CS5340 Final Project Report

Yao Yujian **Pallav Shinghal**

Abstract

This paper presents a method to determine the joint-level pose of a single hand in a continuous set of depth images ("depth video"). Our work builds upon previous research on single-image pose estimation and uses local temporal information to reduce estimation error, random artifacts and motion jitter.

1 Prior work

Significant advances have recently been made in the estimation of 3-dimensional co-ordinates of human hands/bodies using input from depth cameras such as the Microsoft Kinect and the SoftKinetic DepthSense ([4], [5] and [7]). These works combine a variety of statistical methods to predict the pose of a hand from the 2-dimensional depth value projection of a single 3-dimensional point cloud. Although these works produce pose estimations at a significant frame rate (12 to 25 frames per second), they are unable to take advantage of correlations between hand positions captured in a short interval. As such, in the estimation of a continuous series of depth images, they accumulate independent error and are unable to correctly assess the hand pose in some self-occlusive positions. This leads to significant "jitter" in the series of pose estimations.

In this work, we build upon the single-frame pose estimation algorithm of [7], which is as follows. The hand pose estimation is done in three main steps:

1. A Hough forest [1] is constructed, with a number of decision trees that combine input from a number of pixels to predict the lateral position of the base of the hand, as well as the orientation of the whole hand. This is used to determine the appropriate rotational and lateral transformations to map the existing depth image to a canonicalized form of the depth image of the hand.
2. The rotational mapping produced in step 1 is applied over the per-pixel depth-invariant input features [5] of a second Hough forest to create a number of 3D hand pose predictions.
3. The candidate predictions generated by step 2 are passed through an "estimation by synthesis" [6] method that minimizes the distance between a synthetic image generated from the predicted model and the actual depth image provided.

On average, this method provides good single-frame pose estimations from the depth input. However, it suffers from all the problems discussed above: independent error; incorrect assessment in self-occlusive poses; and significant jitter in a continuous series of estimations. In addition, this method occasionally produces a "flipped" prediction of the hand orientation, likely due to the high correlation between the synthetic images of hands in reverse orientations, as shown in Figure 1.

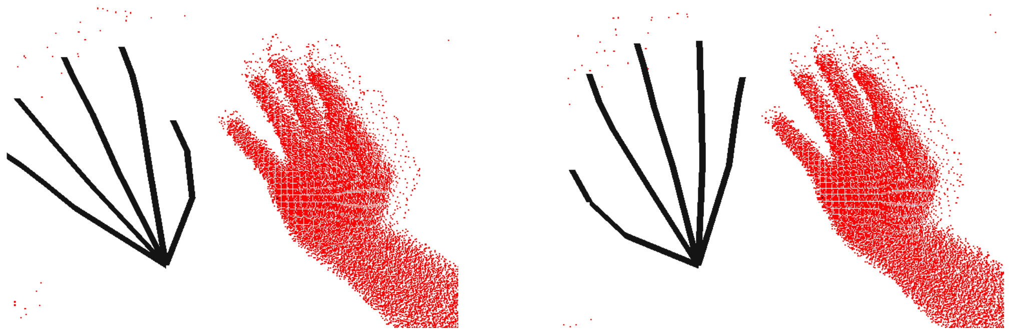


Figure 1: Black: the output of hand engine, red: the depth image. In the first frame the fingers are correctly classified, but in the next frame the fingers are mis-classified. This can be improved by exploiting the temporal correlation of two frames.

In the next section, we present a few different methods to exploit local temporal correlations to produce smoother and more accurate pose estimation videos.

2 Algorithm

We build upon the "hand engine" from [7] by piping its output into another filter, consisting of a smoothing filter and a correction algorithm to fix mis-classification of fingers, as illustrated in Figure 2.

This section also presents an attempt at using a Kalman Filter for smoothing.

2.1 Smoothing

2.1.1 Kalman Filter

We created a Kalman Filter [2] with the following properties:

State

- Coordinates and *velocity* of every point.
- *velocity* at frame i is the displacement vector from the coordinates of the previous frame to those of the current frame.

Transition

- Add the *velocity* vector to the coordinates vector of the current frame.

This method was found to create excessive "smoothing". It was experimentally observed not to follow the hand action in certain hand movements (eg. a hand "clamping" action).

As a solution to the problem of over-smoothing, the *velocity* of all joints except the base of the hand was changed from a linear velocity to *angular velocity*.

The *angular velocity* is represented as the rotation matrix that transforms the vector representing the joint-edges (corresponding to the fingerbones) in the previous frame to the ones in the current frame.

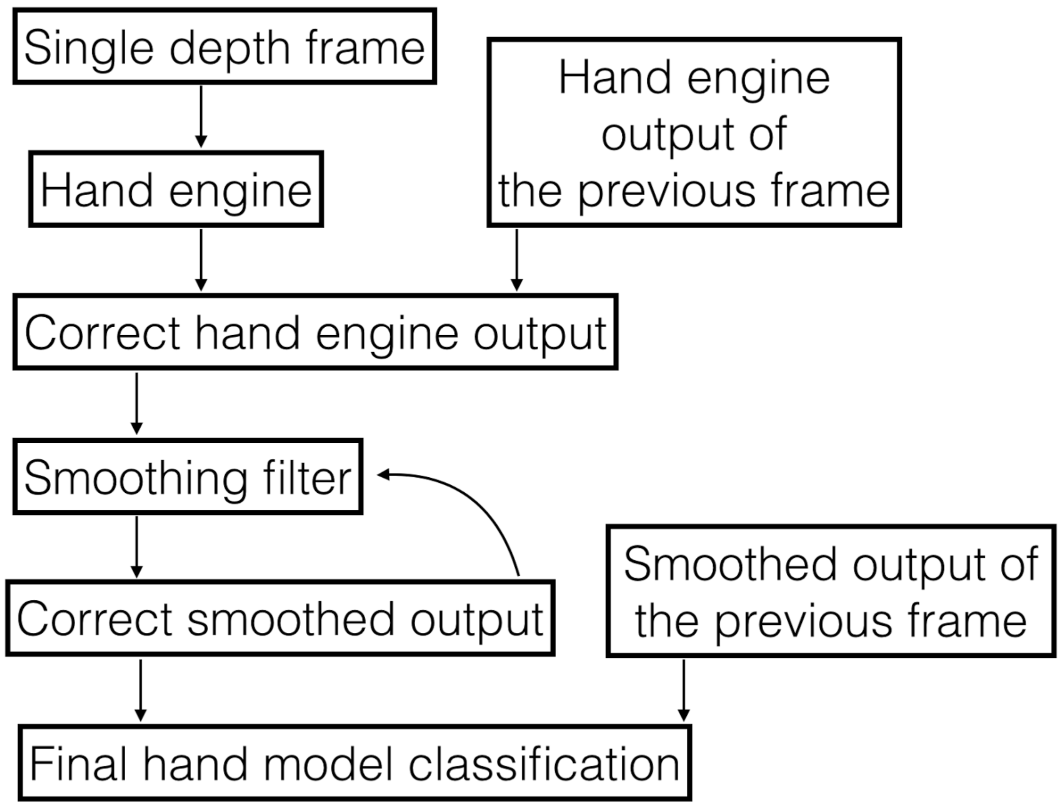

Single depth frame
Hand engine
Hand engine output of the previous frame
Correct hand engine output
Smoothing filter
Correct smoothed output
Smoothed output of the previous frame
Final hand model classification

Figure 2: Overview of the proposed algorithm

To make the angular velocity-based transition function linear, we applied the the angular velocity on the current frame to get a prediction for the next frame and then used the displacement from the current frame to the predicted frame as *velocity*. This solves the problems mentioned above, but the smoothness is still not as good as the PD motion-smoothing filter in the following section.

2.1.2 PD Motion-Smoothing Filter

To facilitate better smoothing, we created another filter inspired by closed-loop proportional-derivative controllers [3].

Using first-order frame distance ("velocity") and second-order frame distance ("acceleration"), the filter produces a smoothed output frame, trying to resist excessive acceleration, using the following equations:

$$x_t = x_{t-1} + (K_v \cdot v_t) + (K_a \cdot a_t)$$

$$v_t = z_t x_{t-1}$$

$$a_t = v_t v_{t-1}$$

Here, z_t is the set of joint coordinates observed at time t, x_t is the "smoothed" output frame for time t, v_t is the first-order frame-displacement at time t and a_t is the second-order frame-displacement at time t. K_v and K_a are constants, presently determined through a manual process.

The filter produces jitter-free output, but in cases where a the hand engine produces a "flipped" hand, the smoothed version results in a "squeezed" hand, due to the drastic change in coordinates (high velocity) of the frame transition, as illustrated in Figure 3.

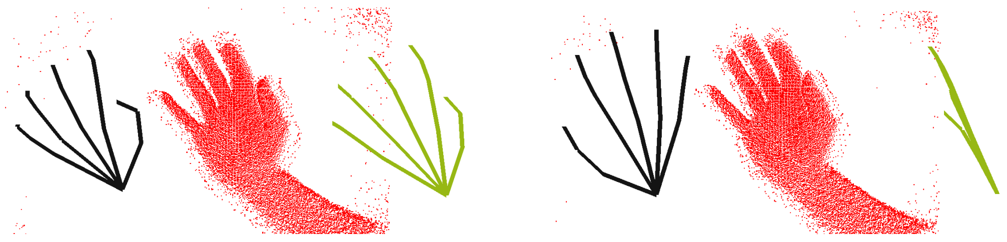


Figure 3: Black: the hand model predicted by the hand engine. Red: the depth image. Green: the smoothed hand model. Due to the misclassification of fingers, the smoothing algorithm attempts to even out the joint positions in between frames, resulting in the squeezed hand.

2.1.3 Fixing mis-classification of fingers

We created a "flip"-detection method based on the heuristic that metacarpophalangeal joints cannot be too close together in a regular human hand (Figure 4). The algorithm is as follows.

1. Compute the maximum pairwise distance D_i between the metacarpophalangeal joints in frame i.
2. If the D_i is smaller than half of D_{max} - the maximum distance observed, consider the model invalid.

3. If the model is invalid, correct the input to the smoothing filter. Get the mid-points between the joints in the previous frame $i-1$ and the joints in the current frame i. If the hand model so constructed is invalid (by the definition given in step 2), discard the current frame and use the previous frame as input to the smoothing filter.
4. If the output of the smoothing filter is invalid, discard it and use the smoothed output for the previous frame instead.

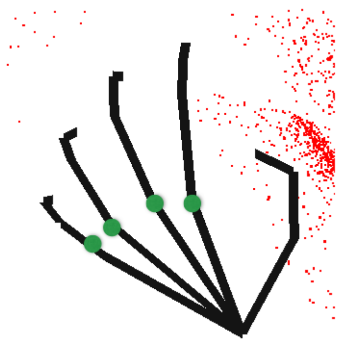
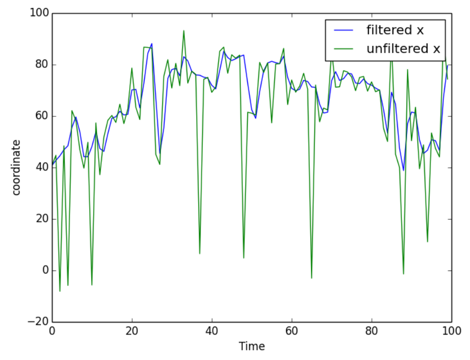


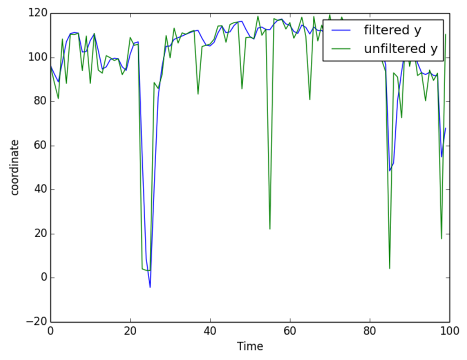
Figure 4: metacarpophalangeal joints (highlighted in green) are usually far apart, and it is assumed impossible for a human to squeeze them together

3 Illustration of smoothing

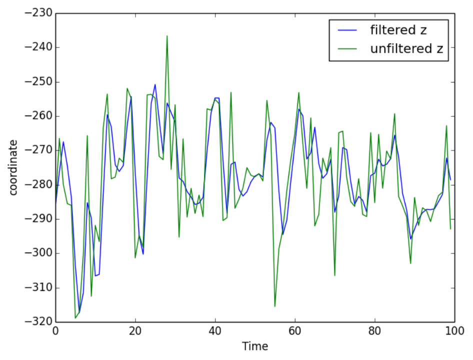
As seen in Figure 5 and 6, the trajectory of the fingertip is considerably smoother than the original output, and most instances of mis-classifications of fingers (manifested in sudden changes in position) are corrected.



(a) x coordinate of the tip of the index finger

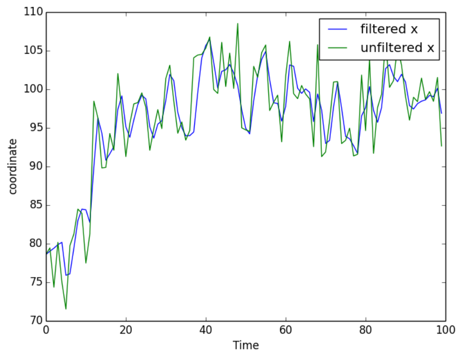


(b) y coordinate of the tip of the index finger

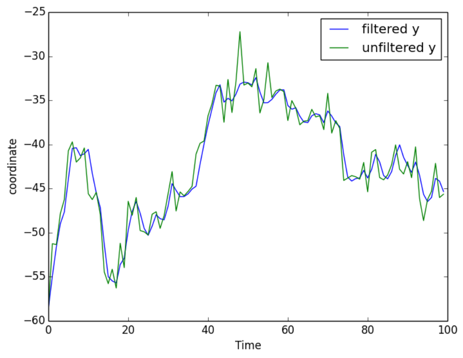


(c) z coordinate of the tip of the index finger

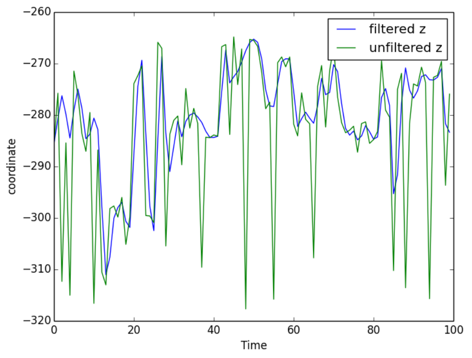
Figure 5: Coordinates of the tip of the index finger over time



(a) x coordinate of the base of hand



(b) y coordinate of the base of hand



(c) z coordinate of the base of hand

Figure 6: Coordinates of the base of the hand

4 Illustration of final output model

Figure 7 and 8 shows the comparison between the original hand pose estimation algorithm and our improved version.

5 Future works

The current method uses only one predicted outcome from the original hand engine, the result of "estimation by synthesis". The other predictions generated by the Hough forest are discarded in the process, leading to a less rich source of data for our algorithm.

We may instead be able to exploit the prediction candidates generated in step 2 of the hand engine to find the smoothest and most accurate joint trajectory.

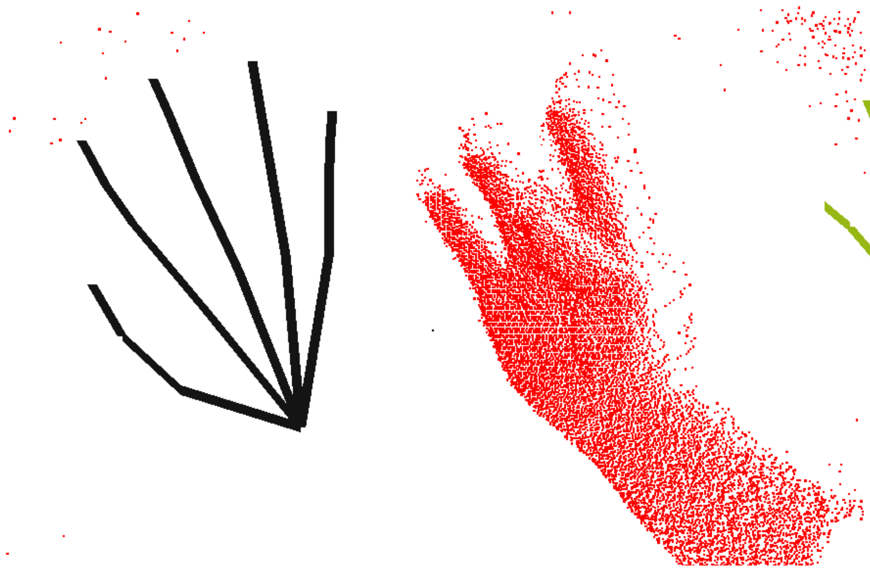


Figure 7: Red: depth image. Black: the classifier fails to classify the hand model, including the hand orientation (the thumb was identified to be on the left when in fact it is on the right). Green: by exploiting temporal information, we correctly identify the current hand gesture.

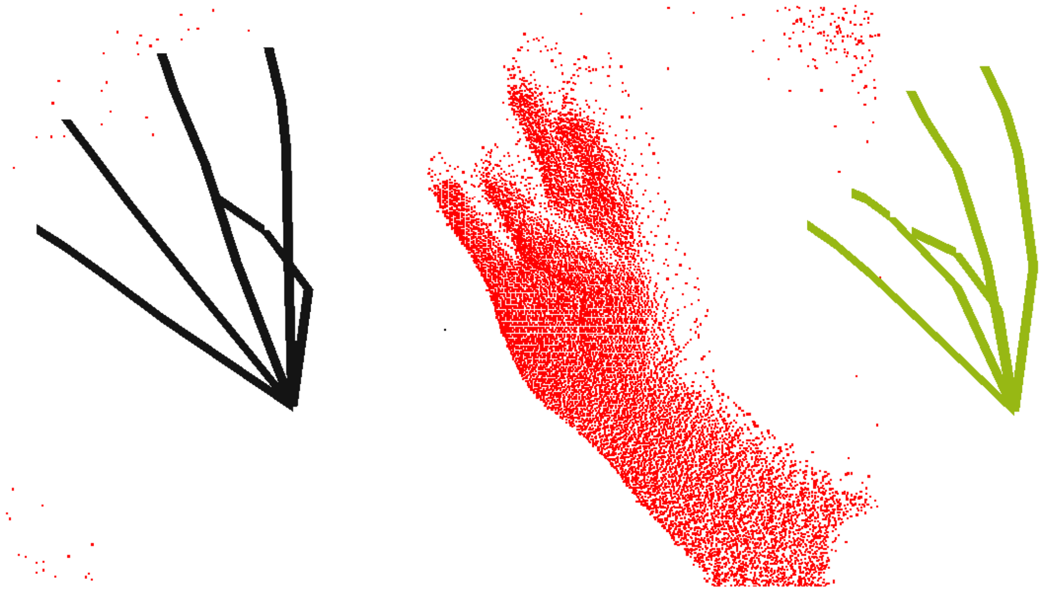


Figure 8: Red: depth image. Black: the classifier fails to classify the ring finger model. Green: by exploiting temporal information, we correctly identify the ring finger model.

6 References

[1] Gall, Juergen, and Victor Lempitsky. "Class-specific hough forests for object detection." Decision Forests for Computer Vision and Medical Image Analysis. Springer London, 2013. 143-157.

[2] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. Journal of Fluids Engineering, 82(1), 35-45.

[3] Minorsky., N. (1922), DIRECTIONAL STABILITY OF AUTOMATICALLY STEERED BODIES. Journal of the American Society for Naval Engineers, 34: 280–309. doi: 10.1111/j.1559-3584.1922.tb04958.x

[4] Qian, C., Sun, X., Wei, Y., Tang, X., & Sun, J. Realtime and Robust Hand Tracking from Depth.

[5] Shotton, Jamie, et al. "Real-time human pose recognition in parts from single depth images." Communications of the ACM 56.1 (2013): 116-124.

[6] U. Grenander. Pattern Theory: From Representation to Inference. Oxford University Press, 2007

[7] Xu, Chi, and Li Cheng. "Efficient Hand Pose Estimation from a Single Depth Image." *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013.