Real-time 2D pose tracking in mobile phones

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 ventures 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 as an external mouse or manipulating CAD objects in a more intuitive way.

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 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 world-frame of reference.

2. Related work

The problem of pose estimation is a well-known and widely studied research problem. [13] provides a good overview of on image-based camera localization techniques including PnP problems, Simultaneous localization and mapping (SLAM), Sturcture from Motion (SFM). Filter based SLAM method was first proposed in [3]. [12] showed that keyframe-based SLAM can give more accurate results than filter-based SLAM. [6] 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. [8] used inertial unit and a rolling-shutter camera to track motion in real-time in mobile devices. [10] proposed an optimization-based visual-inertial camera localization for mobile devices. [4] compares various 2D SLAM algorithms using different metrics. [6], [8], [10] motivate the idea of using vision techniques on mobile hardware. [5] provides a simple inter-frame rotation estimator under rapid camera movement and keyframe-based re-localization method.

3. Preliminary result

We assume a rigid environment with a decent number of good features to track over time. The device is constrained to have only translation in a 2D space (ex. on a computer desk).

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[9], SURF [2], KAZE[1], ORB[11], and BRISK[7] algorithms. Common examples of DM are FLANNBASED, BRUTEFORCE, BRUTEFORCE_HAMMING algorithms.

Let FD(F_i) represent keypoints in ith frame, len(FD(F_i)) represent number of keypoints in ith frame, DE(F_i) represent numeric descriptors of keypoints 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.

Essentially we set an anchor frame that has more than a threshold number of keypoints. Then for every subsequent frame we detect keypoints and match with the anchor frame. This way every keypoint in the anchor frame has a correspondence with a keypoint in jth frame. We filter out bad matches using a nearest neighbour ratio test. If the number

Algorithm 1: Preliminary 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

of good matches exceed a certain number we calculate displacement of each keypoint from anchor frame to jth frame. The magnitude of total displacement of camera is taken as the median of displacements of all keypoints. Th_{anchor}, $Th_{goodframe}$, Th_{NN} are tuning parameters here.

Specifically, we used the video feed from the front camera of a OnePlus 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 evaulation of the implementation is conducted. Figures 1 illustrates the result of tracking when mobile is moved in a square fashion. Figure 2 illustrates circular movement, figure 3 illustrates random closed loop movement and finally 4 illustrates a straight line movement.

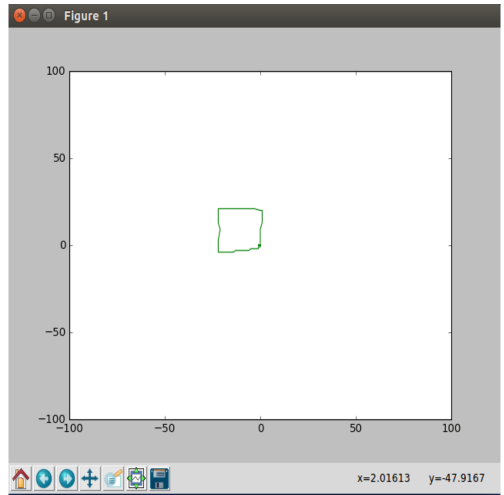

Figure 1. Square movement

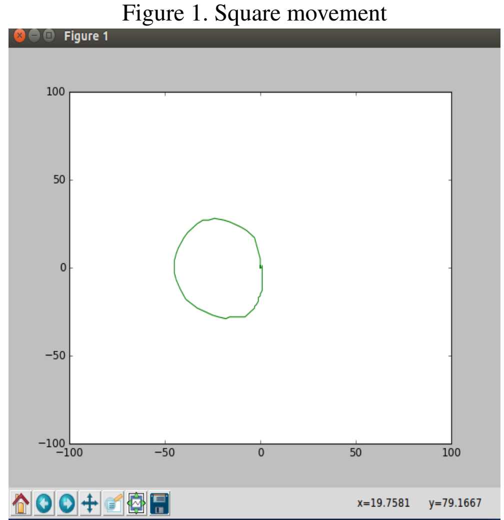

Figure 2. Circular movement

4. Next steps

While the preliminary work does a decent job for a naive tracking system implementation, it stills has some serious defects. The main defects are listed below

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

2. It does not handle rotations in 2D

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

4. 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 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. [5] provides a simple method for rotation estimation using second order minimization over SE(2) group in pixel space.

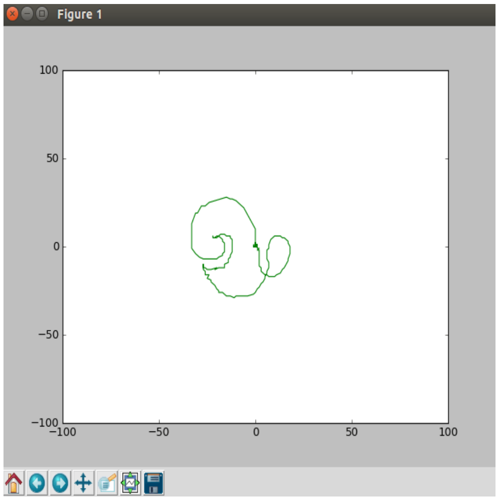

Figure 3. Random closed loop movement

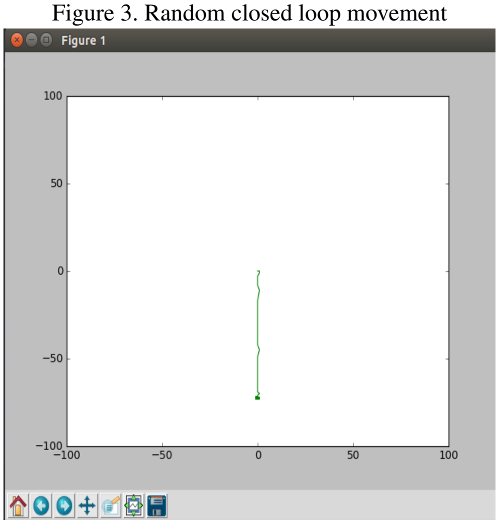

Figure 4. Straight line

A simple solution to fourth problem is use all or a selected subset of detectors running in separate threads and choose the one with the greatest number of good keypoint matches. [6] motivates the idea of using multiple threads for independent operations.

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. We plan to handle such temporary occlusions failure-recovery has to be implemented. [5] proposes a method to recover from such failures using dense feature comparison between incoming new frames and previously saved good frames and selecting the one least image difference and then aligning the current frame back to one of the reference frames.

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.

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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.