Using a colour Threshold-prediction algorithm to track motion

October 21, 2018

Name	SID
Martin Brandel	450116047
Option	1

Contents

1 Introduction

Computer vision is a rapidly growing area of research, with untold opportunity and rapidly growing use-cases. One of the ways computers can visualize objects is through object tracking using colour thresholds. This report details the process of using colour threshold detection to track the movement of body parts on a monkey animation and the processing markers to predict the locations of the joints when they are not all visible.

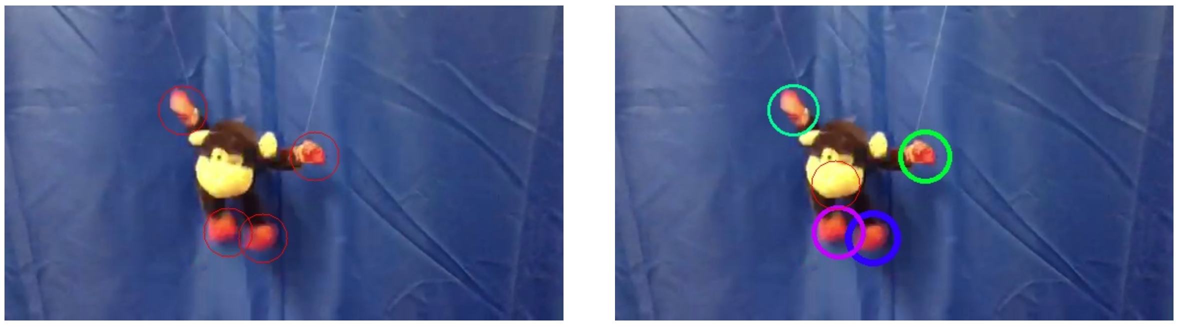

(a) Colour recognition (b) Object recognition

Figure 1: Example prediction position recognition

2 Implementation

2.1 Colour Recognition

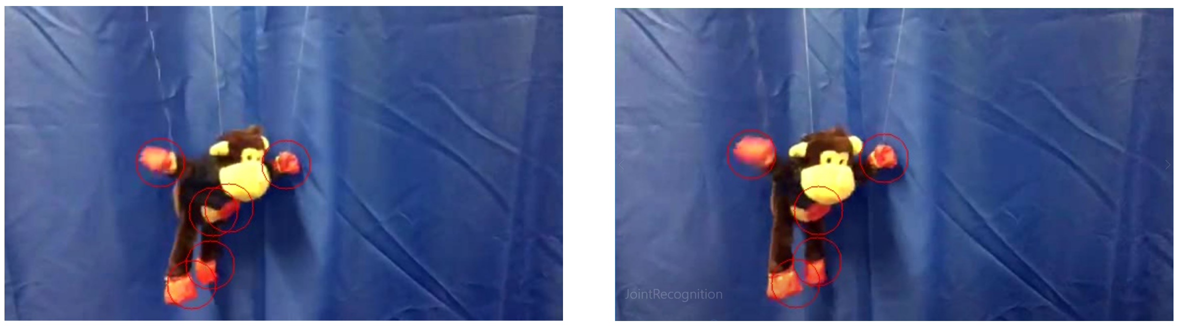
The first step required is to have the computer recognize the red areas on the Monkey. This is done by moving through each pixel in the frame and only performing actions on pixels with a RGB value above a certain threshold. Recognizing a wrong area as an object is more difficult to process than if we assume all the recognized areas are a joint, so setting the threshold high enough not to catch other sections is critical.

Following this, the first passing pixel is saved into a python list within another list. Each following pixel within a certain range of the first is saved into that corresponding list. If it is further away from the original pixel than our checking range, we save it into a new list in the next index of the parent list. Subsequent pixels are checked between the two and so on. This groups the pixels into "zones" which we can assume are our red markers. The center of all the pixels are then calculated and the location of the marker is recorded.

This method is not able to differentiate between two markers which are on top of each other, however this can be processed after in the body part recognition step. This algorithm also differentiates when two markers are very close together or barely touching. This is because the total length of the two markers would be larger than the total check range for the separate zones.

2.2 Colour Recognition Result

For this video a R value of over 150 and G value of under 70 or and R value of over 230 and G under 110 was used. This covers the vast majority of cases where a red marker on the monkey is visible, including if it is shaded or in the light. The grouping range was found to be ideal at 50 pixels, that being the maximum size a glove or marker becomes during the video.



(a) Threshold of second R = 175 (b) Threshold of second R = 230

Figure 2: Example prediction position recognition

2.3 Body Recognition

To match the recorded markers with the correct body part, further processing was required. This is because colour cannot be used to differentiate between each joint. To recognize which joint is corresponding with which marker,their relative position was analyzed and the markers were matched to each joint. In the case where there are less or more than 5 markers, a predicted position of the markers calculated from their previous positions are is used, snapping each joint to a marker if it is located close to it.

2.4 Body Recognition Result

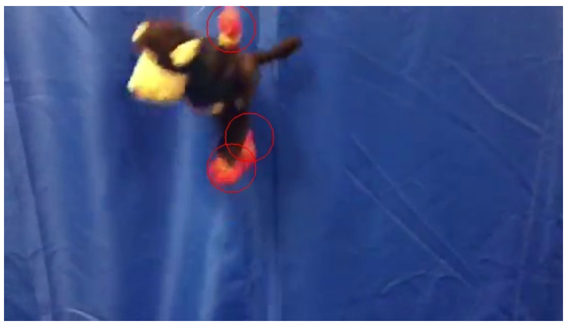
The conditions used to recognize the joints were as follows: First, the two lowest markers are designated as either foot, left and right separated by their relative x position. This leaves the remaining 3 markers as the hands and body, which is then split among them in order of their x position. This covers all the cases correctly. In our figures, two shades of green are the two hands, blue and purple the feet, and red the center.

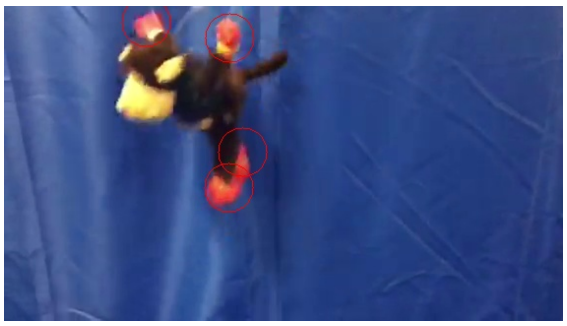
2.5 Body Prediction Result

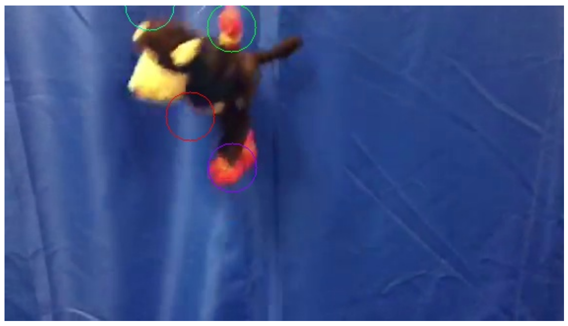
In the case where there are more or less than 5 markers, it means one or more of them is missing or there is an extra one. Since there is no easy way of knowing which marker is missing, we approached this with a predictive method like follows:

- Find the center of the markers.
- Find the distance traveled from the previous frame.
- If the travel distance is too high, set it to 0.
- Predict the joint positions by adding the movement of the center of all the joints.
- Move through each prediction: If it is within snapping distance, snap it to the closest marker.

The maximum travel distance needs to be implemented, because in the case where a marker on the edge of the puppet disappears, the center will look like it has traveled a large distance in the wrong direction as seen in the bottom figures. If the maximum travel distance is too high it will do the same thing. We found a maximum travel distance of 30 pixels to be sufficient.


(a) Marker recognition frame 944


(b) Marker recognition frame 945


(c) Joint recognition frame 944

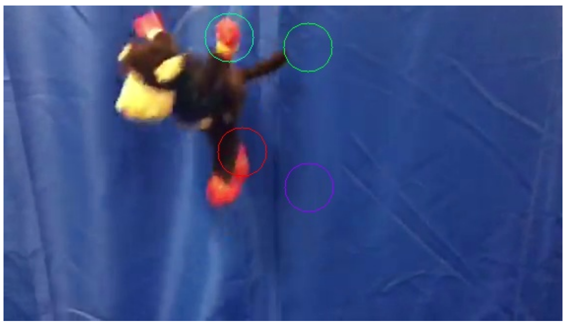
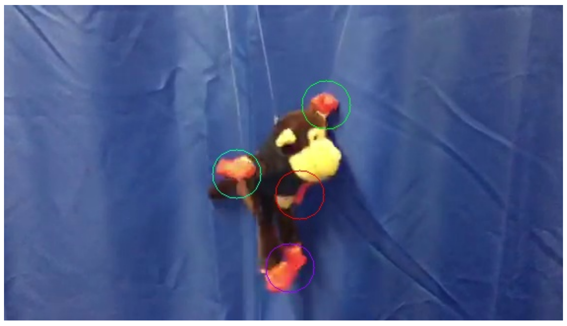
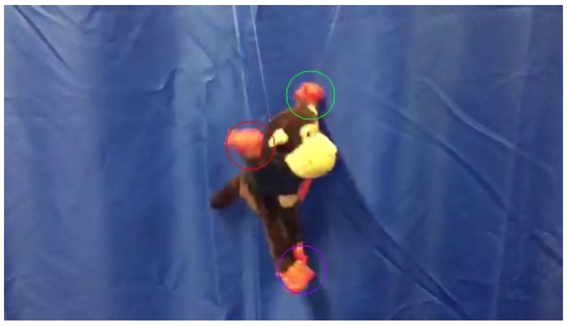

(d) Joint recognition frame 945

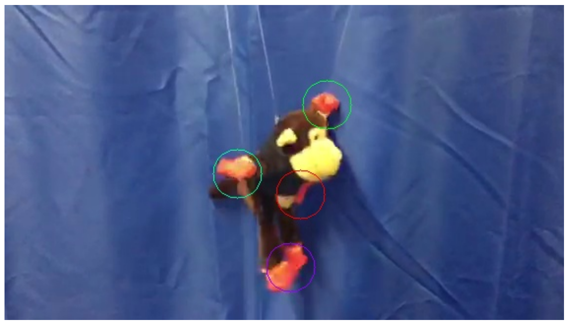
Figure 3: Too low maximum distance restriction

Secondly, the snapping distance was also important to calibrate. If it was too high, it would snap to markers when the prediction would have been a much better estimate, while if it was too low, it would drift far from the joint, when a more accurate position is available.

We found a snapping distance of 40 to be ideal for this video. Below are example of problems with incorrect snapping distances.


(a) Too high snapping distance frame 37


(b) Too high snapping distance frame 39


(c) Too low snapping distance frame 37

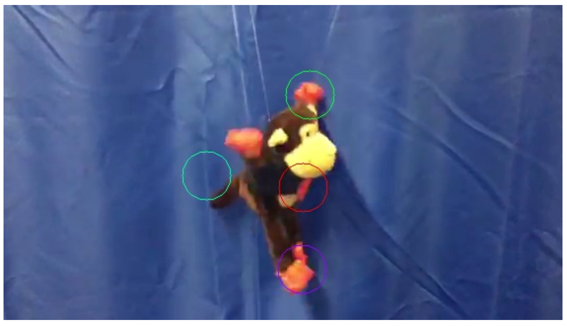
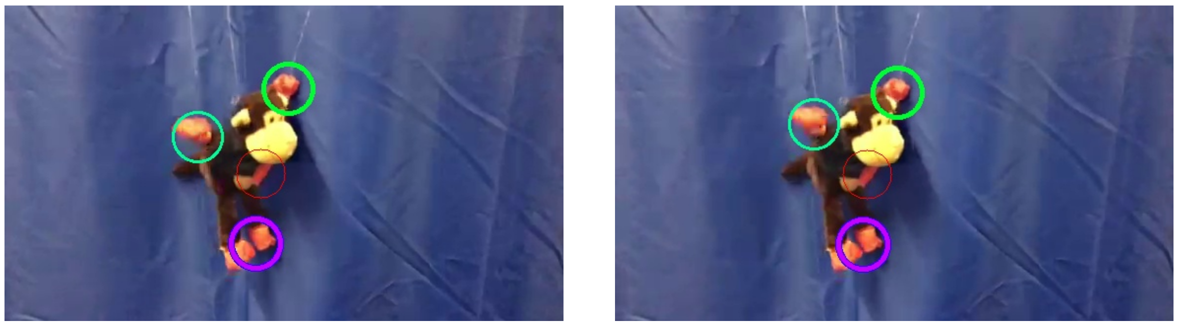

(d) Too low snapping distance frame 39

Figure 4: Incorrect snapping distances

3 Results

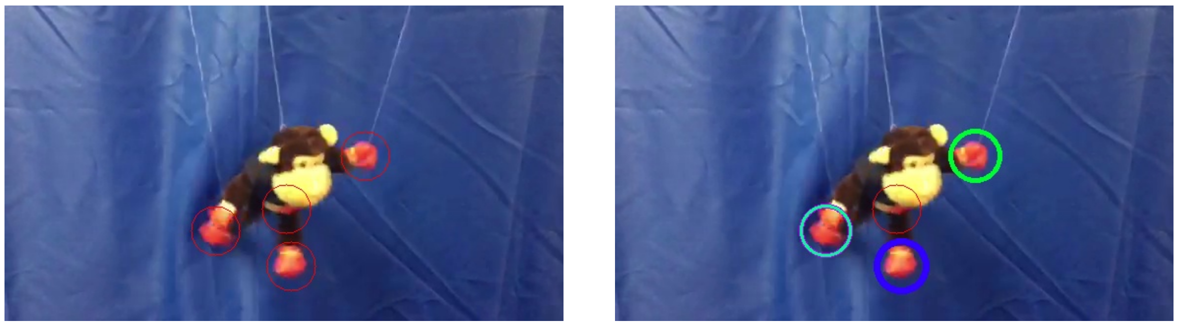
The final version gives us a comprehensive solution which tracks the puppet 99% of the time accurately. The figures below demonstrate some edge cases that this algorithm has dealt with well and badly.



(a) Frame 4 joint detection (b) Frame 5 joint detection

Figure 5: Feet collide case

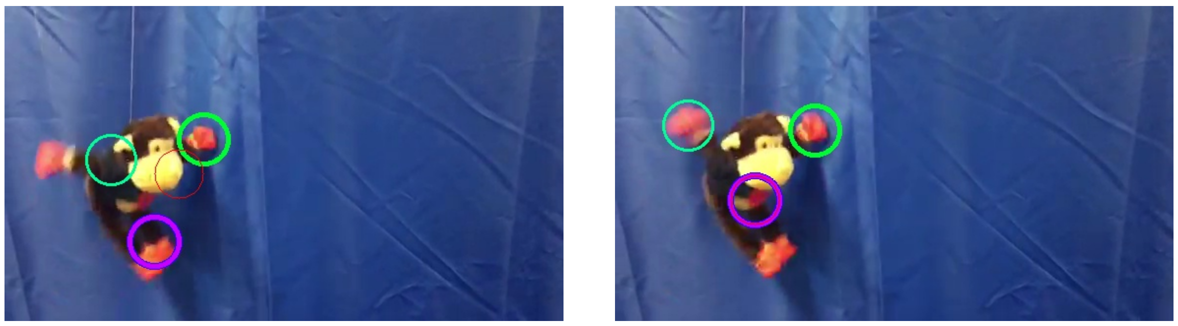
As seen in the figure above, in the case where the feet are together and the model is moving slowly, the algorithm tracks movement perfectly. The feet are joined but all trackers follow their local markers. This can also be seen in other scenes with certain body parts touching as seem below.



(a) Frame 388 marker detection (b) Frame 388 joint detection

Figure 6: Hands collide case

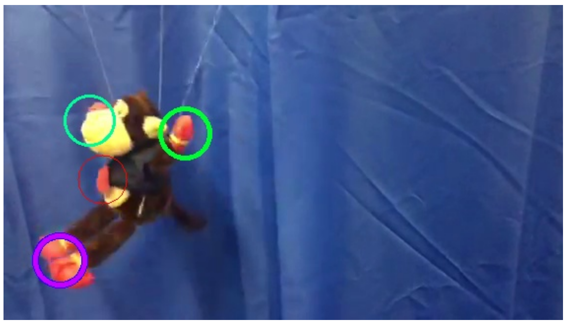
There are other cases where even this algorithm still has problems with. One of those cases is if another marker is closer to the predicted case than the correct one. A joint may then snap to a wrong marker. This could be solved with more extensive positional checks on marker snapping.



(a) Frame 767 joint detection (b) Frame 767 joint detection

Figure 7: Wrong marker snap case

The last problem case is joint marker drift. If a predicted case falls outside of where the markers are, it can continue moving in that direction until it is snapped back due to all 5 markers being seen. This is seen in one moment in the video as seen below.


(a) Frame 794 joint detection

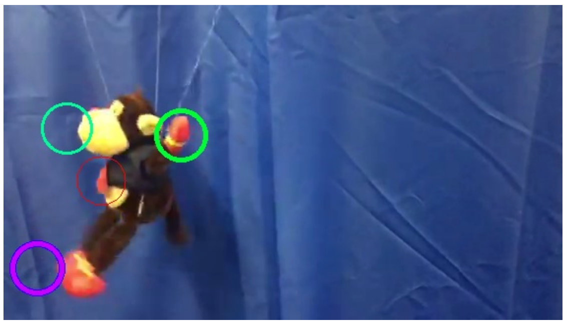
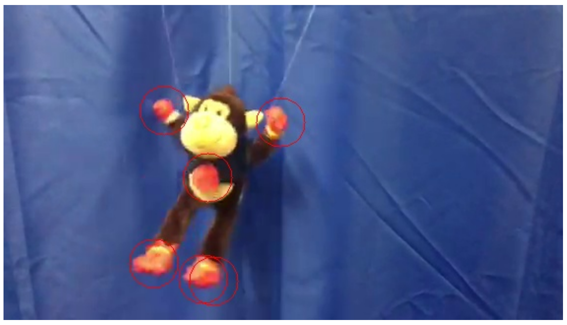

(b) Frame 795 joint detection

Figure 8: Marker float case

The last example to give is the more than 5 marker case. It handles this well as seen bellow.


(a) Frame 505 marker detection

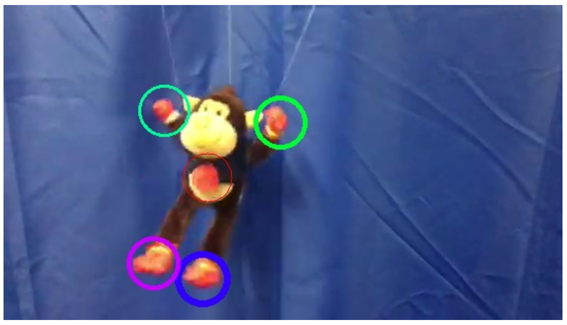

(b) Frame 505 joint detection

Figure 9: More than 5 markers case

4 Animation

The output animation video is a game where you have to shoot down the flying space monster with your ship. The monster follows the movements of the monkey puppet exactly as recorded by the above algorithm with its heads. The different facing heads represent the arms and legs while the larger one is the body. A screen-cap can be seen below.

Bullets are reflected when they collide with the smaller heads, while they damage the monster if they hit the larger head. As the monster takes damage the health snake on the top of the screen becomes shorter. This same health snake also snakes up and down in accordance with the y change of the monster.

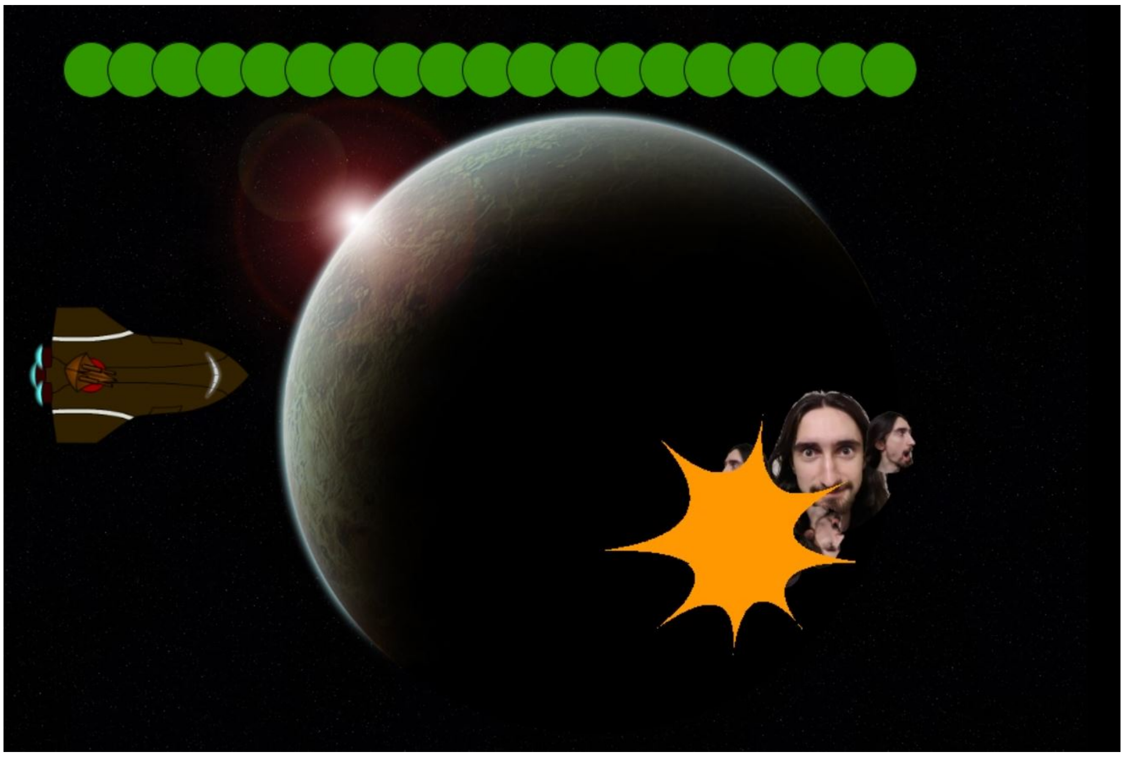


Figure 10: Space Monster game utilizing the monkey animation

5 Conclusion

In conclusion, the prediction-threshold algorithm works substantially well in detecting the movement of the monkey puppet, and can translate it to animation fairly successfully. The optimum constants for an accurate translation was found to be 40 pixels for the snap distance, 30 pixels for the maximum movement distance and two separate colour filters of R¿150, G¡70 and R¿230,G¡110.

6 References

No outside sources were referenced for facts or methods used in this report other than materials and education received in COMP3419. A course at the University of Sydney (2018)