Real-Time Monocular 6DoF Object Pose Tracking on Smartphone GPUs

Valentin Miu
vmiu@bournemouth.ac.uk
Centre for Digital Entertainment, Bournemouth University
Beauty Labs International Ltd., Cambridge
Oleg Fryazinov
ofryazinov@bournemouth.ac.uk
The National Centre for Computer Animation,
Bournemouth University

Six degrees of freedom (6DoF) monocular pose estimation is an essential part of augmented reality (AR) and mixed reality pipelines, and machine learning-based solutions currently dominate the cutting edge. Since most end consumer-oriented AR targets smartphones, this creates two limitations: the need to optimize for real-time operations on low-powered devices, and the restriction to monocular pose estimation, as opposed to more precise stereo camera, multi-camera or depth camera pose estimation often used in robotics and autonomous driving.

In this paper we present an approach that allows a real-time 6DoF pose tracking for objects in RGB video. By using a modified set of mobile-specific convolutional neural networks and simple tracking, we are able to achieve fast performance on modern and mid-range smartphones (Android and iOS). We show how tracking a moving region of interest (a method previously used for face, hand and bounding box tracking) can drastically improve the performance of 6DoF pose estimations with machine learning.

The system is implemented by using TensorFlow Lite inference of the machine learning model in Unity, but the tracking technique is applicable to any augmented reality platform, on both smartphones and desktop computers. The tracking scheme consists of two architecturally identical networks, but with differing input sizes and training parameters. The first is the detection network, with an input size of 224x384, which is trained on the entire input frame. The second is the tracking network, which is trained on a 224x224 square crop around the padded ground truth 2D bounding box of the object. Both networks are based on the single-shot-pose approach [3], outputting the 2D projections of the 3D bounding box corners and centroid using a modified YOLOv2 network. To optimize the networks for smartphone usage, we replace the feature extractor (originally Darknet-19 in) with MobileNetV2.

The tracking scheme follows a simplified version of that in BlazeFace [1]. This uses the detection network once and the tracking network on every subsequent frame, using a crop around the previously detected keypoints. If the tracked detection confidence drops below a threshold, the full-frame detection network is run again to re-detect the object.

Part of the training data was created by manually aligning the CAD model of the object with ground truth "keyframes", and then interpolating between these using a traditional tracking algorithm ([2]), which gave intermediate poses. The rest was generated by attaching a VIVE tracker to the base of the curler. The tracker was removed from the frames automatically with Photoshop, using the projection of the posed tracker CAD model. About 47000 frames were annotated in total, and overlaid onto random backgrounds from the Pascal VOC dataset. The foreground was interpolated between just the curler and the curler and subject holding it, using a color-difference weighted distance function.

The system was trained and used to track handheld curlers from an RGB video stream, using differently sized bounding boxes. Both networks are trained for 100 epochs. Some visual results (on isolated frames, without tracking) are shown in Fig.1. While the full-frame model cannot accurately predict pose, it is sufficiently accurate to provide a region of interest to be tracked.

To evaluate the tracking scenario itself, we use a group of 597 unseen annotated frames and isolate frame sequences with some degree of temporal coherence (that is, the time difference between subsequent frames is less than 0.5 seconds). We use a confidence threshold of 0.4 for the tracking network, under which we fall back to the full-frame detection network. The results are shown in 1.

Preliminary benchmark results on the YCB dataset show reasonably high values for about two thirds of the objects. Due to the weaker but faster feature extractor used, they are noticeably lower than state-of-the-art. The remaining third of the objects show unusually low accuracies, which appear to be due to the lack of occlusion augmentations during training.

To use machine learning on mobile with Unity, we use a proprietary augmented reality system, using a C++ plugin integrating the C++ API of TensorFlow Lite, called from Unity with C# bindings. This system allows for inference of arbitrary models within Unity on Android and iOS CPUs and GPUs, as long as they are supported in TensorFlow Lite. We also use it for rapid facial segmentation, facial keypoints (which also use tracking for eye and mouth keypoints and segmentation), and fingernail segmentation with internally designed and trained models.

These results show that tracking with separate convolutional neural networks can significantly improve the performance of this estimation and make it applicable for real-time applications, such as augmented reality. While the precision of the inferred pose is limited, it is shown to be greatly improved by the use of the tracking network to increase effective resolution. Given the importance of minimizing input size for smartphone-oriented machine learning models, restriction to a region of interest is likely required to ensure sufficiently rapid inference, especially on less performant mobile devices.

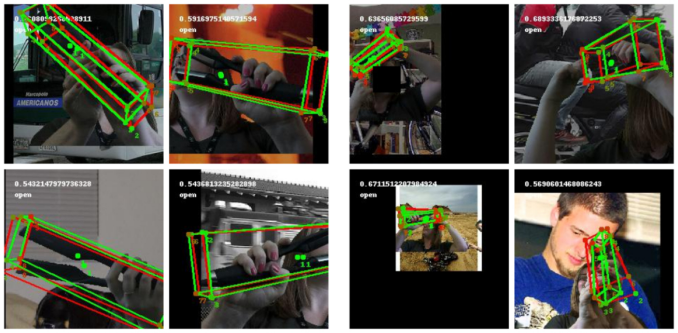

Figure 1: Some evaluation results for cropped tracking network (left four), and detection network (right four, run in these examples at 224x224 on square inputs). Ground truth is shown in green, prediction in red

Table 1: Average errors of the tracking and detection networks during tracking on video, both using the MobilenetV2 feature extractor network. The 224x224 input-sized networks are the tracking (square crop) networks, while the 224x384 are the full-frame detection networks. Mean intersection over union (mIOU) is provided for the detection network only, as it is relevant to the quality of the tracking crop region

Input	mIOU	X-axis error	Y-axis error	Z-axis error
224x224	-	37.529°	37.538°	21.617°
224x384	0.817	82.006°	89.553°	58.145°

[1] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs. 7 2019. URL http://arxiv.org/abs/1907.05047.

[2] Changhyun Choi and Henrik I. Christensen. 3D textureless object detection and tracking: An edge-based approach. *IEEE International Conference on Intelligent Robots and Systems*, pages 3877–3884, 10 2012. ISSN 21530858. doi: 10.1109/IROS.2012.6386065. URL http://ieeexplore.ieee.org/document/6386065/.

[3] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-Time Seamless Single Shot 6D Object Pose Prediction. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 292–301. IEEE, 12 2018. ISBN 9781538664209. doi: 10.1109/CVPR.2018.00038. URL https://ieeexplore.ieee.org/document/8578136/http://arxiv.org/abs/1711.08848.