Mobile Augmented Reality: Fast, Precise, and Smooth Planar Object Tracking

Dmitrii Matveichev
Department of Computer Science
National Taipei University
Taipei, Taiwan
xapocmat@gmail.com

Daw-Tung Lin
Department of Computer Science
National Taipei University
Taipei, Taiwan
dalton@gm.ntpu.edu.tw

***Abstract*— We propose an innovative method for combining sparse optical flow tracking and descriptor matching algorithms. The proposed approach solves the following problems that are inherent to keypoint-based and optical flow-based tracking algorithms: spatial jitter, extreme scale transformation, extreme perspective transformation, degradation in the number of tracking points, and drifting of tracking points. Our algorithm provides smooth object-position tracking under six degrees of freedom transformations with a small computational cost for providing a high-quality real-time AR experience on mobile platforms. We experimentally demonstrate that our approach outperforms the state-of-the-art tracking algorithms while offering faster computational time. A mobile augmented reality (AR) application, which is developed using our approach, delivers planar object tracking with 30 FPS on modern mobile phones for a camera resolution of 1280x720. Finally, we compare the performance of our AR application with that of the Vuforia-based AR application. The test results show that our AR application delivers better AR experience than Vuforia in terms of smooth transition of object-pose between video frames.**

I. Introduction

Our work is focused on planar object tracking (POT) problem and its application in mobile augmented reality (AR). The POT problem can be formulated as follows: find the precise planar object position on a sequence of video frames, using known object image.

II. Overview of Proposed Algorithm

We propose to combine OF tracking and image patch descriptors into one robust tracking algorithm. The algorithm is split into two phases: detection and tracking.

The detection phase is performed, If the object was not detected in the previous frame. Detection is done by means of ORB binary descriptors detection and matching. Based on matched points we compute pose and output: homography transformation matrix; frame image; detection result (whether an object was detected or not). The tracking phase is performed if the object was detected in the previous frame. For every frame in the tracking phase, the following steps are performed. 1) Using the homography matrix, warp the current and previous frames so that the detected object is located in the center of a frame and is of the same size as that of the original object image. 2) Using the homography matrix, project the object points p_t to the current warped frame (points p_p). 3) Do sparse OF point tracking, with following input: warped previous frame image, warped current frame image and projected

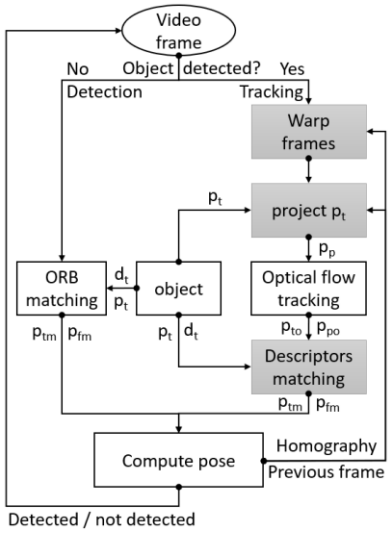

Fig. 1. (dt: object descriptors; pt: 2D object points; ptm: matched object points; pfm: matched frame points. pp: projected object points; ppo: successfully tracked projected object points; pto: successfully tracked target points)

object points (p_p). 4) Flter the tracked points (p_{to} and p_{po}) using the descriptor matching based approach (it is different from traditional descriptors matching, and you may find detailed description in our paper). 5) Project matched frame points (p_{fm}) back to the current frame. 6) compute the object pose using matched points (p_{tm} and p_{fm}). 7) output of each frame: homography transformation matrix, frame image, and detection result (whether an object was detected or not).

Therefore, by its design OBD: eliminates the drifting problem (guarantee that the tracked points correspond to the same points on the template image), eliminates OF points number degradation problem (thanks to object points projection, number of input points for OF is always equal to the number of object points), increases the accuracy of computed pose and maintain smooth object pose transition between frames. Additionally, in OBD algorithm the descriptors matching is basically one XOR operation between two matrices of the same size. Thus, it is not computationally expensive.

III. Evaluation

To evaluate OBD and compare it with state-of-the-art POT algorithms, we used “Planar object tracking in the wild: A benchmark [4]. As evaluation metrics we used alignment error (5) and spatial jitter (8).

$$e_{al} = \frac{1}{4}\sum_{i=1}^{4}\sqrt{(x_i - x_i^*)^2 + (y_i - y_i^*)^2} \quad (5)$$

$$J_t = \sqrt{(x_i - x_{iprev})^2 + (y_i - y_{iprev})^2} \quad (6)$$

$$J^* = \sqrt{(x_i^* - x_{iprev}^*)^2 + (y_i^* - y_{iprev}^*)^2} \quad (7)$$

$$J = \frac{1}{N}\sum_{i=1}^{N} J_t - J^*. \quad (8)$$

A. Results analysis

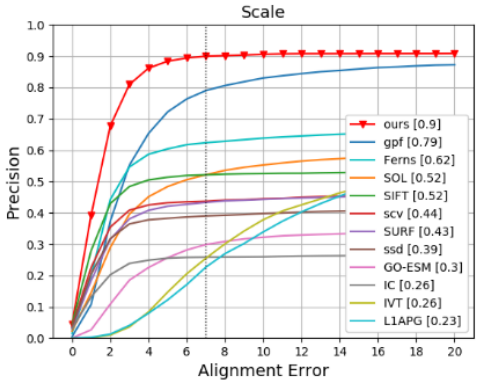

Fig. 2. Scale video sequences evaluation

In the scale video sequences OBD dramatically outperformed other trackers and successfully detected the object pose in 90% of the frames (with eAL = 7).

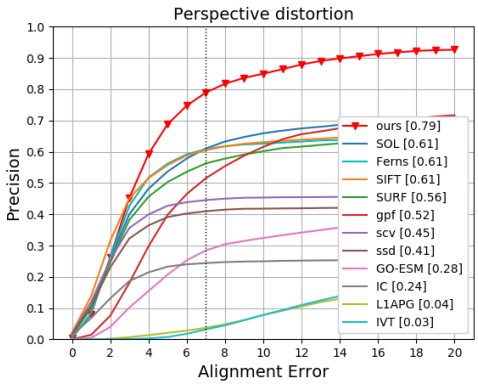

Fig. 3. Perspective distortion video sequences evaluation

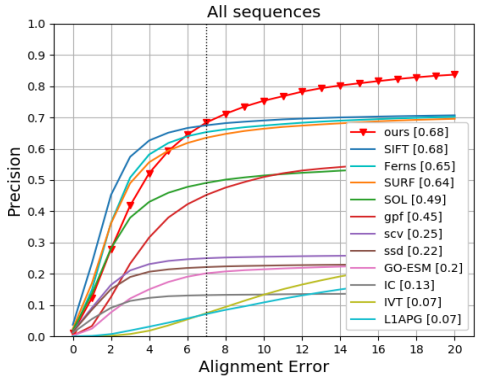

Fig. 4. All video sequences evaluation

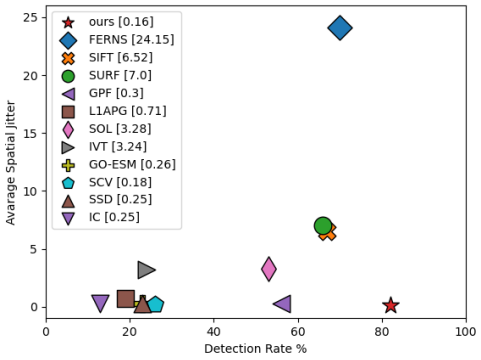

Fig. 5. Average spatial jitter POT algorithms. (detection rate: percentage of frames with eAL smaller then 20.)

In the perspective distortion video sequences OBD also outperformed all the other trackers and successfully detected the object pose in 79% of video frames (with eAL = 7). Finally, OBD demonstrated the best average performance across all sequences, and it successfully detected object for 82% frames (with eAL less than 20).

OBD has almost zero spatial jitter across all video sequences with the highest detection rate. GPF [28] tracker has almost the same spatial jitter, however its detection rate is only 56% and all other POT algorithms with low spatial jitter has detection rate below 30%.

B. Proof of Concept Mobile AR Application

To evaluate the OBD performance on mobile platforms, we implemented an AR application using the Unity 3D platform [34]. We used the OpenCV library [33] as a native plugin for the Unity 3D platform. With more than 30 different planar objects in matching database modern phones (such as Google Pixel 3 and iPhones older than iPhone 6s) run with 30 FPS on 1280x720 camera resolution. In Fig. 6 we demonstrate the example screenshots taken using the developed mobile AR application.

To compare our AR application with other state-of-the- art AR products, we developed a Vuforia [3] based AR application. For Vuforia and our AR application databases, we used identical template images. Our tests showed that for some objects (such, as objects in Fig. 6), the object pose detected by Vuforia had spatial jitter, while our AR application provided precise and smooth object-pose transition between camera frames.

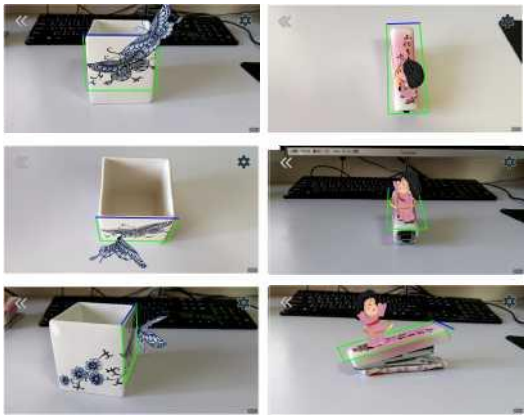

Fig. 6. Mobile AR example screenshots for two different planar objects with rendered 3D models