LaTeX Detecting Hand Direction for Computer Interaction

Ben Mattinson and David Gaddy
Massachusetts Institute of Technology
77 Massachusetts Ave. Cambridge, MA 02139
bmatt@mit.edu, dgaddy@mit.edu

Abstract

In this paper, we present a system for determining

This paper examines previous work in hand detection and gesture recognition and proposes a framework for an application that allows users to interact with a computer by pointing at the screen. Previous work on gesture detection has primarily been focused on detecting hands and estimating hand poses. Several features of hands have been used in detection including hand color, shape, or depth information. To estimate hand poses, many have focused on recognizing a limited set of hand gestures by identifying finger positions or training classifiers. More recently, work has been done on estimating continuous hand poses instead of a discrete set of hand gestures using convolution neural networks. In our proposal, we outline a human-computer interaction system for controlling a computer by pointing on screen. This approach differs from many of those previously taken in that it focuses on detecting the angle of a single finger in relation to the screen rather than on estimating hand poses or recognizing gestures.

1. Introduction

Gesture recognition is an important problem for Human Computer Interaction because hands provide a very natural way to interact with the computer. Users are used to using their hands to manipulate the world and often use their hands when communicating ideas to other people. Giving computers the ability to recognize gestures would allow people to manipulate objects on a computer in a way that mimics the way they manipulate the world and communicate with others, instead of forcing them to use an artificial form of manipulation like a computer mouse. There are also other related applications for this technology, such as the interpretation of sign language. The task of gesture recognition is often divided into two parts: first detecting hands in an image, and second recognizing the gesture a hand is making. Detecting the hands deals with finding local patches of pixels where a hand appears in an image. In order to do this, the computer must be able to differentiate hands from other objects in the background and from other parts of the body. Recognizing hands differs from many object recognition tasks, such as face detection, because the hands have many degrees of freedom that allow them to appear very different in images based on the gesture being made. For this reason, hand detection is often done by modeling skin color, but it has also been done by classifying shapes and combinations of the two methods. Hand gesture recognition deals with the problem of, given a patch of pixels that represents a hand, determining the configuration the hand is in. One approach to this problem is modeling the joints of the hand and using image data to determine the position of each joint, but because of the complexity of this model, an example based approach is often used instead. In the example based approach, hand images are compared against examples using various features to find an example with a similar hand configuration. This approach is similar to the problem of object classification since different configurations of a hand look like different objects when projected onto a two dimensional image.

2. Previous Work

2.1. Hand detection in images

Because the shape of hands varies greatly based on the position and configuration of a hand, color is often used as a primary feature for detecting hands in images. In [9], Bergh et al. modeled skin colors with a Gaussian Mixture Model of skin color and combined this with a color histogram of faces found with a face detector. By combining both the general skin color model with a scene specific skin color, they were able to detect skin accurately. To distinguish hands from other skin, they took the largest connected regions of skin that were not the face (determined using the face detector). Their method was reported to be more robust to lighting and different users than more basic color models, though they still encountered problems when faces and hands overlapped in images. Other methods for detecting hands involve looking at shape. Ong and Bowden

[6] trained a tree of boosted classifiers to find hands in images. To deal with the many different possible shapes of the hands, the first level of the tree proposed possible hand locations that were then put through classifiers lower in the tree that were trained to detect specific hand configurations. They reported a 99.8% success rate on hand detection. One last cue used for detecting hands is depth information. To deal with the problems of detecting hands with color alone (namely overlapping skin regions), Bergh and Gool used depth images taken with an infrared depth camera in addition to color in [10]. By placing the restriction that the hands be held a certain threshold in front of the face (determined using a combination of a face detector and depth image) they were able to accurately detect hand regions, even when they overlapped with other skin.

2.2. Hand gesture recognition

One approach to recognizing hand gestures is to build a model of the joints of the hand and to try to determine the location of these points on an image. Erol et al. give a survey of such techniques in [2]. The full kinematic model of a hand has 27 degrees of freedom. To deal with all of these degrees of freedom, constraints of the ranges of motion for each joint and joint angle dependencies are considered. In addition, methods such as placing markers on the hands or tracking the movement of hands through time have been used to make detecting the joint locations possible. Another approach involves comparing the hand image to a set of example images for different gestures. This approach builds a classifier that attempts to put a given image into a class that represents the type of gesture being made. To do this, a vector of features is extracted from the image and compared against the features on the given examples. A variety of features and classifier types have been used to do this. One approach, used by Freeman and Roth [3], used a histogram of gradient orientations in the image as a feature vector. Using gradients made these features invariant to lighting and using a histogram made it invariant to translation. By finding the training feature vector that is most similar to the vector for a given image, they were able to distinguish between 10 different hand gestures. More recently, neural networks have been employed for accurate continuous hand pose estimation. Tompson et al. [8] used a deep convolutional neural network to identify the positions of key points on hands in RGB+Depth images. They trained the neural network to recognize certain hand features and used the results to infer the hand pose.

2.3. Finger Detection and Gesture Recognition

Several different methods of detecting fingers in images exist. Most involve first detecting a hand in the image and using information about the hand position and components to identify fingers. Hands can be detected using some of the aforementioned techniques, e.g. color or shape. Lee and Lee [5] used color and information about consecutive motion between frames of video for hand detection. After isolating the hand from the image, they examined the curvature of the hand region for sharp curves to identify fingertips. To measure the direction of each fingertip, Lee and Lee selected two points along the edge of the finger, one on each side a set distance from the fingertip. They then measured the direction of the vector between the average of these two points and the fingertip.

3. Methods

3.1. Problem Setting

focusing on finding orientation of hands

clear background - for general images, could potentially use background subtraction, with added information based on skin clor as described in prior work

3.2. Data Collection and Evaluation

We collected a set of images to evaluate the relative effectiveness of different methods. These images were collected with a standard webcam. The hand to recognize was approximately three feet away, a distance similar to what someone using a computer might be at. A dot was displayed on the screen and the user pointed at the dot as it scanned across the screen. The dot moved along a grid with spacing of fjdk;fdjsklf pixels and for each location a picture was taken of the user pointing at that location from the webcam. We repeated this procedure fdjkslf times, collecting a total of fdjsklf images for fdjsklfj different locations.

The data was split into two sets, a training set and a test set. The test set consisted of images taken from a different scan of the screen. This was done because images collected during the same scanning process tended to have very similar locations and artificially raised the scores received.

We trained our models on the training set and then collected performance metrics on the test set.

Talk about how there is some noise in the data. Pointing at a location on the screen is very inexact. Can we do some sort of short experiment to show that humans cannot really tell where they are pointing?

3.3. Overview of Methods

Created feature vectors to describe image. We evaluated two different methods of extracting these feature vectors: convolutional neural networks and histograms of gradient orientations. We then trained regressors from these features to the ground truth location. We experimented with several different types of regressors and report here the results for least squares linear regression and K-nearest neighbors regression. Other regression methods did not provide any

significant improvement. We used the scikit-learn library implementation of these methods [7].

3.4. Convolutional Neural Networks

One source of features we evaluated was convolutional neural networks. Convolutional neural networks have been used to get state of the art results in object recognition tasks such as fdjkas;fj that competition [**?**]. It has been found that the values of neurons in the upper layers of the network provide good features for computer vision tasks, even though the network was not trained for these tasks [**?**]. For our features, we used the neural network model trained by fdjslf People for fjksdlfj task. [4] We ran their model and extracted the outputs of the final fully connected layer before the softmax layer that

3.5. Histograms of Gradients

We also evaluated the Histogram of Oriented Gradients (HOG) feature descriptor [1] as a means to detect the location and orientation of the pointing hand. This expands on work done by Freeman and Roth, who used gradients to detect hand gestures. We used a variant of the R-HOG algorithm described by Dalal and Triggs in [1]. We divided the image into an 8×12 grid, yielding approximately 53×60 pixel cells. We divided orientations into 9 bins from 0°-360°. In contrast to the block-normalization scheme used by Dalal and Triggs, all cells were scaled by the same value so the l^2 norm of the cells was 1.

3.6. Classifier

We experimented with several different classifiers including k-Nearest Neighbors, linear regression, and SVM. We found that a combination of k-nearest neighbors and linear regression worked best across a wide variety of data. We trained the k-nearest neighbors and linear regression models from sci-kit learn separately on 15 sets of test data comprised of the HOG features computed from each image. These models each predicted a location based on HOG input. These locations were averaged together and fed into a smoothing algorithm to compute the final location.

3.7. Smoothing

In order to make the system more robust to noise, we both smooth the location and eliminate outliers. Let $x[n]$ be the location at frame n and $u[n]$ be the location output from the k-NN and linear regression combination at frame n. Outliers are eliminated if

$$\sqrt{\sum_{i=n-l}^{n} (u[n] - x[i])^2} >= \epsilon \tag{1}$$

where l and ϵ are parameters representing the amount of history to remember for computing outliers and the sensitivity to outliers, respectively. We used $l = 10$ and $\epsilon = 200$. Smoothing is implemented by the formula

$$x[n+1] = \begin{cases} \lambda x[n] + (1-\lambda)u[n] & \text{if not an outlier} \\ x[n] & \text{otherwise} \end{cases} \tag{2}$$

where λ is a smoothing parameter between 0 and 1. We used $\lambda = 0.5$ to balance quick responsiveness with robustness to noise.

4. Results

4.1. Performance on Test Data

4.1.1 Performance of Difference Classifiers

4.2. Evaluation as a Computer Interaction Device

For convolutional neural networks, the time to process an image was too high to allow for real time computer interaction, since each frame took several seconds to process. Using a GPU to process frames would greatly increase the speed and should allow for the possibility of using these features in a real time system.

We used the HOG feature based system for moving a dot on a screen to test its effectiveness as an input device.

5. Conclusion

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[2] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1):52–73, 2007.

[3] W. T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *International Workshop on Automatic Face and Gesture Recognition*, volume 12, pages 296–301, 1995.

[4] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[5] D. Lee and S. Lee. Vision-based finger action recognition by angle detection and contour analysis. *ETRI Journal*, 33(3):415–422, 2011.

[6] E.-J. Ong and R. Bowden. A boosted classifier tree for hand shape detection. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 889–894. IEEE, 2004.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (TOG)*, 33(5):169, 2014.

[9] M. Van den Bergh, E. Koller-Meier, F. Bosché, and L. Van Gool. Haarlet-based hand gesture recognition for 3d interaction. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1–8. IEEE, 2009.

[10] M. Van den Bergh and L. Van Gool. Combining rgb and tof cameras for real-time 3d hand gesture interaction. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 66–72. IEEE, 2011.