Developmentally-Inspired Learning of a Robust Face Tracker

Greg Borenstein
MIT Center for Brains, Minds, and Machines: Aspects of a Computational Theory of Intelligence
15 December 2014

Abstract

In their 2014 paper, "From simple innate biases to complex visual concepts" [1], Ullman, Harari, and Dorfman demonstrated a process for unsupervised learning of object trackers inspired by early childhood development. This work attempts to extend that methodology to bootstrap a face tracker that is robust to occlusion, rotation, and transformation. We start with visual tools that approximate those available to infants, namely a frontal face detector and a motion estimator. We use these tools to model the spatial continuity of faces between detections, producing an estimated motion track for the face. Based on this track we capture a series of face images in poses that are not recognized by the frontal face detector. These images are used to train a new face detector that can supplement the existing frontal face detector in order to improve detection results in alternative poses. We evaluate this approach on an embarrassingly small dataset and point towards next steps and challenges to improve and formalize this work. Additionally, we modestly brag about the integration of a number of tools from this process into the open source Processing creative coding toolkit and we suggest some intriguing additional ideas for how to use this approach to more effectively model occluded faces.

Background

In [1], Ullman et al define a methodology for learning object detection that emulates infant cognitive development. They start with computer vision tools that parallel "a plausible innate or early acquired bias based on cognitive and perceptual findings". They then use these tools to gather a set of training images from a series of videos in an unsupervised manner. These images set then form the basis for a conventional machine learning process through which they train an object detector.

The specific use case undertaken in [1] was hand tracking. Ullman et al tracked a series of "mover" event in which a subject's hand manipulated objects on a table. These mover events are defined by sequences in the video in which the state of an area of the image differed before and after motion was detected (due to the subject's hand having displaced an object there). They cite the cognitive development literature to argue that this mover detection capacity constitutes "an empirically motivated innate mechanism" that is present in infants. Starting solely with these mover events, they were able to capture a set of images and accompanying labels that allowed them to train a robust hand tracker without further supervision.

In this study, we seek to use a similar methodology to improve on the results of the widely-used Viola-Jones face detection technique. [2] Studies of the development of infant vision have shown a preference for frontal face-like stimuli as young as six weeks and with rapid

improvement between six and 12 weeks. [3] We take this as motivation for using Viola-Jones with a frontal face cascade as a starting point.

The developmental literature also demonstrates that infants have significant capacity for detecting motion. A survey of the literature by Franz Kauffman summarized the results thusly: “infants are able to detect visual motion very early in life and do extract information which leads to the perception of form.” [4] Infants as young as one month have been shown to detect very rapid motion at a level comparable to adults. Detection of slower forms of motion develop steadily with a significant improvement already demonstrated by three months. Interestingly, the last capacity to develop seems to be the detection of very slow motion in the presence of occlusion, which children struggle with up to the age of four. [4]

We chose to model motion using optical flow, specifically, the Farnebeck dense motion estimation algorithm [5]. We felt that this per-pixel motion estimate was a more accurate match for infant vision than a sparse feature-based optical flow variant.

In a series of videos of moving and occluded faces, we used an optical flow-based motion estimate to capture training examples whenever a Viola-Jones detection was not available. We then used these images to train a Support Vector Machine [6] classifier and used this classifier in a multi-scale search to detect occluded or non-frontal faces in images.

Process

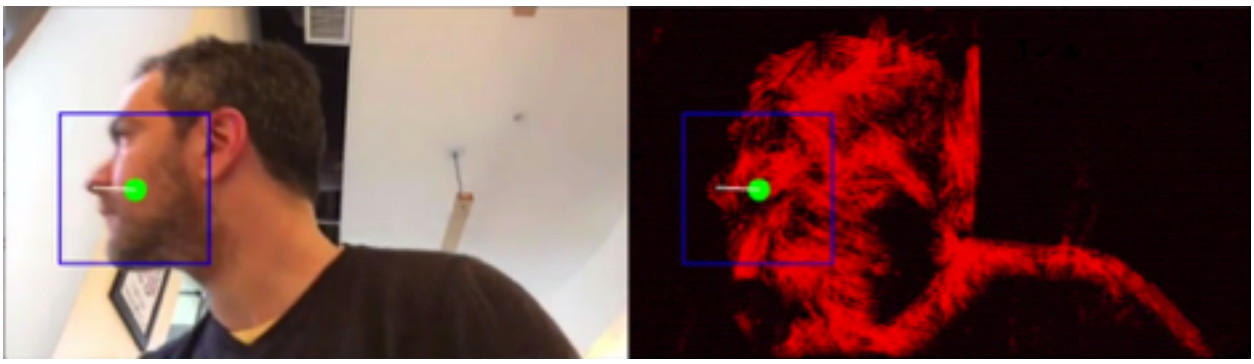
This section documents the process and challenges of implementing the proposed technique. All coding was done in the Processing creative coding toolkit. [8] The steps involved were as follows: I learned to use OpenCV’s dense optical flow algorithm and then wrapped it to make it usable in Processing. I built a hybrid FlowTracker that tracked faces with Viola-Jones and then continued the track when the face was lost using the average optical flow around the last-seen face position. I refined this track by cutting it off after after too much time and motion occurred since the last detection. I used this track to capture a series of images and labels to train a classifier. I designed a feature vector using histogram of oriented gradients and trained a model using Support Vector Machines. I explored OpenCV’s built-in facilities for multi-scale object detection with a custom model, found significant technical problems and implemented my own primitive multi-scale detection system. I built a tool to capture user annotations of videos to establish ground truth in order to evaluate the face tracks created by various iterations of my detector. I automated the process of comparing detector outputs to the ground truth track in order to produce a numerical evaluation.

All of my code is available at [13].

Implementing Optical Flow with OpenCV

OpenCV implements two flavors of optical flow: Farneback, which is a dense approach and PyrLK which is a sparse, feature-based approach. After some investigation of both of these techniques, I chose to go with Farneback because its is more biologically realistic (i.e. using motion throughout the visual field rather than some more specific concept of higher level features), closer to the approach taken in the Ullman paper, and easier to implement.

Having made that decision, I wrapped the OpenCV Farneback functions into OpenCV for Processing, the OpenCV wrapper for Processing that I maintain (landed in version 0.5.2). [7]



AN EXAMPLE OF MAINTAINING AN ACCURATE FACE TRACK USING OPTICAL FLOW IN THE ABSENCE OF A VIOLA-JONES FACE DETECTION IN ORDER TO CAPTURE TRAINING IMAGES.

Tracking with Optical Flow in the Absence of Faces

With the optical flow implementation in hand, I used it to build a FlowTracker class responsible for tracking the movement of a given point in the image based on the surrounding optical flow. The FlowTracker picks up from the center of the last seen face on any frame where the face disappears and tracks based on the optical flow in the video within a square around that point. Implementation available at [10].

You can see video of the FlowTracker augmenting the OpenCV face detection at [9]. (The face rectangle is green when OpenCV's Viola-Jones detection is responsible for it and blue when it's being updated through optical flow. The white line in the middle of the image represents the average flow within the face rectangle at any given point in the video.) Once the FlowTracker was up and running I used it to capture images from my test video as an initial input to the machine learning module.

Currently, the FlowTracker produces some false positives that should be easily eliminated: it should have a decay rate so if it hasn't seen a face in a while it doesn't just wander into some part of the image. It should also deal better with boundaries of the image. In the meantime, I separated the good samples from the bad ones by hand, setting aside the bad ones to use as negative examples

---

Feature Selection and OpenCV HOG Descriptor Implementation

Based on previous experience doing object detection (and various literature we looked at this semester for computer vision object detection and how edge processing works in the ventral visual stream) I decided to use Histogram of Oriented Gradients [11] as the source of my features for these images.

OpenCV includes a HOGDescriptor class that can compute HOG features for a given image. It also has a detectMultiScale function that can do multiscale detection within an image based on a trained classifier.

After some poking around with this obscure API, I was able to successfully compute the HOG descriptors and use them in an old example I had around that used HOG plus an SVM classifier to do hand gesture recognition. That example had used a separate HOG library so I was able to validate that substituting in this new OpenCV implementation worked successfully.

One troubling thing I learned in this process is that while OpenCV's detectMultiScale functionality can accept an SVM model to do the detection, absurdly, it cannot use the SVM models trained by its own machine learning module. Or by libsvm [12] for that matter. There seem to be some people online who have figured out a transform to do on the support vectors yielded by libsvm in order to get detectMultiScale to accept them. I'll burn that bridge when I come to it, i.e. after training is working well I'll see if I can figure out this transform or I can just manually implement some less efficient version of windowed search as a stopgap.

Setting Up a Machine Learning Environment

Once I had HOG feature calculation up and running, I dove into setting up a training environment. I'd previously written a wrapper class for OpenCV's various machine learning classes (and libsvm since OpenCV's svm implementation is based on an old, crappy version of libsvm). It includes a unified interface to many different classification algorithms and some tools for doing things like crossfold validation, calculating evaluation metrics, etc.

I took the hand-labeled images from my initial optical flow-based tracker used them as a training set. Here's some numbers from the initial results (with 4-fold cross validation):

========CUMULATIVE RESULT (4 folds)================
accuracy: 0.87261903
precision: 0.9166667
recall: 0.7916667
f-measure: 0.81666666

Looking pretty good.

This is based on a really small data set as it's just from one video of me waving my face around in front of a camera. So, the next step is to expand the data set to more videos and see if these numbers hold. And, in parallel to work on improving the selection of positive and negative training examples in the FlowTracker to get the training process fully automated. Once that's working, all that will be left is implementing multiscale detection.

After testing a number of machine learning algorithms (KNN, RandomForests, AdaBoost, and SVM). SVM performed the best by a significant margin, which is convenient since it is also the only model supported by OpenCV’s detectMultiScale functionality, as mentioned above.

Track Decay and Automatic Negative Example Selection

In order to improve the classification results I modified the FlowTracker code to reduce the number of face-free images captured as positive training examples. As time passes since the last Viola-Jones detection the FlowTracker's results are less reliable. Therefore, in order to reduce the number of false positives I added a check that looks for a large amount of total motion having occurred since the last face detection and ceases adding new positive training samples. After some experimentation with a number of test videos the parameters I landed on were to cutoff new samples when both more than 300ms have passed and the tracker has moved more than 100 pixels based on the optical flow input.

In order to add negative examples, I had the training program randomly sample 100x100 pixel regions of the image in any frame where no face was detected. While this created some negative samples it resulted in far fewer than there were positive samples. An additional enhancement would be to grab a part of the image not overlapping a face in frames where faces were detected. Lacking that I simply pooled negative examples from many test videos since random bits of wall and background should all be approximately the same. In fact, the greater variety of negative samples the better, most likely.

Building a Training Set

Once this refinement to the sample capture strategy was in place I recorded a series of videos with different forms of face motion for training and testing. I made videos with only rotations, videos with only occlusions, and videos with both rotations and occlusions. In each category I made multiple videos so I could train the classifier on one and test on the others. While big enough for basic validation this data set is ridiculously small. It consists entirely of videos of me and only captured in a small number of environments. (As discussed below, one of the most obvious next steps to improve this project would be to create a larger and more diverse training and evaluation data set.)

Multi-Scale Detection

Once I had a trained model, in order to use it on real video I needed the ability to perform detections at multiple scales and locations within an image. As mentioned above, OpenCV's highly efficient detectMultiScale implementation is not compatible with either model files created by Libsvm or OpenCV's own CvSVM class. I spent a significant amount of time attempting to massage the model file outputted by Libsvm including writing a complete parser for its file format. [14]

Unfortunately, even after all of this work while I was able to get detectMultiScale to load my model file I was never able to get it to perform detections successfully. So, after a couple of days of wrestling with this problem, I gave up and implemented my own (very primitive and inefficient multi-scale search). My implementation performs multiple copies of the image's pixels which makes it slow and limits it to only searching a small subset of the image and only at a limited number of scales to achieve reasonable performance.

To use this primitive multi-scale detection technique to best effect, I setup my tracker to search around the last location seen by the Viola-Jones face tracker and then to update the center of the search only when a positive detection was made. Still the result is significantly worse than a proper OpenCV-style image pyramid implementation and one of the biggest improvements to this project's results would be to complete that implementation. Another significant research contribution could be made by rewriting OpenCV's HOGDetector detectMultiScale function so that it can accept any CvStatModel subclass (i.e. any of OpenCV's own machine learning classes).

Hand-Labeling for Evaluation

Finally, in order to evaluate the results of my tracker, I built a simple application that let me hand-label test videos to establish a ground truth against which individual variations of my tracker could be tested. This application plays back a video while recording the mouse position, which a user keeps centered over the face of the subject. The user holds down a key to indicate that a face is present and releases the key to indicate the absence of a face. Track position and face presence data for each frame are recorded in a csv file.

I also added functionality to my tracker detection script to output a csv file in the same format based on the track calculated by both the plain Viola-Jones detector and my enhanced detector. Then I wrote another simple evaluation application that compares a ground truth track created by hand with a track calculated by any of the detectors on the same video.

I present the results of this evaluation below in the Conclusion section.

Evaluation

In order to evaluate the effectiveness of the approach presented here we evaluate three versions of the tracker in three different scenarios, comparing each one to the results of the baseline un-enhanced Viola-Jones face tracker. The three scenarios are as follows: a series of videos consisting of only facial rotations, a series of videos containing only occlusions of the face (by hands and arms), and a series of videos congaing complex naturalistic scenes involving a moving subject, facial rotations, and occlusions. In each of these scenarios we test three different models with our detector: one trained on only rotations, one trained on only occlusions, and one trained on both rotations and occlusions. In all of the scenarios using our custom trackers, the tracker is only invoked when Viola-Jones fails to find a face. Therefore, we're looking to see how well these trackers can supplement that baseline tracking strategy. Beyond the differences in the positive examples included in the training set, all other model parameters were kept the same between training sessions. None of the training images were drawn from videos used for testing.

The evaluation results were as follows:

Test Type	Training Set	Accuracy	False Positives	False Negatives	Average Distance Off (pixels)
rotation	Viola-Jones	0.496	3	593	104.3
	Rotation+VJ	0.789	16	234	83.2
	Occlusion+VJ	0.571	4	504	83.7
	Hybrid+VJ	0.801	119	117	86.4
occlusion	Viola-Jones	0.809	0	164	119.8
	Rotation+VJ	0.858	47	75	100.7
	Occlusion+VJ	0.818	17	139	109.3
	Hybrid+VJ	0.847	103	28	100.8
complex	Viola-Jones	0.638	0	439	52.6
	Rotation+VJ	0.807	2	232	39.7
	Occlusion+VJ	0.644	0	431	50.8
	Hybrid+VJ	0.853	74	104	39.4

As you can see, all of the detectors significantly improved on the baseline Viola-Jones tracker in both overall accuracy and (slightly less consistently) in the average distance of deviations from the position of the ground truth track. However, there are some significant difference between the quality of the models that are worth discussing.

While the model trained with positive samples drawn from both rotation and occlusion sets (referred to here as the hybrid model) had a slight edge on accuracy in both the videos featuring face rotations and complex motion, this improvement came at the cost of a large increase in the number of false positives. One of the chief characteristics of the performance of Viola-Jones face trackers is their very low rate of false positives. This hybrid model, on the other hand displayed false positives ranging from 6% and 12% of the frames in each video. This strongly implies that the hybrid model is biased towards a positive classification response.

The model trained with only samples of an occluded face, on the other hand, showed the least improvement over the vanilla Viola-Jones tracker. While it produced the fewest false positives, it also only significantly improved on the vanilla tracker in the case of the rotation-only tests and even then it did so significantly less than the other two models. And, in fact, the classifier trained only with rotation samples even outperformed the occlusion-trained tracker on the occlusion-only test set.

Which brings us to the most successful of these three models: the one trained from the video demonstrating only rotations of the face. This tracker significantly outperformed the baseline on all three tests and showed only a modest increase in false positives compared to the hybrid model. Its performance on the rotation-only set is particularly impressive where it almost doubled the accuracy of the Viola-Jones tracker with only 14 frames of false positive

predictions. The results on the more realistic complex et is almost as impressive with a nearly 20% improvement over the baseline with only an additional two false positives.

Given the conclusion described in [4] that perception of moving objects under occlusion develops in children as late as four years old, the results here around occlusions are quite intriguing. Particularly, the difficulties of false positives found when integrating occluded examples in the training of the hybrid model presents a potentially powerful parallel to that result in infant development.

Conclusions

This project demonstrates that Ullman and Harari's methodology for bootstrapping an object tracker using developmentally realistic unsupervised learning techniques shows significant promise for successful application to face detection. We built the basic infrastructure and performed the initial tests of a system that uses optical flow to capture training samples from the false negatives of a traditional Viola-Jones face tracker in order to train a more sophisticated detector that is robust to rotation and occlusions of the face and can significantly supplement a Viola-Jones tracker. We showed that such a detector trained specifically on a rotating face significantly outperforms one trained on a face with occlusions or one trained with both rotations and occlusions.

The obvious next step to expand this initial exploration would be to create a substantially larger, more diverse, and more systematic set of videos for training and testing as well as the hand-labeled ground truth to go with them. Ullman and Harari use over an hour of total video time in training their model; this work involves substantially less than ten minutes of video. Additionally, a better multi-scale search implementation would provide a more effective test of the results of the trackers trained here and likely dramatically improve the results.

On a more substantial level two interesting research questions emerged from this exploration. First, it seems possible to integrate optical flow directly into the final tracker rather than simply using it in the training phase. Specifically, a motion estimate could be used as an additional factor in evaluating a potential face tracker prediction to determine if it is consistent with the local motion of the video.

Secondly, these results imply that occlusions are much more difficult to overcome than rotations. And, further, that attempting to integrate examples of occluded faces into an overall training set weakens the quality of the entire set because it significantly complicates the decision boundary between samples that should be predicted positive and negative. Considering Ullman and Harari's original "mover" work, we wonder if a similar approach could be taken here to specifically detect occlusions of the face. A series of images of the face could be captured based on a Viola-Jones detector enhanced with optical flow as we did here, but then these samples could be aligned based on image features and compared in order to produce data similar to the mover events in Ullman and Harari's work. Potentially such an approach could model an occluded face at a finer-grained level and separate these cases of an occluded face from a simply rotating one which seems potentially tractable using the approach explored here.

Finally, an additional benefit of this project was that a number of the techniques used here were integrated into OpenCV for Processing [7], an OpenCV wrapper for the Processing creative

coding library [8] that attempts to make it easy for artists, designers, and students to work with and learn computer vision. In the course of working on this project, modules for optical flow and a number of machine learning algorithms were developed and refined and the optical flow module was landed in OpenCV for Processing and released.

References

[1] Ullman, Shimon, Daniel Harari, and Nimrod Dorfman. "From simple innate biases to complex visual concepts." Proceedings of the National Academy of Sciences 109.44 (2012): 18215-18220.

[2] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.

[3] Mondloch, Catherine J., et al. "Face perception during early infancy." Psychological Science 10.5 (1999): 419-422.

[4] Kaufmann, Franz. "Development of motion perception in early infancy." European Journal of Pediatrics 154.4 (1995): S48-S53.

[5] Gunnar Farneback, Two-frame motion estimation based on polynomial expansion, Lecture Notes in Computer Science, 2003, (2749), , 363-370.

[6] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

[7] https://github.com/atduskgreg/opencv-processing/releases/tag/v0.5.2

[8] http://processing.org

[9] https://vimeo.com/114073283

[10] https://github.com/atduskgreg/facemover/blob/master/facemover_training/FlowTracker.pde

[11] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[12] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[13] https://github.com/atduskgreg/facemover/

[14] https://github.com/atduskgreg/facemover/blob/master/facemover_learning/LibsvmModelLoader.pde