CS231A Final Project: Body Pose From Optical Flow

Winnie Lin
Stanford University
winnielin@stanford.edu

Boris Perkhounkov
Stanford University
secondauthor@i2.org

1. Introduction

1.1. Overview

Given videos with human movement as input, we wish to track the position and velocities of body joints for each extracted frame. We experiment with various methods involving combining optical flow and other visual cues, producing results that are less than satisfactory, but gaining quite a bit of insight in the process.

1.2. Motivation

The joint tracking problem is one that was initially motivated by applications within markerless motion capture, and it can also be further considered as a subcase of the human pose estimation problem, which has been widely studied and researched but mostly in the context of estimation from single images. Only in recent years have there been more work focused on techniques involving video input and temporal data, and a few pieces of literature that have been valuable to us are briefly outlined below.

2. Existing work

2.1. Review of previous work

We were initially inspired by the Fragkiadaki *et al*. [3] paper on human pose prediction, where temporal data was utilized in the form of recurrent neural networks. Originally intending on utilizing neural networks in our method, [8] provided a solution for temporal data extraction via representation as optical flow.

Romero *et al*. [6] presented a method of human pose tracking through optical flow, utilizing flow differences of various positions as a feature vector to perform body part segmentation on sequential images. Other papers such as [7] [9] also utilize motion for pose tracking by the tracking of rigid body parts, but operate under stricter limitations where the body moves relatively parallel to the image plane and does not change direction.

Figure 1. Joint positions overlaid on optical flow (visualized with hue-coded angles and corresponding magnitudes)

2.2. Comparison

We explore different possibilities of utilizing optical flow in joint position tracking, using no prior knowledge of human bone structure. Similar to [6] we devise a method highly dependent on optical flow, and attempt to combine it with other visual cues such as image patch similarity and feature detection via Support Vector Machines trained on Histogram of Oriented Gradients.

3. Technical description

3.1. Technical Background

Following is a description of the techniques and algorithms we experimented with within this project.

3.1.1 Farneback Optical Flow

Optical flow describes the motion of pixel values throughout the image, and extractions can be classified as

1. Sparse feature-tracking flow that is Lagrangian in nature, where feature points of interest are tracked through time, and
2. Dense position-based flow that is Eulerian in nature, where flow vectors are calculated for each pixel position for every frame.

In this project, we obtain the dense optical flow per image using an implementation of Farneback Optical Flow[2], where the flow field is generated first via approximating the neighborhood of a fixed pixel using a quadratic approximation for consecutive frames, then by approximating the difference between two quadratic equations as some linear translation.

Mathematically, this implementation is given as follows: Given pixel position p, its neighborhood $\mathcal{N}(p)$, and the corresponding image rgb values $I_t(\mathcal{N}(p))$ at frame t, we approximate

$$\begin{aligned} I_t(\mathcal{N}(p)) &\simeq [f_t(x)]_{x\in\mathcal{N}(p)} = x^T A_t x + b_t^T x + c_t \\ I_{t+1}(\mathcal{N}(p)) &\simeq [f_{t+1}(x)]_{x\in\mathcal{N}(p)} = x^T A_{t+1} x + b_{t+1}^T x + c_{t+1} \\ [f_{t+1}(x)]_{x\in\mathcal{N}(p)} &\sim [f_t(x+d_t)]_{x\in\mathcal{N}(p)} \end{aligned}$$

Solving for this and setting A to be symmetric with the property $x_T A x = x_t \frac{1}{2}(A + A^T)x)$, we get

$$\begin{aligned} f_{t+1}(x) =& x^T A_{t+1} x + b_{t+1}^T x + c_{t+1} \\ =& (x+d_t)^T A_t (x+d_t) + b_t^T (x+d_t) + c_t \\ =& x^T A_t x + ((A_t + A_t^T)d_t + b_t)^T x \\ &+ d_t^T A_t d_t + b_t^T d_t + c_t \end{aligned}$$

$$\Rightarrow A_t = A_{t+1}\ ,\ d_t = (2A_t)^{-1}(b_{t+1} - b_t)$$

$A_t \neq A_{t+1}$ in most practical cases, therefore d_t is computed as

$$d_t = (A_t + A_{t+1})^{-1}(b_{t+1} - b_t).$$

Additional constraints on flow field smoothness are added via a weighting of neighborhood pixel translations, and coarse to fine multiscale computations are required to capture large scale translations.

3.1.2 Histogram of Oriented Gradients

As described in class, a image patch's HOG descriptor is a high dimensional feature vector, created by concatenating normalized histograms of binned gradients for subpatches within the image patch. Combined with classification tools such as support vector machines, it can be used to detect objects of a particular appearance.

3.1.3 Linear Kalman Filtering

In the most general sense, Kalman filters [5] are effective in dealing with measurement error in updates of an object's physical state through time, by combining knowledge from prior timesteps and data from current measurements to arrive at the most likely state. In our case, the state of the object will be represented as a 4D vector representing 2D position and velocity.

A Gaussian random variable represents the probability distribution of a state at each timestep, and is used to generate a prediction for the next state via a given transformation. Given some measured observation for the next state, we convert that to another Gaussian random variable using empirically estimated/observed measurement error, then multiply the 2 probability distributions together to get the final distribution for the next timestep. The next timestep's state is then set to the peak of that final distribution, which happens to be the mean of the Gaussian.

At each timestep, the object's previous state $s^{(t)}$ is described with a probability distribution in the state space via a mean state vector $\mu^{(t)} = (\mu_p^{(t)}, \mu_v^{(t)})^T$ and a Covariance Matrix $C^{(t)}$. We predict the next state $\mu = (\mu_p, \mu_v)^T$ by a simple linear relationship

$$\begin{aligned} \mu_p &= \mu_p^{(t)} + \mu_v^{(t)} \cdot \Delta t \\ \mu_v &= \mu_v^{(t)} \end{aligned}$$

Or

$$\mu = P\mu^{(t)} \text{ with } P = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

And the estimated Covariance matrix would thus be

$$C = Cov(Ps^{(t)}) + Err_p = P^T C^{(t)} P + Err_p$$

where Err_p is an empirically found distribution smoothing covariance matrix that models potential noise in the predicted distribution.

Now given a measured observation s_m and covariance matrix Err_m corresponding to our belief about measurement noise, we get our final distributions by multiplying the two Gaussian probability density functions, arriving at a new Gaussian distribution with

$$\begin{aligned} \mu^{(t+1)} &= (Err_m(Err_m + C)^{-1}\mu + C(Err_m + C)^{-1}s_m) \\ &= (I - K)\mu + Ks_m \\ C^{(t+1)} &= Err_m(Err_m + C)^{-1}C \\ &= (I - K)C \end{aligned}$$

$K = C(C + Err_m)^{-1}$ is commonly referred to as the *Kalman Gain*, which corresponds to the weight/importance

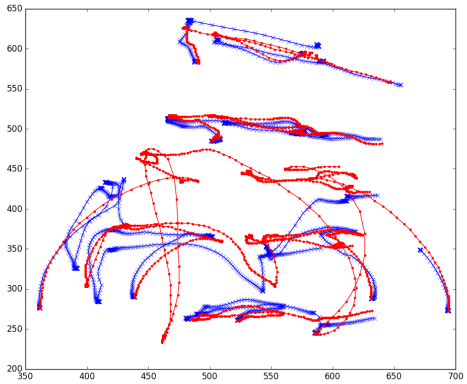

Figure 2. A comparison of joint trajectories via flow-based propagation and ground truth. 400 frames of 9 different joints are plotted, and ground truth is labeled in red, while our trajectories are labeled in blue.

of the measurement relative to the prediction. In practice, the matrices Err_p and Err_m need to be carefully chosen through parameter search or empirical observations, so that K eventually reaches a steady state.

3.2. Methods and Results

3.2.1 Propagation via Optical Flow

The first method we attempted was a direct flow propagation with

$$p_j{}^{(t+1)} = p_j{}^{(t)} + v_j{}^{(t)} \cdot \Delta t$$
$$v_j{}^{(t+1)} = \mathcal{G} * \mathcal{F}^{(t+1)}(\mathcal{N}(p_j^{(t+1)}))$$

where the position p_j of each joint j is estimated directly using the current state's velocity, and the velocity v_j is obtained by convolving a 2D Gaussian Kernel $\mathcal{G}$ scaled to sum to 1 with the optical flow $\mathcal{F}(\mathcal{N}(p_j))$ of the updated joint position's neighborhood. With a larger Gaussian Kernel and corresponding neighborhood, joints are less sensitive to noise but more easily influenced by flow of neighboring pixels, and we empirically found that a neighborhood of 32 pixels works fairly well.

Main issues with this approach was that it performed well only under the condition that no joints cross over. It is ambiguous how flow should propagate joints when one joint crosses over another, and introducing artificial velocity damping does not effectively solve this issue. Furthermore, overly large movements cause errors that propagate through time.

We then revised our approach with intentions of addressing these problems.

Figure 3. Optical flow based propagation does not accurately handle fast hand movements and crossing joints.

3.2.2 Patch Similarity Calculations

The first thing we tried was to adjust and refine the joint positions by comparing the image patch centered at the previous frame's joint position to image patches in the neigborhood of the predicted position.

We started with a simple patch similarity metric

$$\mathcal{G} * abs(I_t(\mathcal{N}(p_j)) - I_{t+1}(\mathcal{N}(\tilde{p}_j)))$$

Which is the sum of absolute pixel differences weighted by a centered gaussian kernel. The weighting is to focus on the joint and reduce difference caused by the translation of the surroundings. However, this algorithm turns out to work less well than directly propagating via optical flow. We think that it is due to fact that extra error is introduced when there are non discrete translations of joints (ex. translations that are not of dicrete pixel units,) which causes drifting when there is small motion in the joints. This error again propagates throughout the timesteps.

Algorithm 1 Pseudocode for image patch differences

```
Initialize image patch and position for first frame, set velocity to 0
for frame do
    for joint do
        predicted position = previous position + previous velocity * timestep size
        Calculate similarities between previous image patch and image patches in predicted positions neighborhood
        Position = center of best fitting image patch
        Velocity = optical flow in the neighborhood of position
        Image patch = cropped subimage centered at current position
    end for
end for
```

3.2.3 Kalman Filtering with HOG

Our next approach was to attempt to resolve ambiguities by tracking each major joint with a Kalman filter, getting our measurements from first predicting where the joint would next be, then searching for the optimal joint position within a neighborhood of the prediction via Support Vector Machines individually trained on HOG features of each joint.

Figure 4. Example of our flow propagated joint tracking method

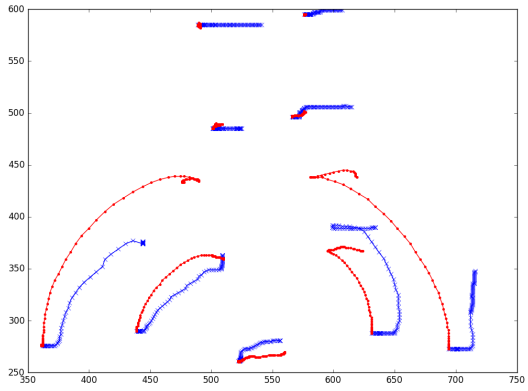

Figure 5. A comparison of joint trajectories via optical flow+image patch matching and ground truth. 100 frames of 9 different joints are plotted, and ground truth is labeled in red, while our trajectories are labeled in blue. A lot of drifting occurs initially when joints are relatively static, which we think is due to ambiguity in small non-discrete motion.

Measured velocity would then be the Gaussian convolved neighborhood optical flow identical to our first approach, and in the case of no HOG detection we would just use the predicted position and update accordingly.

Figure 6. Simple image patch similarity calculations

Algorithm 2 Pseudocode for HOG+Kalman Filtering

```
Initialize 1 Kalman Filter, load 1 SVM per joint
for frame do
    for joint do
        initPos = KalmanFilter.PredictPosition
        Search for positively classified HOG descriptor in initPos.Neighborhood
        if descriptor found then
            measuredPos = descriptor.Position
        else
            measuredPos = initPos and do HOG search in entire image for next frame
        end if
        measuredVelocity = optical flow in the neighborhood of measured position
        KalmanFilter.Update(measuredPos, measuredVelocity)
        finalPos = KalmanFilter.CurrentState
    end for
end for
```

After implementing the pseudocode as described above, we used ground truth joint positions of various videos to crop image patches centered at joints for training our SVM, and pruned occlusions and repeating images out. However, we soon realized that the SVMs trained were too low quality for use, as attempting to train a single SVM on the many

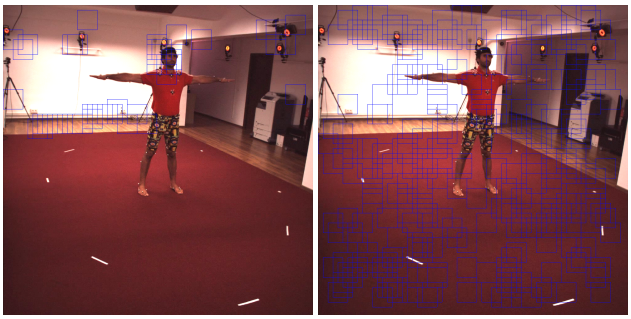

Figure 7. Head detection (left) and foot detection (right)

poses and angles of the corresponding joint resulted in a large number of false positives. This was rather sad.

3.3. Experimental setup

We implemented everything in Python, utilizing OpenCV [1]'s optical flow and video processing functions. We utilized the Human3.6M dataset [4] that includes videos of human movement and ground truth 2D joint positions captured via motion capture.

4. Conclusions and Future Work

Despite less than satisfactory results, this was an intriguing project to work on. Furthermore, through talking to another group working on body pose estimation via an extension of [10], we think combining optical flow with a similar pose-angle aware HOG detector system seems promising, and we'd like to explore similar possiblities in the future.

5. Acknowledgements

We'd like to thank the teaching staff for their help and guidance this past quarter, and to particularly thank Professor Savarese for his suggestions on existing literature, and Kevin Chen for his advice on Neural Networks.

References

[1] G. Bradski. *Dr. Dobb's Journal of Software Tools*.

[2] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image analysis*, pages 363–370. Springer, 2003.

[3] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015.

[4] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[5] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[6] J. Romero, M. Loper, and M. J. Black. Flowcap: 2d human pose from optical flow. In *German Conference on Pattern Recognition (GCPR)*, 2015.

[7] H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Computer VisionECCV 2000*, pages 702–718. Springer, 2000.

[8] D. Teney and M. Hebert. Learning to extract motion from videos in convolutional neural networks. *CoRR*, abs/1601.07532, 2016.

[9] Y. Yacob and L. Davis. Learned temporal models of image motion. In *Computer Vision, 1998. Sixth International Conference on*, pages 446–453. IEEE, 1998.

[10] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1385–1392. IEEE, 2011.