tracking

The inspiration for this project came from the desire to make effective use of the inertial sensors of a mobile phone. With the right combination of inertial sensors we are able to track both the 3D position and orientation of the phone. We will first discuss what can be extracted from the accelerometers alone, and then present the benefit of adding gyroscopes to the equation. We will also present some of the challenges involved and how we handled them.

3.1. Accelerometers

A linear accelerometer measures the acceleration along the axis with which it is aligned. Integrating the acceleration over time gives the velocity, assuming that the initial velocity is known. By double integrating we are able to obtain the position at each time point, given that we also know the initial position. We assume that the sensor is initially located at the origin and that it has zero velocity; we then have the following equations, using Simpson's trapezoid rule for integration:

$$\begin{aligned} v_0 &= 0 \\ p_0 &= 0 \\ v_i &= v_{i-1} + \Delta t(a_i + a_{i-1})/2 \\ p_i &= p_{i-1} + \Delta t(v_i + v_{i-1})/2 \end{aligned} \tag{1}$$

where a_i is the measured acceleration at time step i.

The N900 (and the sensor box) has three accelerometers, one aligned with each of the phone's axes. We can therefore measure the acceleration of the phone in the three dimensions of its local coordinate frame. Figure 1 shows the two coordinate frames to which we will refer. The one on the left is the inertial coordinate system, where N stands for north, E for east and D for down. The one on the right is the phone's local coordinate system. If we assume that the phone's coordinate frame is perfectly aligned with the inertial frame and that it does not rotate with respect to the inertial frame over time, we can use the above equations to

calculate the 3D position of the phone at any given time. Of course it is necessary to first subtract gravity from the acceleration measured with the y-axis accelerometer. In practice it is not this simple because the no-orientation assumption fails. Even if users try not to rotate the phone they inevitably do, and for our purposes we want users to be able to move the phone freely. Gravity then no longer affects only the acceleration measured by the y-axis sensor but also the x-axis sensor (if the phone is rotated around the N-axis) and the z-axis sensor (if the phone is rotated around the E-axis).

Let us consider what else can be obtained from the accelerometers alone. If the phone is not undergoing any linear movement we can use the measured acceleration due to gravity to calculate the roll (orientation around the N-axis) and pitch (orientation around the E-axis) of the phone. If the phone is neither rolled nor pitched, the acceleration vector a equals the gravity vector g. If the phone is oriented 90° clockwise around the N-axis, $a = [\|g\| \ 0 \ 0]^T$. If it is oriented 45° clockwise around the N-axis, $a = [\frac{\|g\|}{\sqrt{2}} \ \frac{\|g\|}{\sqrt{2}} \ 0]^T$. To calculate the roll and pitch in general we have the following equations:

$$\begin{aligned} \phi &= arctan(\frac{-a_x}{a_y}) \\ \theta &= arctan(\frac{a_z}{a_y}) \end{aligned} \tag{2}$$

These equations can be used to estimate the orientation of the phone at each point of time and to subtract gravity from the acceleration vector accordingly. That is done by calculating the roll and pitch each time the phone is not accelerating and interpolating the angles between each two calculated points in time. However, this does not handle yaw (orientation around the D-axis) and most critically this assumes that the angle changes linearly, when in fact the phone can be rotated $x°$ and then back within one motion, so gravity change during that motion is never accounted for. In order to more accurately measure the position and orientation of the phone we have incorporated gyroscopes.

3.2. Gyroscopes

We added an external sensor box to the phone which contains three gyroscopes[1], one measuring the angular velocity around each axis of the phone. The gyroscopes allow us to calculate the 3D orientation of the phone's coordinate frame with respect to the inertial frame. The angular velocity with respect to the inertial frame can be calculated from the local angular velocity, ω, using the equations below. The actual angles are then obtained by integrating the inertial angular velocity and assuming the initial orientation to be zero. With these angles we can rotate the acceleration vector to get a new acceleration vector, a_I, which represents the acceleration of the device in the inertial frame. (We will refer to the acceleration in the local coordinate frame as a_L.) We can then use Eq. 2 to measure the velocity and 3D position of the phone in the inertial frame. The following equations are used to calculate the orientation and to rotate the acceleration vector:

$$\begin{aligned} \theta' &= \omega_y \, cos\phi - \omega_x sin\phi \\ \phi' &= \omega_z + (\omega_y \, sin\phi + \omega_x \, cos\phi) * tan\theta \\ \psi' &= (\omega_y \, sin\phi + \omega_x \, cos\phi) \, / \, cos\theta \end{aligned} \tag{3}$$

$$C_L^I = \begin{bmatrix} cos\psi \, cos\theta & sin\psi \, cos\theta & -sin\theta \\ c_{1,0} & c_{1,1} & cos\theta \, sin\phi \\ c_{2,0} & c_{2,1} & cos\theta \, cos\phi \end{bmatrix} \tag{4}$$

$$\begin{aligned} c_{1,0} &= -sin\psi \, cos\phi + cos\psi \, sin\theta \, sin\phi \\ c_{1,1} &= cos\psi \, cos\phi + sin\psi \, sin\theta \, sin\phi \\ c_{2,0} &= sin\psi \, sin\theta + cos\psi \, sin\theta \, cos\phi \\ c_{2,1} &= -cos\psi \, sin\phi + sin\psi \, sin\theta \, sin\phi \end{aligned}$$

$$a_I = C_L^I a_L - g \tag{5}$$

where θ, ϕ and ψ represent the $pitch$, $roll$ and yaw respectively, and C_L^I is the rotation matrix used to rotate the local acceleration vector to the global acceleration vector.

3.3. Challenges

Adding the gyroscopes is not enough to get a good pose estimation. Both sensors, the gyroscopes and the accelerometers, have small measurement errors, and due to the double integration step of acceleration and single integration of angular velocity these errors accumulate over time. Additionally, the error in orientation will affect the position estimation which is dependent on the orientation. Many types of errors can affect the measurement, for example bias error, bias instability error, white noise and temperature effects [12]. The bias is a constant offset from what the real measured value should be. It can easily be accounted for by averaging the recorded values of the sensors while the device is in static state and then subtracting this error from future measurements. Handling the other errors is more difficult.

To account for the other errors we globally manipulate the data in a post-processing step. We note that for linear movement, the area under the acceleration curve on each side of $a = 0$ should be equal, otherwise there will be residual velocity and the position will drift. After rotating the acceleration vectors at each time step, according to the measured orientation, the trendline of the curve should therefore ideally be $a = 0$. In our experience the slope of it was usually close to zero, but it was still shifted from from $a = 0$.

[1] Several phones already come with gyroscopes, such as iPhone 4, Motorola Droid and Nokia N95.

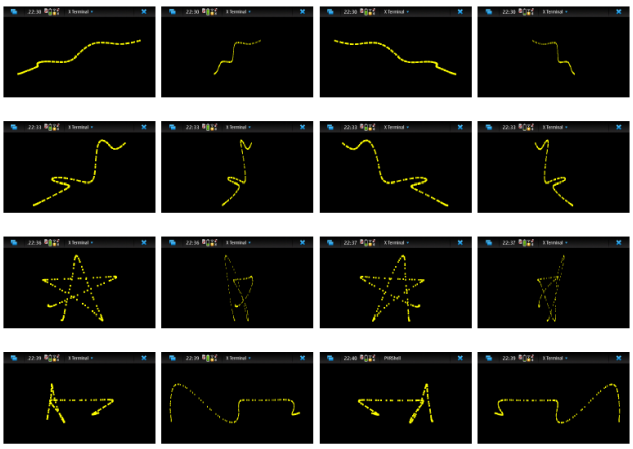

Figure 2. Trajectories calculated from a single recorded IMU sequence using: only acceleration (top row), acceleration and angular velocity (second row), acceleration, angular velocity post processing (third row), acceleration and post processing (bottom row). Each column shows a distinct view of the trajectories as they rotate in 3D space.

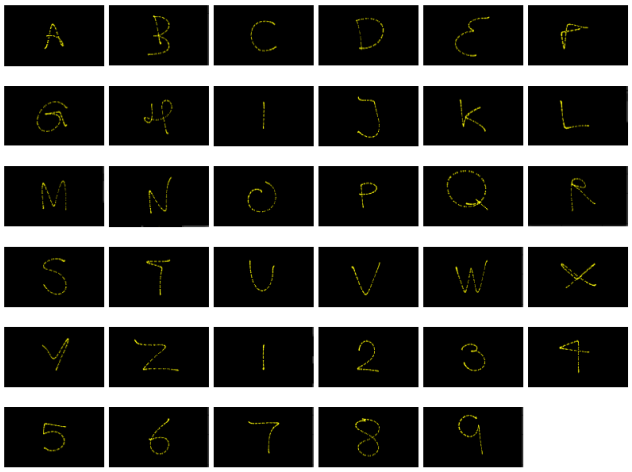

Figure 3. Trajectories generated by writing out the alphabet and numbers in the air while holding the phone by hand. The position of the phone at each point of time is calculated by our inertial tracker. These 35 shapes were recorded one after another in a single try.

We therefore manipulate the acceleration curves by shifting them down (or up) such that their trendline aligns with $a = 0$.

This has a big impact on the final output as can be seen in Figure 2. The figure shows the comparison of four different methods applied to the same data to estimate the trajectory of the movement from which the data was recorded. In this case the movement was in the shape of a star in the D-E plane. The trajectory shown in the top row was obtained by using only the acceleration, the one in the second row included the angular velocity and rotating the acceleration vector, and the third row shows the result when the post-processing step has been applied to the data. Since this has such a great effect on the final output we also tried applying that step to the acceleration data alone, yielding the trajectory shown in row four. This shows that even though this movement was recorded while trying not to rotate the phone, the unintended rotation has a significant impact on the estimate.

Figure 3 shows the results of our tracker after tracing the alphabet and digits in the air while holding the phone. The trajectories were recorded all in one take.

4. Vision based pose estimation

We can now use this inertial tracker to estimate the transformation of the camera as it moves between two different views. We label the time when each image is taken and use the sequence of IMUs (inertial measurement units) recorded in that time interval to calculate the pose at the final point. The assumption that the phone is steady at the beginning and end of the sequence holds in this case since the user usually tries to hold the camera steady in order not to produce a blurry image.

For a more accurate estimation we use a vision based algorithm, with this initial estimate from the inertial sensors as input to speed up the process. We use 860x652 images.

4.1. Warping

When matching keypoints between two images that were taken from different angles, the keypoints' descriptors need to be robust to variations in scale, rotation, translation and illumination. Calculating descriptors with these properties can be very time consuming.

However, since we already have an estimate of the rotation, R, and the translation, t, between the two images we can warp one of them into the perspective frame of the other, such that they approximately align with each other. Then the feature descriptors need not be as robust to some of the factors mentioned above.

The perspective projection is given by:

$$P = KHK^{-1} \tag{6}$$

where K is the camera matrix containing the intrinsic parameters of the camera, and H is the homography for a plane with normal n (in the camera's frame) and located at a distance d from the camera's principal point, given by:

$$H = R - \frac{tn^T}{d} \tag{7}$$

In the case where there is no translation between the images, it can be easily seen that P will be the same for each

point in the image, since the homography will be the same for each plane. However if there is translation, the perspective projection is different for each plane. In our approach we choose a plane which is parallel to the camera's image plane and at a distance d. For our evaluation we used the assumption that the distance is 3 meters. A better (but more intrusive) way would be to have the user input an estimate of the distance to the points of interest. If the motion of the camera is primarily rotation, this distance estimation has little as no effect, but if the phone was translated, an underestimate of the distance results in too much translation of the image, and vise versa.

By warping the corners of each image onto the other image we obtain bounding boxes that define regions of interest which contain features common to both images. This means that fewer features need to be calculated per image.

4.2. Keypoint detection

After warping the images we search for keypoints in each image. In order to quickly detect the interest points we use an adaptive algorithm which has the FAST feature point detector at its core. We found that using FAST alone was indeed very fast, but the number of points detected depends greatly on the parameter b (the threshold used to detect whether the intensity difference between two points is great enough to be considered a corner). In our experience, images with high contrast produced a high number of feature points, but ones with lower contrast did not, which means that lower contrast images need a lower value of b. The same applies within a single image: if there is a very bright area in one part of the image (for example a window seen form the inside of a house), most of the keypoints detected are concentrated on that area.

In order to ensure that all images, no matter what their contrast, will get an adequate number of feature points (without the high contrast images getting too many), as well as making sure that the points are well distributed within the image, we have implemented the following algorithm. We split each image into overlapping patches. For each patch we calculate the standard deviation of its pixel intensity values, and use that value as an initial input to FAST. We then iteratively adjust that value and call FAST again until the number of feature points is with in our desired range, or until a maximum number of iterations has been reached. Figure 4 shows the result of using our adaptive algorithm versus using FAST alone. It can be seen that using FAST with $b = 30$ results in too many keypoints in the left image and too few in the right image, while our method detects an adequate number of keypoints in both images.

This is of course more time consuming than the pure FAST algorithm, since the overlapping areas are examined more than once, and each patch may be examined more than once, but the performance of the pose estimation algorithm is greatly improved in terms of accuracy.

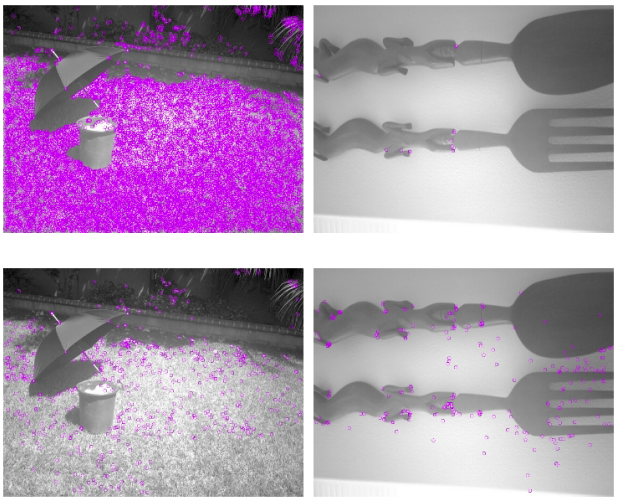

Figure 4. Keypoints in the upper two images were detected by the FAST feature detector, with $b = 30$. The ones in the lower two used our adaptive method.

4.3. Keypoint descriptors

Once the images have been warped to represent the same perspective view, the descriptors used to match keypoints do not need to be invariant to rotation and scale (given that our inertial based estimation is reasonable). We can therefore use descriptors as simple as image patches. We have chosen to use an image patch of size 17x17, which we normalize with respect to intensity values of the pixels such that the highest value of each patch is the maximum intensity value. This way the descriptor becomes more robust to changes in illumination. We also store the average intensity value of each normalized patch.

4.4. Keypoint matching

To match the keypoints between images we use a nearest neighbor search. For each keypoint in the first image we loop though each keypoint in the second image and select the point with the least squared sum of difference in image patch intensity values, to be the match.

We use two methods for pruning the search. First, we compare the average intensity values of the two descriptors, and if the difference is greater than 20 (on a 0-255 scale), we discard that candidate as a match. Second, we check whether the location of the candidate keypoint is close enough to the warped position of the keypoint under examination; if not then we continue. Not only does this speed up the algorithm but it also helps eliminate outliers, which eventually makes finding the fundamental matrix faster and more robust.

Figure 5. Results of warping the second image onto the first image, based on transformation estimates of our method. Shown are four of the image pairs between which the camera was primarily rotated. The black borders represent the seams between the two images.

4.5. Pose estimation

Once we have all the candidates for keypoint correspondences we use the eight point algorithm [13] and RANSAC to find the fundamental matrix which has the highest number of inliers. Candidates that do not support the chosen matrix are discarded. We then calculate the essential matrix and use singular value decomposition to find four possible rotation matrices and two possible translation vectors [14]. The $[R|t]$ combination is selected that deems the 3D points of the inliers correspondences to be at a positive depth as seen from both cameras, and with the lowest reprojection error.

5. Experimental results

In order to evaluate how effective using the inertial sensors is in improving the pose estimation process, we compare the four following approaches:

i) Our proposed method
ii) Our proposed method without inertial assistance
iii) Hessian detector + SURF descriptor
iv) Hessian detector + SURF descriptor with inertial based warping.

The Hessian detector finds keypoints at different scales and the SURF features are robust to scale, orientation, affine distortion and illumination changes [15]. Therefore this combination is often used in applications that deal with images that are temporally far apart. For our evaluation we used OpenCV's implementation, with the default values of 3 octaves and 4 layers per octave.

Our experiments consist of 29 image pairs and the IMU sequences associated with each pair. The actual transformations of the experiments include rotation about each axis, translation along each axis, and a combination of the two. The images are of both indoors and outdoors scenes.

Figure 6 shows a breakdown of the time it takes each approach to obtain the pose estimation. It shows that for robust features, detecting the keypoints and calculating the descriptors is by far the most time consuming part. By first

Time comparison

	i)	ii)	iii)	iv)
Image warping	1052	0	0	1068
Feature detection and description	2877	3089	47478	33358
Feature point matching	884	2132	3736	1956
Finding transformation	1727	3073	2500	2694
Total	**6541**	**8294**	**53714**	**39075**

Figure 6. Average time (in msec) that it took each method to estimate the pose.

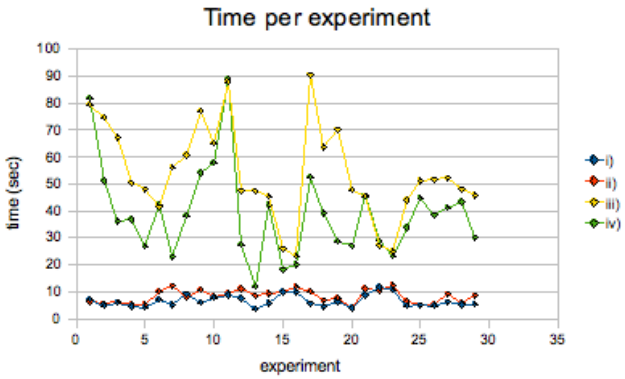

Figure 7. Total time that it took each method to estimate the pose for each experiment. The yellow graph represents the SURF based approach, and the green one the assisted SURF based approach. The two lower lines represent our method, with and with out warping.

warping the images according to the inertial based pose estimation, that process is sped up by almost 30%. This is because with knowing approximately how the images overlap we can limit the image area in which we search for features. Figure 7 shows the average time per experiment for each approach. The top two graphs represent the assisted and unassisted SURF based approaches and the bottom two the patch based approaches. In the cases where the time for iii) and iv) is the same, the transformation between images was either small or it was a rotation about the camera's z-axis, in which case the bounding box we

	Our approach							SURF based approach							Inertial sensors						
	Rotation				Translation			Rotation				Translation			Rotation				Translation		
	x	y	z	angle	x	y	z	x	y	z	angle	x	y	z	x	y	z	angle	x	y	z
1								0.1	**1.0**	-0.2	**4**	0.04	0.02	0.49	**-0.9**	-0.3	0.1	**9**	0.04	0.01	**0.49**
3	0.0	**1.0**	0.0	**26**	-0.03	0.01	-0.16	0.0	**1.0**	0.0	**26**	-0.07	0.03	-0.14	0.0	**1.0**	0.0	**27**	**0.16**	0.00	0.03
4	-0.1	**1.0**	0.0	**18**	0.00	0.00	0.03	-0.1	**1.0**	0.0	**19**	-0.01	0.01	0.03	0.1	**1.0**	0.1	**18**	0.03	0.00	0.01
5	**-1.0**	0.0	0.0	**22**	-0.03	0.02	-0.04	**-1.0**	0.0	0.0	**22**	0.00	-0.02	-0.05	**-1.0**	-0.1	0.1	**22**	0.02	0.00	-0.05
6	0.0	0.1	**1.0**	**53**	0.08	-0.11	0.49	0.0	0.1	**1.0**	**53**	0.41	-0.26	-0.15	0.0	0.0	**1.0**	**52**	0.02	**0.40**	**0.31**
7	0.0	**1.0**	0.0	**26**	-0.05	-0.01	0.04	0.1	**1.0**	0.0	**25**	0.03	-0.01	-0.06	0.1	**1.0**	0.0	**24**	0.06	-0.02	0.01
10	**-1.0**	0.2	0.1	**17**	-0.01	-0.30	0.03	**-1.0**	0.1	0.1	**18**	-0.03	-0.29	-0.10	**-1.0**	0.0	0.1	**15**	0.03	0.30	-0.04
11	**-1.0**	0.1	0.2	**6**	0.04	-0.08	0.53	**-0.9**	0.1	0.3	**7**	0.04	-0.09	0.53	**-0.9**	0.0	0.4	**6**	**0.15**	-0.08	0.51
12	**1.0**	0.0	0.0	**22**	0.00	-0.08	-0.01	**1.0**	0.0	0.0	**22**	0.00	-0.08	0.01	**1.0**	0.0	0.0	**23**	0.00	-0.09	0.00
13	0.0	**1.0**	0.0	**31**	-0.03	-0.01	0.10	-0.1	**1.0**	0.1	**31**	-0.10	-0.01	-0.02	0.0	**1.0**	-0.1	**31**	-0.09	0.00	-0.05
14	0.0	0.1	**1.0**	**67**	0.02	-0.03	0.00	0.0	0.1	**1.0**	**67**	0.02	0.00	-0.03	0.1	0.0	**1.0**	**65**	0.03	0.01	0.00
15	0.1	**1.0**	0.0	**18**	-0.08	0.01	0.05	0.1	**1.0**	0.0	**19**	0.00	0.00	0.09	0.1	**1.0**	0.0	**17**	0.08	0.01	-0.05
16	-0.2	**-0.9**	-0.3	**15**	-0.04	-0.01	-0.05	-0.1	**-1.0**	0.0	**19**	0.01	-0.02	-0.06	0.0	**-1.0**	0.1	**18**	-0.07	0.01	0.00
17	-0.1	**-1.0**	-0.2	**21**	0.12	0.01	-0.07	0.0	**-1.0**	-0.2	**22**	0.13	0.01	-0.03	0.1	**-1.0**	-0.2	**18**	**0.13**	0.00	0.01
18								0.0	**-1.0**	-0.1	**27**	0.13	0.00	0.08	-0.1	**-1.0**	-0.1	**26**	**0.15**	-0.02	0.00
19	0.0	**1.0**	0.0	**25**	-0.42	0.13	-0.74	0.0	**1.0**	0.0	**27**	-0.68	-0.09	-0.53	0.0	**1.0**	0.0	**24**	**0.82**	-0.06	**0.26**
20	**1.0**	0.0	0.0	**25**	0.20	0.05	0.58	**1.0**	0.0	0.0	**29**	-0.04	-0.56	-0.24	**1.0**	0.1	0.0	**24**	0.00	0.02	**-0.61**
21								-0.2	**0.9**	-0.5	**2**	0.02	0.08	-1.00	-0.5	**0.8**	0.3	**2**	-0.02	0.01	**-0.32**
22	-0.2	**-1.0**	-0.1	**11**	0.00	0.00	0.02	-0.1	**-1.0**	-0.1	**10**	0.00	0.00	-0.02	0.0	**-1.0**	0.0	**11**	-0.01	0.00	-0.02
24	**1.0**	0.0	0.0	**18**	0.00	-0.01	-0.11	**1.0**	0.0	0.1	**17**	0.00	0.00	0.11	**1.0**	0.0	0.0	**17**	-0.07	-0.09	0.00
25	**1.0**	-0.1	-0.2	**14**	0.01	0.14	0.02	**1.0**	-0.1	-0.2	**12**	0.02	0.14	0.02	**1.0**	-0.2	-0.3	**11**	**-0.10**	0.06	0.07
26	0.0	**1.0**	0.0	**19**	-0.04	0.01	-0.07	0.0	**1.0**	0.0	**19**	0.00	0.00	0.08	0.1	**1.0**	0.0	**18**	0.07	0.03	0.00
27	0.2	**1.0**	-0.2	**19**	0.14	0.00	0.01	0.1	**1.0**	-0.2	**20**	0.12	-0.01	0.08	0.0	**1.0**	0.1	**19**	**0.12**	-0.06	-0.04
28	0.1	**1.0**	0.0	**16**	0.10	0.00	-0.01	0.0	**1.0**	-0.1	**19**	-0.09	-0.01	0.02	0.1	**1.0**	0.1	**16**	0.08	-0.02	-0.06
29	**1.0**	-0.1	-0.1	**12**	-0.01	0.59	-0.06	**1.0**	0.2	-0.1	**10**	0.00	0.46	-0.38	**1.0**	0.2	0.0	**9**	0.09	**0.57**	-0.14

Figure 8. Transformation between cameras estimated by our algorithm, the SURF based algorithm and the inertial sensors.

used to mark the regions of interest still covers the entire images. In order to speed the process up considerably and reliably, simpler features need to be used. As can be seen in Figure 6, the patch based feature extraction is more than ten times faster than the SURF based one.

However, not all methods were successful for each experiment. We classify an experiment as unsuccessful if it fails to find a fundamental matrix or if the point matches deemed as inliers by the fundamental matrix are obvious outliers. In four cases, neither the patch based nor the SURF based approaches were successful. In three additional cases, which involved a big translation between the two images, our approach was unsuccessful. This is due to the scale dependency of our feature descriptor. As we mentioned before, we use the assumption that the plane on which the points of interest in the scene lie is parallel to the image plane and that the distance from the phone to that plane is three meters. If the distance assumption is bad the warped images will have different scales. And even if the assumptions are good, objects not lying on that plane will have different scales in the warped image.

For five additional cases, our method without the inertial assistance failed. These cases include image pairs between which the transformation is rotation about he camera's z-axis, translation along the z-axis, and images where there are few distinct features. The benefit of having an initial pose estimation in cases like these is that by warping the images, the rotation and scale is, ideally, closer to being the same in both images. By pruning the search for keypoint matches we also decrease the chance of outliers.

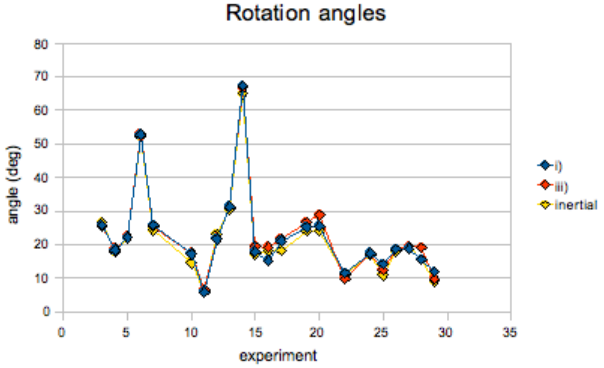

Figure 9. The rotation angle of our method (blue) compared with the SURF based method (red) and the inertial sensor based estimate (yellow).

Next we compare the pose estimation of our method, with that of the SURF based method and the inertial based estimation. Figure 8 shows the output for the three methods on all the experiments, excluding the four that were unsuccessful by all methods. The left three columns contain unit vectors which represent the rotation axes, the next column represents the rotation angles, and the final three the translations. Since the fundamental matrix can only be determined up to scale, the translation vectors are normalized. In order to get a better comparison with the inertial based estimation, we have scaled the translation vectors of the vision

based methods to have the same scale as the inertial based one.

In Figure 9 we have plotted the rotation angles for the 22 experiments that were successful by both methods. We do not have a ground truth to compare with, but these graphs are obtained by two completely different means (purely visual based and purely inertial based) as well as a combination of the two, and the fact that they are so similar suggests that these are close to the truth. It can be seen that our method and the SURF based one agree on the angle in 14 cases, in the other 8 they are still close. Those eight experiments contain images with very few distinct features (such as close-ups of silver spoons on a table), and image-pairs for which translation was a big contributing factor.

Using the results from our method, we again warped image 2 onto image 1. For the experiments where the transformation primarily consisted of rotation, the images aligned well. Figure 5 shows results for four of our experiments.

6. Conclusion

We have introduced a multisensor method for estimating the transformation of a camera between two images taken. We have shown that we can use the inertial sensors of the phone to track its position as it moves between the viewing angles of the two images, and use the pose estimation based on the tracking to enhance the process of finding a visual based pose estimation. We showed that by warping the images based on that estimation we are able to use simpler, and hence less time consuming, feature descriptors.

For transformations that mainly consist of rotation, our method was able to find a pose estimation as good as that of an algorithm which uses features robust to scale and rotation, in less than tenth of the time. However if there is a large transformation, and our estimate of linear movement is not good, or our assumption about the distance to the scene does not hold, the descriptor's lack of invariance to scale can lead to bad results.

It can be concluded that for applications which assume that the camera is only rotated, such as image stitching, using the inertial sensors can be extremely beneficial. For applications such as 3D reconstruction, feature descriptors invariant to scale are necessary, because even when the images are warped based on an accurate pose estimation, objects in the images will have different scales, due to parallax. The transformation from the sensors, however, can still be used to determine epipolar lines, which can be used to limit the search for matching features.

In our approach we have used the inertial based pose estimation mainly to assist the process of matching feature points. In future work, we would also like to use it as an initial guess in the search for the actual pose.

7. Aknowledgements

We would like to thank Nokia Resesarch Center for providing us with a mobile phone and an external sensor box for our research.

References

[1] S.-J. Cho et al. Magic wand: A hand-drawn gesture input device in 3-d space with inertial sensors. *9th Int'l Workshop on Frontiers in Handwriting Recognition*, 2004.

[2] D. Vlasic et al. Practical motion capture in everyday surroundings. *ACM Transactions on Graphics, Volume 26, No. 3, Article 35*, 2007.

[3] M. Aron et al. Use of inertial sensors to support video tracking. *Computer Animation and Virtual Worlds, Volume 18, Issue 1, pp. 57–68*, 2007.

[4] G. Klein and T. Drummond. Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing, 22(10), pp. 769–776*, 2004.

[5] J.D. Hol et al. Robust real-time tracking by fusing measurements from inertial and vision sensors. *Journal of Real-Time Image Processing, Volume 2, Numbers 2-3*, 2007.

[6] G. Bleser and D. Stricker. Advanced tracking through efficient image processing and visual-inertial sensor fusion. *Computers and Graphics, Volume 33, Issue 1*, 2009.

[7] K. Pulli, M. Tico, and Yingen Xiong. Mobile panoramic imaging system. *6th IEEE Workshop on Embedded Computer Vision*, 2010.

[8] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. *Intl. Conf. on Computer Vision, Volume 2, pp. 1508–1515*, 2005.

[9] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *Euro. Conf. on Computer Vision*, 2006.

[10] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. *Conf. on Comp. Vision and Patt. Rec., pp. 510–517*, 2005.

[11] A. Adams et al. The Frankencamera: An experimental platform for computational photography. *SIGGRAPH*, 2010.

[12] O.J. Woodman. An introduction to inertial navigation. *Tech. Rep. UCAMCL-TR-696, University of Cambridge*, 2007.

[13] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Trans. Pattern Anal. Machine Intell., Volume 19, pp. 580–593*, 1997.

[14] W. Wang and H.T. Tsui. An SVD decomposition of essential matrix with eight solutions for the relative positions of two perspective cameras. *ICPR vol. 1, pp. 13–62, 15th International Conference on Pattern Recognition*, 2000.

[15] H. Bay *et al*. SURF: Speeded up robust features. *Computer Vision and Image Understanding, Volume 110, No. 3, pp. 346–359*, 2008.