2012-36

Kinect Hand Recognition and Tracking

Authors: Alex Drake

Abstract: The goal of this research project was to be able to identify and track a hand using the depth image from a Microsoft Kinect. The ability to do this would have uses in sign language recognition, rehabilitation, and gesture recognition amongst others. The Microsoft Kinect is currently capable of identifying and tracking whole bodies. Our approach was to follow the method that Microsoft used but apply it to detailed hand recognition instead of the body. The basic strategy they used was to take a large amount of labeled depth data and build a decision tree to identify which part of the body each pixel of an image belongs to.

Type of Report: MS Project Report

Kinect Hand Recognition and Tracking

Alex Drake

Introduction

The goal of this research project was to be able to identify and track a hand using the depth image from a Microsoft Kinect. The ability to do this would have uses in sign language recognition, rehabilitation, and gesture recognition amongst others. The Microsoft Kinect is currently capable of identifying and tracking whole bodies. Our approach was to follow the method that Microsoft used but apply it to detailed hand recognition instead of the body. The basic strategy they used was to take a large amount of labeled depth data and build a decision tree to identify which part of the body each pixel of an image belongs to.

Microsoft's Research

The Microsoft researchers had motion capture and 3D modeling at their disposal. Using these tools they were able to create a large amount of varied data which they labeled.

[1]

Using this data they trained a decision tree to determine what label a pixel should have. The features they used to determine build their decision tree consisted of two vectors centered at the pixel in question. To calculate the feature value for each pixel they measure the difference in depth between the two vector points. For a pixel, *x*, and two vectors, *u* & *v*, the feature value is calculated as:

$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$ [1]

The depth values are normalized so that they are depth invariant, by dividing *u* and *v* by d_I(x). This normalization guarantees that features won't be dependent on how close to the camera you are.

The decision tree is built by looking at a large number of different features and at each node of the decision tree, splitting using a feature which most cleanly divides the training pixels. With an understanding of Microsoft's approach to body recognition we set out to apply their method to hand recognition.

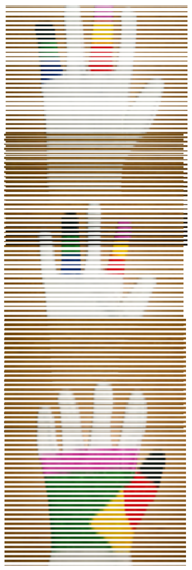
Data Collection

To implement this approach we needed a lot of data. However, we didn't have any motion capture or 3D models at our disposal. The first task we had was to find a way to engineer a method we could implement to automatically label hand regions.

The solution we came up with was to use the fact that the Kinect has an RGB camera to physically label hands with colors. If we could color-code gloves and match the RGB camera image with the depth camera image then we would have similar data to labeled motion capture or 3D models.

Our first attempt at creating gloves was to use tie-die to color the different segments of the glove. However, this method proved unacceptable. The colors faded and bled together and each glove took hours to make. Our second approach was to use paint markers. The results were much better. The creation of the gloves took much less time and the colors were brighter and didn't bleed together.

Once we had found a method for creating the gloves we had to choose a set of colors to use. The best set of colors would be one for which the smallest distance between any two colors was as far apart as possible. We had 16 different paint colors to choose from. Labeling every section of a glove with a different paint color would require the use of colors which are too similar to each other to dependably distinguish. Instead of using one glove, we decided to use three different gloves. If we keep track of which glove a pixel comes from in addition to its color then we can distinguish the red on one glove from the red on another and create a larger number of labels with less colors.

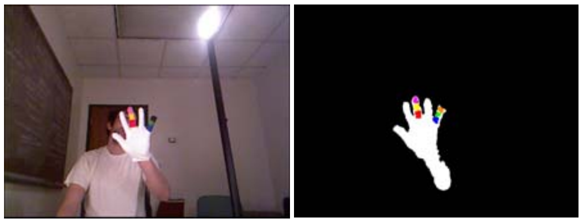


We set up a small lab to collect our data. The lab consisted of a small room with no windows. With the door closed no outside light got in. This allowed us to control the lighting. We used a series of incandescent light bulbs to illuminate the room. Incandescent bulbs offer a broad spectrum of light wavelengths which was ideal for our purposes.

We had volunteers come to our lab and run through a series of different hand poses using each of the gloves. In total we collected about 5,000 images.

Labeling and Cleaning the Data

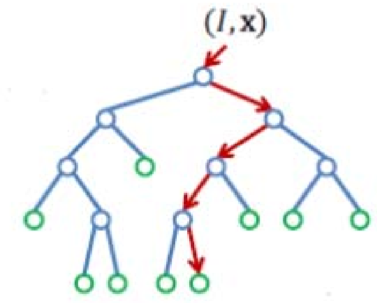
With our images in hand we next had to clean the images and label them based on color before we could use them to build a decision tree. First we applied a depth threshold to each image to isolate the hand in each image. Then we took sample points of each paint color in the image and labeled pixels based on which of the sample colors they were closest to. As a final step we applied an opening algorithm on the pixels. This has the effect of eroding pixels from the edge of colored regions then expanding out evenly from those same areas. The result of running this algorithm is to remove isolated, mislabeled pixels. The final result were images that were ready to be used by our learning algorithm.



Building the Decision Tree

The decision tree is built by reading a large number of sample pixels. For each training pixel we calculate the feature value for a large number of features. Once all of the training pixels and features have been mapped we recursively build the decision tree by determining which feature and threshold divides the pixels into the purest groups. This recursive tree building continues until the pixels at a node have a certain amount of purity (same label) or a maximum tree depth is reached. When either of these two criteria is reached the node is considered a leaf node and given the most prevelant label amongst its pixels.

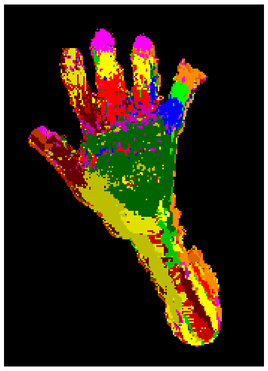
When the decision tree is completed it can be used to rapidly label pixels in an image by traversing the decision tree. Each pixel starts at the root node of the tree and calculates its feature value for the feature at that node. Depending on which side of the threshold its value is, it continues down the tree until it reaches a leaf node. When it reaches a leaf node it labels itself with that leaf's label. This process is extremely fast and we are able to label images at 30+ frames per second.



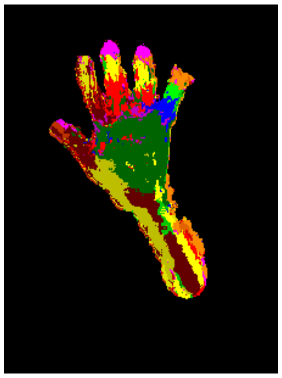
[1]

Implementations

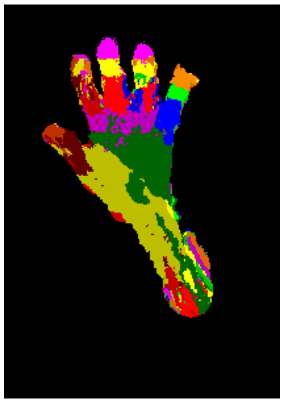
The first decision tree we built had a depth limit of 20, a minimum leaf purity of 93%, and small feature vectors limits. The result of this tree was mixed. It labeled the pinky well and parts of other fingers but was somewhat inaccurate and generally noisy.



We thought that some of the issues in our tree might be due to overfitting to our training data. We attempted to reduce this possibility by lowering our depth limit from 20 to 11 and reducing our minimum leaf purity to 80%. The result of doing this was a slight improvement, primarily in noisyness.

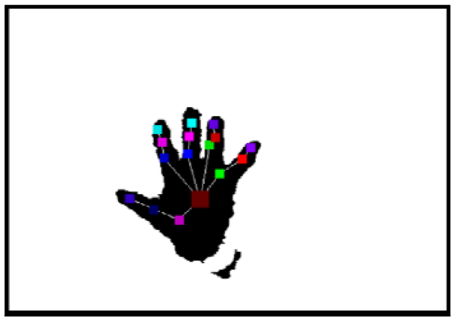


We were still dissatisfied with the results of our decision tree. The next improvement we attempted was to increase the feature vector range. This had a substantial impact on our results. Noise was even further reduced and the overall accuracy was better.



Building a Hand Skeleton

Once we had an acceptably accurate decision tree we attempted to construct a skeleton based on the labels we created. Our initial thought was to use centroids to find the center of each part of the hand. However this method would be very sensitive to noise. Even a small mislabled area could pull the center of a region far from its appropriate location. Instead we decided to implement a mean-shift algorithm which is the same method that the Microsoft Researchers used to build the whole body skeleton. We found that our skeleton labeling works fairly well. It is sensitive to situation where our decision tree mislabels an area and is therefore only as accurate as our labeling is.



Discussion

We were able to follow the technique used by Microsoft to do whole body tracking and apply this to detailed hand tracking. Going forward with this reasearch we would like to find a better method for data capture such as motion capture or 3D models. We would also like to increase the scale of our implementation. We worked with many fewer images, poses, and people than the Kinect researchers did. Even with these areas for improvement we were able to produce a fairly accurate hand skeleton tracker. We feel that this research demonstrates that the method of labeling and tracking used can effectively applied to human hands.

Acknowledgements

This project was implemented in close collaboration with my colleague Jed Jackoway. Our faculty research project advisor was Robert Pless. We were also advised by PhD graduate student Austin Abrams.

References

[1] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., et al. (2011). Real-Time Human Pose Recognition in Parts from Single Depth Images. (http://research.microsoft.com/pubs/145347/BodyPartRecognition.pdf)