

HAND GESTURE RECOGNITION USING NEURAL NETWORKS FOR ROBOTIC ARM CONTROL

Gunawan Ariyanto*, Pak Keung Patrick Li†, Hing-Wah Kwok‡, Ge Yan†

School of Electrical Engineering, Faculty of Engineering, Muhammadiyah University of Surakarta
Jl. A. Yani Tromol Pos 1, Pabelan, Kartasura, Surakarta 57102
e-mail: gariyanto@gmail.com*, {3060375,3134552,3017105}@cse.unsw.edu.au†

ABSTRAK

Hand gesture recognition is one of well-research vision system projects. There are numerous projects which use different techniques to tackle different aspect of the problem. In this paper, we present a project to use hand gesture recognition system for controlling a 5 degree of freedom robotic arm, i.e. SCORBOT. We use Cam-shift Algorithm, Principal Component Analysis (PCA) and Artificial Neural Networks (ANNs) to build the image-based hand gesture recognition system. Cam-shift is used to track the hand image and PCA is used to reduce the feature dimensions of image. At the end of the recognition system, we employ Artificial Neural Networks to classify the image of static hand gestures. We have successfully developed natural language to control the robot and our system is able to recognize six different static hand gestures (poses), handle some noise of image acquisitions process, and implement some modes of robot control. The average performance of the system to recognize the static hand gestures is larger than 90% and the robot is able to do some simple jobs by using hand gesture language commands as its input.

Kata kunci: hand gesture recognition, Cam-Shift, PCA, Artificial Neural Networks.

Makalah diterima [tanggal bulan tahun]. Revisi akhir [tanggal bulan tahun].

1. INTRODUCTION

Hand gesture recognition is one of well-research vision system projects. There are numerous projects which use different techniques to tackle different aspect of the problem. Some of the projects are intended to be implemented in real applications, such as to control the industrial robot.

Vision-based hand gesture recognition system to control an industrial robot remotely using two cameras has been build by Juan W. *et al.* [3]. They employed a fuzzy

clustering technique for recognizing tasks. In order to control the robot remotely, they have setup an interface using TCP/IP communication protocol. Birk, H. *et al.* [1] used Principal Component Analysis to extract the features of hand's pose images and then employs a simple distance calculation of each vector to classify the images.

Another research in gesture recognition was conducted at UNSW. Sze *et al.* [4] used Artificial Neural Networks to recognize the hand's pose image. The purpose of this research is for controlling the robotic arm using Rubik's Cube and simple hand gesture. Rubik's cube was chosen because of it has six different colors on each side and they implemented cam-shift algorithm for tracking the cube.

Bradski, G. R. [2] introduced the cam-shift algorithm for object tracking based on color probability distribution. The algorithm is an extension (the adaptive version) of the previously available mean-shift algorithm. Cam-shift has been successfully implemented for tracking the flesh color objects, such as hand and face.

Another research paper which is reported by Roth, W. *et al.* [5] used the idea of orientation histograms to recognize hand gestures. An image is first converted into a feature vector, which is then compared with feature vectors of the training set of gestures. Using a Euclidean distance metric, the authors trained a system that could let the user control a computer graphic crane by hand gestures as well as play games such as scissors/paper/stone.

In this paper, we present a project to use hand gesture recognition system for controlling a 5 degree of freedom robotic arm, i.e. SCORBOT. We use cam-shift algorithm, PCA (Principal Component Analysis) and Artificial Neural Networks to build the image-based hand gesture recognition system.

2. ARCHITECTURE OF OUR SYSTEM

Our system architecture consists of three main modules, i.e. vision/image processing, gesture recognition, and robotic arm driver. Figure 1 shows the overall architecture of our system.

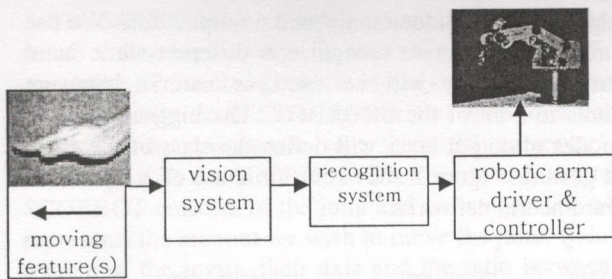


Figure 1 Overall architecture of our system

The vision system uses a frame grabber card to capture motion by a composite camera. The camera is mounted at the top of the shelf looking down on a back carpet placed on the desktop where the user's hand is operating on. Using OpenCV, an open source visual processing library, frames that grabbed by the camera is processed frame by frame hence perform hand gesture recognition to interpret user's command. These commands then send to the robotic arm's controller to perform predefined motion in real time thus allow visual feedback to the user.

The robotic arm used in this project is called SCORBOT, it is a robotic arm with 5 joints that can rotate either horizontally or vertically or both, with the exception of the clamp which opens and closes. It moves with a relative co-ordinate system and each joint can be controlled by commands sent from the computer that interfaces with the SCORBOT.

The vision process, recognition system and robotic arm driver is running on a 700MHz Pentium-base system operating on Debian Linux stable release with OpenCV version beta 4 installed.

2.1 Vision System

The vision system uses cam-shift algorithm to track the user's hand. For each video frame, the raw image is converted to a color probability distribution image via a color histogram model of the hand's color being tracked. Some constraints are applied in our vision system in order to capture and track the user's hand properly. The system assumes the user's hand is longer along extended finger direction and by assuming that we can find an orientation angle which we can use to adjust for palm axis rotation. Due to that reason, user is advised to pull up their sleeve so the orientation can be detected. The system is used in an indoor environment where the light condition is more constant.

Cam-shift Algorithm stands for Continuously Adaptive Mean-Shift Algorithm, as the name suggests it is an adaptive version of the mean shift algorithm. Cam-shift and mean-shift works based on a color probability distribution and a search window. While in mean-shift the search window size is constant, in cam-shift it is able

either to shrink or to expand. Mean-shift algorithm is initially started by choosing a search window size and its initial location. The next step is to compute the mean location in the search window based on equation 1. $I(x,y)$ is the pixel color probability value in the position (x,y) in the image, M_{00} is the zero moment, M_{10} and M_{01} are the first moment for x and y respectively.

$$\begin{aligned}
 M_{00} &= \sum_x \sum_y I(x,y) \\
 M_{10} &= \sum_x \sum_y xI(x,y) \\
 M_{01} &= \sum_x \sum_y yI(x,y) \\
 C_x &= \frac{M_{10}}{M_{00}}, C_y = \frac{M_{01}}{M_{00}}
 \end{aligned} \tag{1}$$

Then the coordinate of the mean location (C_x, C_y) is used as the center of the search window. These steps are repeated until the distance between the center and the mean location less than the predefined threshold.

The cam-shift algorithm adjusts the search window size based on the size and location of the probability distribution which is changing in time when an interested object in video sequences is moving and being tracked. Window size is set to a function of the zero moment.

The output of the cam-shift algorithm is the search window and its position is automatically changed to follow the hand's user movement. The search window is labelled as a region of interest (ROI) and only the image within the ROI will be processed further in the Gesture Recognition System.

2.2 Gesture Recognition System

Gesture recognition system is broken down into two main parts, i.e. the Principal Component Analysis (PCA) and the Artificial Neural Networks (ANNs). PCA is used to produce vector coefficients of an image given an eigen system, and ANNs works to classify those coefficients into some static hand gesture classes.

PCA is a method of reducing the dimension of the problem state. In short, PCA constructs a representation of the data with a set of orthogonal basis vectors generated by a set of sample images. By choosing the most dominant vectors, the dimension of the sample images is reduced at the expense of little loss of information. By projecting a matching image by the same dominant vectors, matching to the sample images can be done by matching the coefficients resulted by the projection. The number of coefficients is equal to the number of dominant vectors used for the projection.

Let us define an image as an object $u = \{u_1, u_2, u_3, \dots, u_n\}$ as a vector in the n -dimensional space. Its components u_i are the image pixel values and in this case n is equal to the number of pixels in the image. Then,

consider a group of input objects $u^i = \{u_1^i, u_2^i, u_3^i, \dots, u_n^i\}$, where $i = 1, 2, \dots, m$ and usually $m \ll n$. The averaged object $\bar{u} = \{\bar{u}_1, \bar{u}_2, \dots, \bar{u}_n\}$ of this group is defined as follows:

$$\bar{u}_i = \frac{1}{m} \sum_{k=1}^m u_k^i \quad (2)$$

Covariance matrix $C = [c_{ij}]$ is a square symmetric matrix $m \times m$:

$$c_{ij} = \sum_{l=1}^n (u_l^i - \bar{u}_i)(u_l^j - \bar{u}_j) \quad (3)$$

Eigen objects basis $e^i = \{e^1, e^2, \dots, e^m\}$, $i = 1, \dots, m$ of the input objects group may be calculated using the following relation:

$$e^i = \frac{1}{\sqrt{\lambda_i}} \sum_{k=1}^m v_k^i (u_k^i - \bar{u}_i) \quad (4)$$

where λ_i and $v^i = \{v_1^i, v_2^i, \dots, v_m^i\}$ are eigenvalues and the corresponding eigenvectors of matrix C . Any input object u^i as well as any other object u may be decomposed in the eigen objects m_1 - D sub-space. Decomposition coefficients of the object u are:

$$w_i = \sum_{l=1}^n e_l^i (u_l - \bar{u}_l) \quad (5)$$

In our system, a set of 60 images is used to create the PCA system. The image size is 100 by 50 pixels and these images are generated from the tracking (search) window by the cam-shift algorithm. We also decide to choose 10 decomposition coefficients of the object (image). Based on the description of our system above, we know that the value of $n = 5000$, $m = 60$, and $m_1 = 10$.

Building the eigen objects is a once-off calculation at the beginning of the program. After the eigen objects is builded, the system will only calculate the decomposition of coefficients vector of every new coming image. The vector is as an output of the PCA system and will be classified by ANNs.

OpenCV provides some libraries to implement PCA algorithm. It has *cvCalcEigenObjects* to calculate the eigen objects as well as the averaged object, and *cvEigenDecomposite* to calculate the decomposition coefficients of the image.

The next step is to use artificial neural networks to classify the decomposition of coefficients vector from PCA. We use three (3) layers feed forward neural networks and back propagation error training algorithm. The hidden layer employs tangent sigmoid as its

activation function and the output layer uses pure linear as its activation function. The network structure consists of 10 input units, 16 hidden units, and 6 output units. We use the neural networks to recognize 6 different static hand gesture and those will be used as natural language symbols to control the SCORBOT. The biggest value in the nodes of output layer will define the class of the static hand gesture. Figure 2 shows the structure of 3 layer feed forward neural networks.

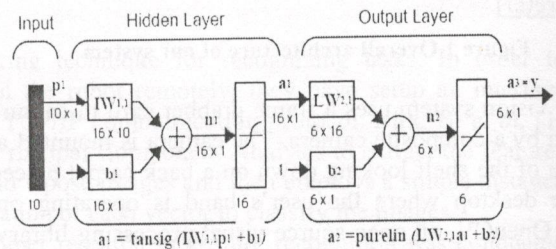


Figure 2 Three layer feed forward neural network

In order to train the ANNs, we use an offline training strategy and choose MATLAB as an implementation tool. There are some parameters that MATLAB need to run the training process. Those parameters are the epoch number, back propagation implementation method, learning rate, and momentum rate. All parameters in Neural Network's training process and its corresponding value are described in table 1.

Table 1 ANN's training process parameters

Parameter	Value
Epoch	1000
Back propagation implementation's method	Gradient descend back propagation
Learning rate	0.1
Momentum rate	0.9

After we define the structure and training algorithm of the neural networks, then we need to collect the training data in a reasonable number. In order to make life easier, we design to collect data online by capturing image every 80-100ms. The number of training data is quite huge, i.e. about 2400 total of sample data with an average of 400 data for each gesture.

We create some piece of MATLAB code for this purpose to read the training data file, to train the Neural Networks, and then to save all the weights in a file. What we get after training phase is a file of ANN's weights that contains the weights matrix of input layer, hidden layer, and bias.

2.3 Robotic Arm Driver

The SCORBOT is a robotic arm with 5 joints that can rotate either horizontally or vertically or both, with the exception of the clamp which opens and closes. It moves with a relative co-ordinate system and each joint can be controlled by commands sent from the computer that interfaces with the SCORBOT. Each command sent to the SCORBOT consists of the joint and an integer unit which represents the amount we wish to move the joint. Below is a table of the joints, their axis and the ratio between the integer unit and the angle.

Table 2 SCORBOT joints and its parameter

Joint	Units	Angle	Ratio	Axis
Base	963	90 degree	10.7 units/degree	Horizontal
Shoulder	756	90 degree	8.4 units/degree	Vertical
Elbow	756	90 degree	8.4 units/degree	Vertical
Wrist	190	180 degree	1.05 units/degree	Vertical
Wrist	190	180 degree	1.05 units/degree	Horizontal

The clamp opens and closes with 190 units.

SCORBOT has an ability to receive and cache multiple commands at a time and perform relative co-ordinate system. The relative co-ordinate system means that after a command is sent to the SCORBOT and it starts moving, the SCORBOT will immediately update itself and process the following command with respect to the position it is moving towards. For example, if a command is sent to the SCORBOT to rotate the base clockwise 1000 units. Let us say this takes 4 seconds, and 2 seconds later another command sent to the SCORBOT to rotate the base anti-clockwise 500 units, then the base will stop instantaneously. Any other command will move the particular joint in conjunction with the base.











Multiple joints can be moved simultaneously for the SCORBOT, but due to the lack of control over the speed of the joints, it is difficult to combine multiple-joint movement to get to a desired position faster. Thus we focus on moving single joint at a time.

3. DEVELOPING NATURAL LANGUAGE

There are many factors which are taken into consideration for the development of natural language to control the SCORBOT. The main question we need to answer is what we want the SCORBOT to do and what the most natural way to achieve this.

Table 3 describes the natural language which we have proposed.

Table 3 The Natural Language

Pose	Hand Movement	Joint	Joint Movement	Mode
	Left or Right	Base	Left or Right	Relative
	Up or Down	Shoulder	Up or Down	Relative
	Diagonals	Elbow	Up or Down	Relative
	Left or Right	Base	Left or Right	Continuous
	Up or Down	Shoulder	Up or Down	Continuous
	Diagonals	Elbow	Up or Down	Continuous
	Left or Right	Wrist	Twist	Relative
	Up or Down	Wrist	Up or Down	Relative
	Writing	N/A	N/A	Writing
	N/A	N/A	N/A	Global Stop

3.1 Determining the Direction of Movement

We restrict potential movement to the eight-directions up, down, left, right and their respective diagonals.

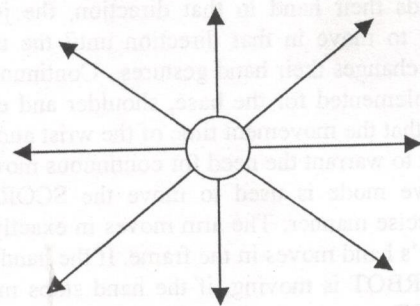


Figure 3 the directions of movement

We consider using a full 360 degrees of movement, and decide to split the grid into eight equal pieces as in figure 3, which would determine the direction of movement from the user.

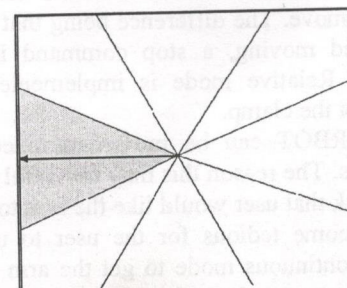


Figure 4 the area of movement

From figure 4, any movement from the centre point (where user's hand is) into the yellow area would be detected as a left movement. The centre point is the

current location of the hand. Movement is detected only between two frames.

3.2 Determining the Type and Mode of Movements

We decide on two different control types, controlling the arm in real-time movements and controlling the arm with pre-defined movements.

Real-time movement is, as the name suggests moving, the SCORBOT relative to the movement of the hand in the frame. With real-time movement, there are two distinct control modes. These are continuous movement mode and relative movement mode.

Continuous mode is useful for moving the SCORBOT in large distances. If for example user need to move the base left 180 degrees it seems more natural to just hold their hand in a left position and allow the SCORBOT to move continuously until the user tells it to stop, rather than issue many smaller left commands until it reaches where the user want the arm to be. To initiate continuous mode, the user places their hand in the "centre box" and then move their hand in one of the eight directions. If the user holds their hand in that direction, the joint would continue to move in that direction until the user moves away or changes their hand gestures. Continuous mode is only implemented for the base, shoulder and elbow. It is decided that the movement time of the wrist and clamp are too short to warrant the need for continuous movement.

Relative mode is used to move the SCORBOT in a more precise manner. The arm moves in exactly the same way user's hand moves in the frame. If the hand is moving, the SCORBOT is moving, if the hand stops moving, the SCORBOT stops moving. This is useful for more precision movement. For example if user is trying to line the arm up to pick something up, the user can use relative mode to just to get the arm to the exact location they want it. Relative mode can be activated from anywhere in the frame. And relative mode is only terminated once the user changes their hand gestures. Similar to continuous mode, a command is sent to the SCORBOT telling it which direction to move. The difference being that each time the hand stopped moving, a stop command is sent to the SCORBOT. Relative mode is implemented for all the joints, except the clamp.

The SCORBOT can be pre-programmed to perform specific tasks. The reason this may be useful is if there is a repetitive task that user would like the arm to perform, and it would become tedious for the user to use either the relative or continuous mode to get the arm into the right position. However, through pre-defined movements it could save a whole lot of tedious movement.

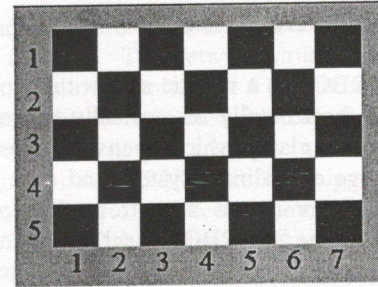


Figure 5 the domain of pre-defined movements

The domain of pre-defined movements is a checker board domain like in figure 5. The checker board as shown above has got x and y grid coordinates. The SCORBOT has been pre-programmed with how to pick up and put down checker pieces. We decide that the input method is to get the user to virtually "write" the co-ordinates of the piece which they want to pick up and then "write" the co-ordinates of the position they want to place the piece.

There is one control mode for pre-defined movement, i.e. writing mode. The user activates writing mode by showing the gesture for writing. Once in writing mode, all the other gestures are disabled. Thus, user can not move the SCORBOT in relative or continuous mode. To re-enter real-time mode, the gesture for abort needs to be shown.

Table 4. The Drawing of Hand Gesture

Number	Drawing	Instructions
1		Down
2		Right, Down, Left, Down, Right
3		Right, Down, Left, Right, Down, Left
4		Down, Right, Down
5		Left, Down, Right, Down, Left







4. RESULT

The overall system can maintain 10-12fps which is considered running in real-time environment. The tracking module using cam-shift works quite well for images with static and uniform background. Without any intrusion of skin colored object, the tracking result is excellent.

On the best of times, the neural network was 90-95% accurate. Table 3 shows the performance and accuracy of the neural networks on 6 different static hand gestures.

SCORBOT was able to do some simple job in real time such as dispatching small and light stuffs based on hand gesture command. The pre-defined movement also worked smoothly to do some simple repetitive tasks like moving stuff from one position to another position based on cheek board coordinate.

Table 5. Neural Networks accuracy

Static Hand Gesture Image	PERFORMANCE		
	Number of True	Number of False	Percent Accuracy
	381	17	95.7%
	477	49	90.7%
	410	11	97.4%
	570	20	96.6%
	471	17	96.5%
	456	16	96.6%

5. CONCLUSION

This paper has presented an alternative solution for controlling robotic arm using hand gesture. Our solution is based on some algorithms ranging from image processing and computer vision to pattern recognition algorithm. Cam-shift algorithm helps to maintain the orientation and position of hand's pose image and PCA is implemented in order to extract the features of images. Eventually, the artificial neural networks works as a recognition system to classify the decomposition of coefficients vector of each hand poses into its corresponding natural language. The experimental results show that the system is able to do its job fairly good with 90-95% accuracy.

6. ACKNOWLEDGMENT

The authors appreciate the Robotics Laboratory of The School of Computer Science, The University of New South Wales, Australia, for providing the SCORBOT and other facilities used in this research project.

REFERENSI

- [1] Birk, H.; Moeslund, T. B., "Recognizing Gesture From the Hand Alphabet Using Principal Component Analysis", Master's Thesis, Laboratory of Image Analysis, Aalborg University, Denmark, 1996.
- [2] Bradski, G.R., "Computer Vision Face Tracking As A Component of A Perceptual User Interface", Workshop on

Applications of Computer Vision, Princeton, NJ, 1998, pages 214-219.

- [3] Juan, W.; Uri, K., "Hand Gesture Telerobotic System Using Fuzzy Clustering Algorithms", Integrated Project Report, Ben-Gurion University of the Negev, 2001.
- [4] Sze, I.; Kong, A.; Li, J., "Robot Control Using Gesture", Experimental Robotics Project Report, School of Computer Science and Engineering, UNSW, Sydney, 2004.
- [5] Roth, M.; Freeman, W., "Orientation Histogram for Hand Gesture Recognition", International Workshop on Automatic Face and Gesture Recognition, Zurich, 1995.
- [6] Open Source Computer Vision Library Reference Manual, Intel Corporation, 2001.