Tracking 2D pose of mobile phones in real-time

Yashasvi Sriram Patkuri
University of Minnesota - Twin Cities
Minneapolis, MN, USA
patku001@umn.edu

Saurabh Mylavaram
University of Minnesota - Twin Cities
Minneapolis, MN, USA
mylav008@umn.edu

Abstract

Mobile phones are ubiquitous now-a-days, packed with a variety of sensors and desktop level computing power, which opens up a lot of new possibilities in human-computer interaction. In this paper, we try to achieve robust real-time 2D pose tracking of mobile phones. This allows mobile phones to be used as a new kind of input tool, for example for manipulating CAD objects in a more intuitive way or even as an external mouse to another computer.

1. Introduction

Mobile phones are ubiquitous nowadays. They are packed with a variety of sensors (viz. camera, IMU, Gyroscope etc...) and are recently competing with desktop hardware in computing performance. This combination of computing power and sensing technologies opens up a lot of ventures in Human-Computer Interaction. In this paper, we propose using mobile phones as a new kind of input tool. Specifically, we propose ways to use video feed from the camera to track its 2D pose over time. This has interesting applications viz. using a mobile phone as an external mouse, as a laser pointer in presentations, for manipulating objects in CAD software etc...

Our goal in this paper is to track the change in 2D pose of the mobile device over time robustly in real-time. Our primary goal is not to perform mapping of the environment nor to calculate the absolute position of mobile phone with respect to any world-frame of reference.

2. Related work

The problem of pose tracking is a well-known and widely studied research problem. [14] provides a good overview of on image-based camera localization techniques including PnP problems, Simultaneous localization and mapping (SLAM), Structure from Motion (SFM). Filter based SLAM method was first proposed in [4]. [13] showed that keyframe-based SLAM can give more accurate results than filter-based SLAM. [7] uses multiple threaded keyframe-based feature SLAM called Parallel tracking and mapping (PTAM) on mobile phones to achieve real-time localization and mapping. This was implemented in an iPhone 3G where the hardware was much less powerful than current mobile phones. [9] used inertial unit and a rolling-shutter camera to track motion in real-time in mobile devices. [11] proposed an optimization-based visual-inertial camera localization for mobile devices. [5] compares various 2D SLAM algorithms using different metrics. [7], [9], [11] motivate the idea of using vision techniques on mobile hardware. [6] provides a simple inter-frame rotation estimator under rapid camera movement and keyframe-based re-localization method.

3. Assumptions

1. The device is constrained to have only translation and rotation in a 2D space (ex. on a computer desk).
2. We assume a static planar scene at a far enough distance with a decent number of good features to track over time.

These assumptions are made primarily by keeping the use case of mobile phone as an external mouse in mind.

4. Baseline method

In the baseline we additionally assume that the device is constrained not to have rotations. Let F_i denote ith frame in the video feed. Let FD, DE, DM represent feature detector, descriptor extractor and descriptor matcher. Common examples of FD, DE are SIFT[10], SURF [3], KAZE[1], ORB[12], and BRISK[8] algorithms. Common examples of DM are FLANNBASED, BRUTEFORCE, BRUTEFORCE_HAMMING algorithms.

Let FD(F_i) represent key-points in ith frame, len(FD(F_i)) represent number of key-points in ith frame, DE(F_i) represent numeric descriptors of key-points found in ith frame, DM(F_i, F_j, k) be a (len(FD(F_i)) x k) matrix representing corresponding distances b/w k nearest

neighbour feature points in F_j for each feature point in F_i in ascending order of distance (i.e. nearest neighbour first). Then we use algorithm 1 to track the displacement of the mobile phone.

Algorithm 1: Baseline algorithm

Init Th_{anchor}, $Th_{goodframe}$, Th_{NN};
F_{anchor} = first F_i s.t. len(FD(F_i)) $> Th_{anchor}$;
forall *subsequent* F_j **do**
  matches = DM(F_{anchor}, F_j, 2);
  gm_disps = [];
  foreach *match in matches* **do**
    NN_ratio = match[0] / match[1];
    if $NN\_ratio < Th_{NN}$ **then**
      gm_disps.append(match);
    end
  end
  if *len*(gm_disps) $> Th_{goodframe}$ **then**
    emit $disp_{jx}$ = -median(gm_disps_x);
    emit $disp_{jy}$ = -median(gm_disps_y);
  end
end

Essentially we set an anchor frame that has more than a threshold number of key-points. Then for every subsequent frame we detect key-points and match with the anchor frame. This way every key-point in the anchor frame has a correspondence with a key-point in jth frame. We filter out bad matches using a nearest neighbour ratio test. If the number of good matches exceed a certain number we calculate displacement of each key-point from anchor frame to jth frame. The magnitude of total displacement of camera is taken as the median of displacements of all key-points. Th_{anchor}, $Th_{goodframe}$, Th_{NN} are tuning parameters here.

Specifically, we used the video feed from the front camera of a One Plus 7 mobile phone, AKAZE feature detector and descriptor extractor, BRUTEFORCE_HAMMING descriptor matcher, Th_{anchor} = 20, $Th_{goodframe}$ = 10, Th_{NN} = 0.7 in this implementation.

The tracking was real-time with almost no lag. An evaluation of the implementation is conducted. Figure 1 illustrates a straight line movement. Figure 2 illustrates the result of tracking when mobile is moved in a square fashion. Figure 3 illustrates circular movement. Figure 4 illustrates random closed loop movement.

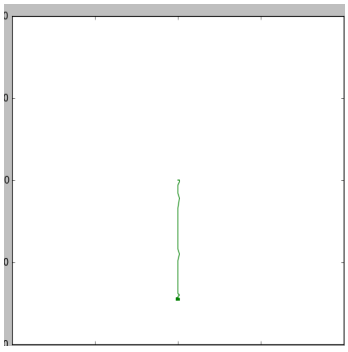

Figure 1. Straight line

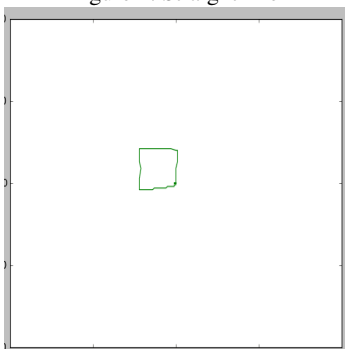

Figure 2. Square movement

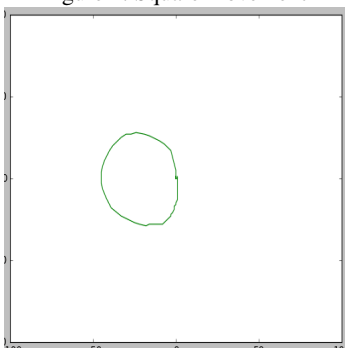

Figure 3. Circular movement

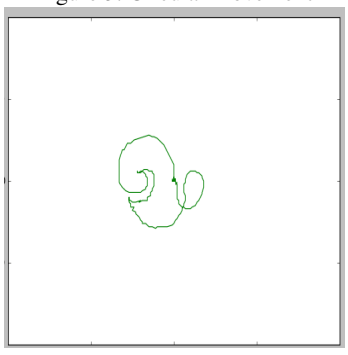

Figure 4. Random closed loop movement

5. Proposed method

While the preliminary work does a decent job, it stills does not handle rotations of the device. The proposed method tries to address this. By the assumptions stated in section 3 the transformation between the key-points in the first frame and key-points in the Nth frame can be approximated as a euclidean transform. By estimating this transform we can decompose it into the translation and rotations

transforms. As the scene the camera sees is assumed to be static the rotation and translation correspond to the camera movement albeit in an opposite sense. [2] describes a simple method for this that can be executed on a mobile hardware in real-time. If we consider A as the set of key-points in first frame and B in the Nth frame we have the following relation between them, where R is the rotation transform and t is the translation transform.

$$R * A + t = B$$

The proposed algorithm is as described in algorithm 2 which builds on algorithm 1.

Algorithm 2: Proposed algorithm

Init Th_{anchor}, $Th_{goodframe}$, Th_{NN};
F_{anchor} = first F_i s.t. len(FD(F_i)) $> Th_{anchor}$;
forall *subsequent* F_j **do**
  matches = DM(F_{anchor}, F_j, 2);
  A = [];
  B = [];
  foreach *match in matches* **do**
    NN_ratio = match[0] / match[1];
    if $NN\_ratio < Th_{NN}$ **then**
      A.append(match.firstframe);
      B.append(match.Nthframe);
    end
  end
  if *len(B)* $> Th_{goodframe}$ **then**
    $A_{recentered} = A - centroid(A)$;
    $B_{recentered} = B - centroid(B)$;
    /* H is a 2x2 covariance matrix */
    $H = A_{recentered} * B_{recentered}$;
    /* SVD = singular value decomposition */
    $U, S, V = SVD(H)$;
    $R = U^T * V$;
    $t = B - R * A$;
    /* Correcting for device movement */
    $R = R^T$;
    $t = t^T$;
    emit R;
    emit t;
  end
end

6. Results

We compare the results of baseline method and proposed method here. First we test for the translation part and then rotation part. Finally we provide paths traced out by both methods in different scenarios.

6.1. Quantitative analysis

Since the metric space and pixel space have different scales, we cannot directly compare the ground truth translation and pixel translation. Therefore we compare the ratios of readings given after displacement of 15cm and 30cm in metric space. The mobile is displaced both along X and Y axes by these distances. The readings and ratios measured are recorded in the table 6.1.

We can see that the pixel space and metric space are linearly related and shall have a one-to-one mapping transformation. Translation computed by both methods is almost similar because in theses cases the proposed method boils down to the baseline method. The effectiveness of proposed method comes into play once there is a complex translation and rotation motions involved as we shall see later.

Ground truth	Baseline	Proposed
15cm X-axis	27	27
30cm X-axis	54	54
15cm Y-axis	26	26
30cm Y-axis	53	52
30cm/15cm X-axis	2	2
30cm/15cm Y-axis	2.04	2

Table 1. Translation measurements

Since the baseline method does not estimate rotation the only comparison we can do is between ground truth and proposed method. The device from the start position is rotated by different angles (marked properly on table it is place) and the rotation reading is recorded. For each angle this is done five times and mean and standard deviation of these five trails are calculated. The results are shows in table 6.1 and visualized in figure 5.

We can observe that at angles multiples of $45°$ we have relatively more accurate measurements, where as there seems to be a negative error in $0° - 45°$ range and a positive error in $45° - 90°$.

Ground	T1	T2	T3	T4	T5	Mean	STD	Error
30 deg	27	28	29	29	28	28.2	0.84	-1.8
45 deg	45	46	46	45	44	45.2	0.84	0.2
60 deg	62	64	63	64	64	63.4	0.90	3.4
90 deg	91	92	89	90	89	90.2	1.30	0.2
180 deg	178	180	180	179	180	179.4	0.90	-0.6

Table 2. Rotation measurements

Some of the important features of the results of the proposed method are listed below

1. When the device is at rest there is translation and rotation readings stay at zero i.e. there is no drift at rest.

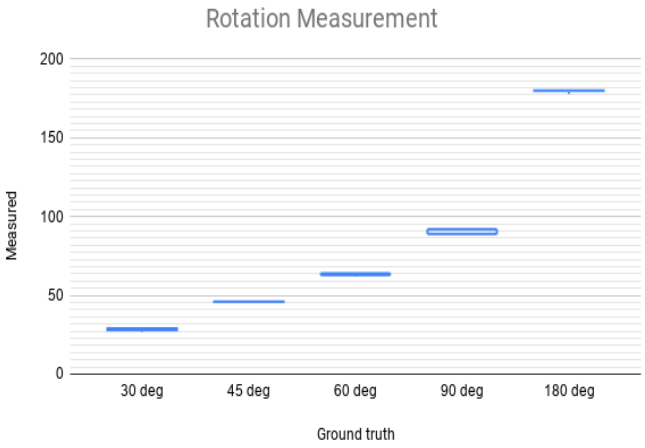

Figure 5. Rotation measurements

2. When the device is suddenly taken away from the table and placed back in the same pose later the tracking resumes from that point i.e. there is a good jerk recovery.

3. The speed at which device moves has no effect on tracking as change in pose is only a function of first frame and Nth frame.

4. The above point also means that there will be no drift accumulation during the movements.

6.2. Qualitative analysis

To compare both baseline and proposed methods in a more practical point of view we plot the paths traced by them simultaneously when the device is moved along different paths. Figure 6 shows these paths when device was moved in a line, square, circle, spiral, star fashion respectively. The green path is drawn by baseline method and the white path is drawn by proposed method. The last image in figure 6 show the letter CV written. Note that in all scenarios in figure 6 the device was never rotated. Therefore we get very correlated paths for both methods.

In figure 7 the device was rotated in a full rotation, along a small arc and finally moved forward and then rotated in a full circle. The rotations are made along axes perpendicular to the plane on which device was placed.

In these images we can see that baseline method and proposed method produce significantly different paths. Particularly in the first image in figure 7 we can see that proposed method (white) produces a good circle while the baseline method (green) produces a curve that shoots out arbitrarily. This emphasizes the role of rotation in these cases.

As a side note we can observe that although the baseline (green) path shoots out, it finally goes back to the starting point to form a closed loop when the device resumes its initial orientation.

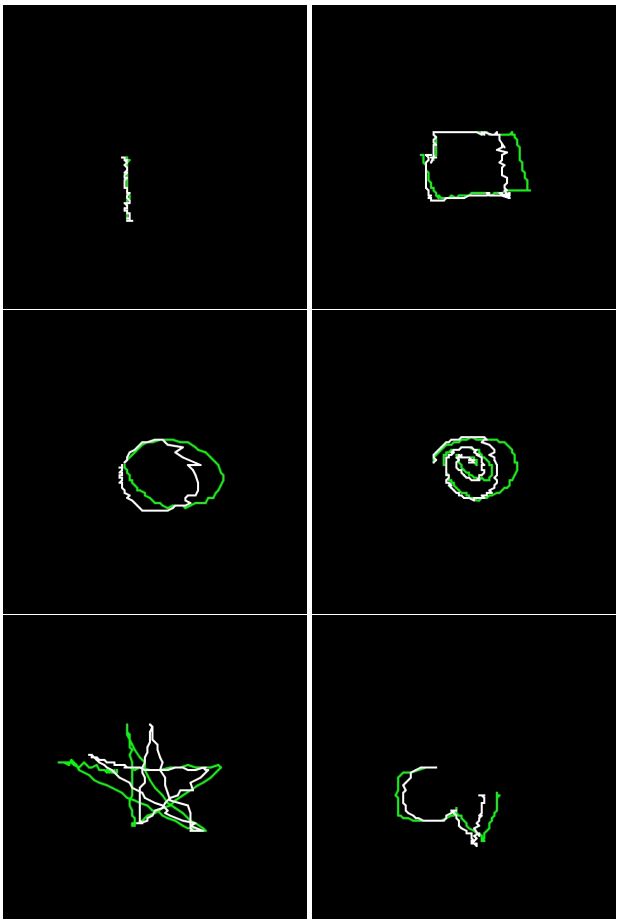

Figure 6. Translation only movements; Line, Square, Circle, Spiral, Star and Letters CV paths respectively. Green path refers to baseline method, white path to proposed method

7. Future work

Some of the possible future work is listed below

1. The algorithm compares 1st (anchor) and Nth frames. If the device moves such that all key-points of anchor frame are out of view then this approach fails

2. It only uses key-point matching technique. It doesn't take advantage of any alignment techniques like Lucas-Kanade or Inverse composition alignment

3. It uses a fixed feature detector irrespective of the nature of the scene.

The first problem can be solved by calculating displacement between consecutive frames. But this can introduce

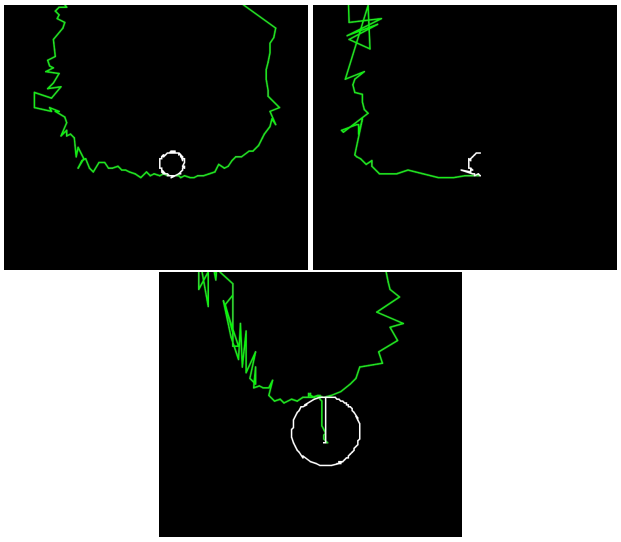

Figure 7. Translation + Rotation movements; Full rotation, Arc, Forward + Full rotation along axis perpendicular to device respectively. Green path refers to baseline method, white path to proposed method

drift in displacement over time due to the accumulation of small errors in each such calculation. Some frames might not have enough good features or contain rapid motion. Therefore a combination of 1-Nth frame and consecutive frame displacement calculations should be used, where more weight is given to the former method and the latter one can be used to correct the later one periodically.

The tracking itself can be improved by extracting image features using algorithms like Shi-Tomasi corner detector and SIFT features and apply tracking algorithms like Lucas-Kanade tracking to obtain the new positions of these features and hence calculate displacement and rotation between frames.

A simple solution to third problem is use all or a selected subset of detectors running in separate threads and choose the one with the greatest number of good key-point matches.

As one of the primary use cases of this system is to be used as an external mouse, occlusions by hand is a common phenomenon and recovery methods for such scenarios can be implemented.

As the change in pose is relative we can have a parameter to adjust so called mouse-sensitivity to adjust the motion of cursor with respect to actual motion of mobile phone. If the front camera is used as the input source then the user has access to the whole screen. Therefore the screen can be customized to any layout of buttons or operations giving the user great control and customizability.

8. Conclusion

In this paper starting from a set of assumptions we first proposed a method to estimate the translation of a mobile phone by comparing features from first and Nth frame. Then we extended that method to estimate both rotation and translation simultaneously by estimating euclidean transform. Then we compared both methods in different scenarios to conclude that the second method is as good as first one in estimating translation and has an additional capability of estimating rotation. Thes methods have good properties like no accumulation of drift, jerk recovery, independence from movement speeds. Although these methods are not fully at the level to be deployed as human-computer interaction systems like external mice and CAD tools, they show good promise for future and encourage us to look at the mobile computing world in a different perspective.

References

[1] Pablo Fern$#225;ndez Alcantarilla, Adrien Bartoli, and Andrew J. Davison. Kaze features. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ECCV'12, pages 214–227, Berlin, Heidelberg, 2012. Springer-Verlag.

[2] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, May 1987.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[4] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067, 2007.

[5] Anton Filatov, Artyom Filatov, Kirill Krinkin, Baian Chen, and Diana Molodan. 2d slam quality evaluation methods. In *Proceedings of the 21st Conference of Open Innovations Association FRUCT*, FRUCT'21, pages 120–126, Helsinki, Finland, Finland, 2017. FRUCT Oy.

[6] Georg Klein and David Murray. Improving the agility of keyframe-based SLAM. In *Proc. 10th European Conference on Computer Vision (ECCV'08)*, pages 802–815, Marseille, October 2008.

[7] Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 83–86. IEEE, 2009.

[8] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2548–2555, Washington, DC, USA, 2011. IEEE Computer Society.

[9] Mingyang Li and Anastasios I Mourikis. Vision-aided inertial navigation with rolling-shutter cameras. *The International Journal of Robotics Research*, 33(11):1490–1507, 2014.

[10] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004.

[11] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *Trans. Rob.*, 34(4):1004–1020, Aug. 2018.

[12] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[13] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Scale drift-aware large scale monocular slam. In *In Proceedings of Robotics: Science and Systems*, 2010.

[14] Yihong Wu, Fulin Tang, and Heping Li. Image-based camera localization: an overview. *Visual Computing for Industry, Biomedicine, and Art*, 1(1):1–13, 2018.