Stereovision based 3D hand gesture recognition for pervasive computing applications

Vikram Shenoy H*, Pankaj Bongale*, Vineet Roy† and Sumam David*
*Department of Electronics and Communication Engineering
National Institute of Technology Karnataka, Surathkal, INDIA
Email: vikkyshenoy@gmail.com
†Texas Instruments India, Bangalore, INDIA

***Abstract*—Hand gesture recognition, being one of the most intuitive means of Human Computer Interface has spawned many applications in the area of pervasive computing devices. This paper presents a solution for hand gesture recognition problem based on 3D stereo imaging techniques employing a low complexity algorithm on a low cost sensor. This method significantly improves upon conventional 2D techniques which are limited in their approach due to various reasons which include but are not limited to occlusion, difficulty in establishing classifiers, segmentation problems using template matching and noise. The innovation here is the use of a hybrid method that combines a simple state machine, skin tone detection and stereo cameras to provide a fast method for detecting simple hand gestures that can be implemented on most low cost System on Chips (SoC). The experimental results from the test setup are promising.**

I. Introduction

In the past few years, we have witnessed a rapid evolution in the field of Human-Computer Interaction (HCI) which necessitates state-of the art research in the field of gesture recognition. Accessibility of low-cost webcams has opened up avenues for a lot of applications in interactive gaming and applications as hand mouse and other natural user interfaces. Some of the major application areas are entertainment [1], medical systems and crisis management applications [2]. Many gaming devices are developed using Hand Gesture Recognition (HGR) such as Kinect, Wii and PSP [3].

Being a more intuitive way of user interface compared to touchscreen, HGR has even attracted researchers in the area of pervasive computing applications [4]. With the advent of inexpensive 2D cameras, HGR seemed possible and many researchers have succeeded in achieving very good results. Despite this success, there are several challenges which affect the results of 2D HGR like the presence of background noise and similar classifiers such as face or other humans. Although some of the studies involving background subtraction [5] for hand tracking may suggest some improvement, the very assumption of an inert background might still lead to misinterpretation.

The challenges associated with 2D HGR have propelled some researchers to work on 3D HGR because of the possibility of recognizing more gestures. 3D hand gesture recognition is made possible using two techniques: Kinect (range vision) and stereoscopic vision. The basic difference between the above two methods is that in Kinect, a RGB and an IR sensor are used whereas in stereoscopic cameras, both sensors are RGB. Stereo vision based approach was chosen for HGR as inexpensive stereo cameras are available compared to the costlier devices with multiple sensors. It is intended for handheld devices, so we only target image based HGR devices. The proposed solution can be easily implemented on many of the SoCs that power todays smart phones. These devices, with their powerful CPUs, GPUs and image/video accelerators are already capable of handling multiple camera streams (front and back camera) in real time. This image processing capability can be easily tuned to handle two standard definition stereo images in real time with lens distortion correction. The rest of the processing can be handled by a combination of DSPs and SIMD capable vision accelerators. To give an idea, Fig.1 shows how the software architecture of our proposed implementation would look like.

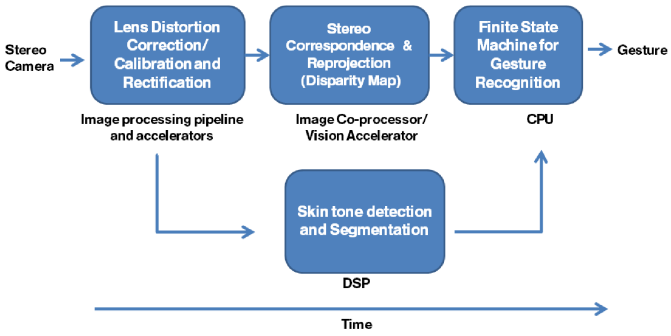

Fig. 1: Software architecture of our proposed implementation

II. Related work

In past decade, many researchers have worked on 2D and 3D hand gesture recognition. Conventional 2D HGR uses three popular techniques namely colour based [6], motion based [7] and appearance/model based [8]. Colour based method involves skin tone detection to separate the hand from background. Motion based segmentation often involves background removal or absolute difference between two consecutive frames, thereby tracking only the moving hand. Appearance based techniques either include supervised method like SVM [9] to learn the user's gestures or unsupervised method,

where a model based approach is used to fit a hand gesture model to recognize a gesture [10]. However, analyzing the scene becomes difficult in case of 2D hand gesture recognition. 3D information helps to mitigate occlusion in complex backgrounds. However, most researchers were skeptical in using 3D HGR, because of its computational complexity. Taking into consideration, its advantages and drawbacks, there is a need to design a simple but computationally faster algorithm which performs an efficient hand gesture recognition. The main objective of this work is to develop a 3D hand gesture recognition for pervasive computing applications like swipe, scroll, open and close gestures. Although Hidden Markov Model (HMM)s are very popular in the field of dynamic HGR, it is not used since the number of dynamic gesture sequences are small. HMM is preferred for gestures involving numbers, circles and sign language terms. Thus, a more simpler Finite State Machine (FSM) based approach for dynamic gesture recognition is chosen [11]. Prior to this stage, the steps involved in getting an accurate depth map like undistortion, stereo calibration, stereo rectification and stereo calibration are briefly discussed in the next section.

III. Stereo Image Pre-processing

The main goal is to map the depth data onto an RGB image so that the foreground hand image from background objects can be segmented. We, human beings are familiar with the stereo imaging capability that our eyes give us. Computers perform this task by finding the correspondence between points that are seen by one imager and the same points as seen by the other imager. With such correspondence and a known baseline separation between cameras, 3D location of the points can be computed [12]. The brief methodology is shown in Fig.2

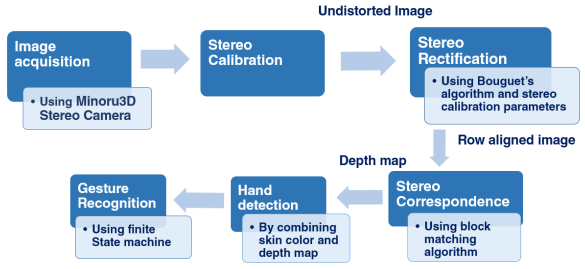

Fig. 2: Flowchart of our proposed methodology

Generating a depth map from left and right camera images involves the following steps:

1) Stereo calibration
2) Stereo rectification and
3) Stereo correspondence
4) Reprojection

A. Stereo Calibration

Stereo calibration is a onetime operation which generates a set of values which are then used to correct the distortion generated by the lens and camera. Mathematical details on lens distortion correction can be found in [12]. In calibration step, rotation (R) and translation (T) vectors are found out using (1) and (2) respectively.

$$R = R_r(R_l)^T \tag{1}$$

$$T = T_r - RT_l \tag{2}$$

where, R_r and R_l are the rotation vectors of right and left cameras respectively. Similarly, T_r and T_l are translation vectors of right and left cameras respectively.

B. Stereo Rectification

Stereo rectification is the process of correcting the individual images so that they appear as if they had been taken by two cameras with row-aligned image planes [12]. There are two popular algorithms to perform stereo rectification, Hartley's algorithm and Bouguet's algorithm. Bouguet's algorithm is chosen over Hartley's algorithm to minimize the reprojection error because of the greater accuracy it offers [12]. Also, the long-term aim is to employ this module to manipulate a pervasive computing device so not having an idea of the scale of the image (drawback of Hartley's algorithm) would be unacceptable.

C. Stereo Correspondence

This step gives us a disparity map, where the disparities are the differences in x-coordinates on the image planes of the same feature viewed in the left and right cameras [12]. Based on the horizontal distance between the locations of a point in the two images and a set of predefined constants, a disparity image is generated. Once the physical coordinates of the cameras or the sizes of objects in the scene are known, depth measurements can be derived from the triangulated disparity measures $d = x^l - x^r$ or $d = x^l - x^r - (c_x^{left} - c_x^{right})$ if the principal rays intersect at a finite distance between the corresponding points in the two different camera views which is illustrated in [12]. Block-matching algorithm was used to perform stereo correspondence.

Block-matching algorithm: This algorithm was developed by Kurt Konolige [13]. The block matching algorithm was chosen over Graph Cut (GC) algorithm since it is faster compared to GC [14]. This algorithm finds only the strongly matching (high-texture) points between the two images. Block matching algorithm works by using small "sum of absolute difference" (SAD) windows to find matching points between the left and right stereo rectified images. SAD window $C(x,y,\delta)$ is mathematically represented as,

$$C(x,y,\delta) = \sum_{y=0}^{wh-1} \sum_{x=0}^{ww-1} |I_R(x,y) - I_L(x+\delta,y)| \tag{3}$$

where, wh is the window height, ww is the window width, δ is the differential value, $I_L(x+\delta,y)$ and $I_R(x,y)$ is the image intensities corresponding to left and right images.

D. Reprojection

If the geometric arrangement of the cameras are known, then the disparity map can be turned into physical distances by triangulation. The output is a depth map. For a given disparity d and 2D point (x, y), a point can be projected into three dimensions using (4) as mentioned in [12].

$$Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} \tag{4}$$

where

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & (c_x - c'_x)/T_x \end{bmatrix} \tag{5}$$

is the re-projection matrix which is the inverse of the projection matrix. In this matrix, c_x and c_y are the principal points and 2D-screen co-ordinates are (*x/w, y/w*). c_x is the principal point x-coordinate from right image for Q matrix of left image. T_x is the translation vector. 3D co-ordinates are given by *X/W,Y/W,Z/W*. The depth Z is calculated as,

$$Z = \frac{fT}{x_l - x_r} \tag{6}$$

where, f is the focal length of camera and T is the baseline distance. x_l and x_r is the position of left and right camera respectively.

IV. Implementation

A. Hand detection

Usually, hand detection is implemented either by using background removal (provided the hand is moving) or skin-tone detection. However, face regions are also visible when skin tone detection is used. Hence, a hybrid method involving both skin colour detection and then depth mapping is used so as to separate out hand from rest of the background.

A skin tone detection model has been built using multiple thresholding of different colour spaces namely YCbCr, HSV and RGB. This takes care of illumination variance and similar coloured background. Most suitable ranges in YCbCr colour space are obtained from algorithm proposed by [15]. Similarly, thresholds in RGB colour space are rooted in the work presented by [16]. Finally the thresholds in HSV colour space are set based on the results shown by [17]. Later on, the depth information is used to distinguish between the hand as foreground object and everything else (including face) as background object. This method has been earlier used by [18], however for depth data, they had used Time of Flight (ToF) cameras instead of stereo camera. Using a face detector (i.e. OpenCV) on the RGB image, the position (x, y) and size (w, h) of the face can be determined. The distance from the face to the camera can be estimated as the average depth values of the face d_{face} as,

$$d_{face} = \frac{1}{wh} \sum_{i=x}^{x+w} \sum_{j=y}^{w+h} I'_{stereo}(i,j) \tag{7}$$

where, I'_{stereo} is the projected depth image. If the face is occluded, the previous position and distance is used. All the objects in front of the face can then be found by applying a fixed threshold t_s. It defines how far the user has to extend the hand towards the camera for it to be accepted as an interaction.

B. Gesture Recognition after getting depth map

Since this study involves only six gestures like *palm open, grab (close), swipe left, swipe right, scroll up and scroll down*, a simple Finite state classifier is sufficient. However, it should be noted that if complex gestures like circle, Arabic numbers (0-9) are to be recognized, then HMM is required. Palm open and close gesture are static, swipe and scroll gestures are dynamic. The methodology is presented in Fig.3. The methodology is based on the fact that, while using HGR for smartphone/ tablet interface, no other objects are present in between stereo camera and user's hand. Hence, the depth information to distinguish between hand (foreground object) and the rest of background is used. The static gestures are recognized by using the pixel count for last 10 consecutive frames. If it is greater than the threshold (in our case, $\tau = 6000$), then it is *palm open* or *halt* gesture else it is *close* or *grab* gesture. This can be actually used to *select* the application running on a smartphone or tablet if the previous gesture is *open* and current gesture is *close*. Once this is done, dynamic gestures are recognized as follows,

C. Dynamic Gesture recognition

To track the movement of the palm in the respective dimension, counters d_x, d_y and d_z are used . In every new frame, it is examined whether the hand has been stable in the last 10 frames. $Hand_{stable}$ consists of two states : *open* and *close*. In case of a stable hand, $Hand_{stable}$ is updated and counters are reset. Otherwise, the absolute displacement of the palm in the last two frames is added to the counters.

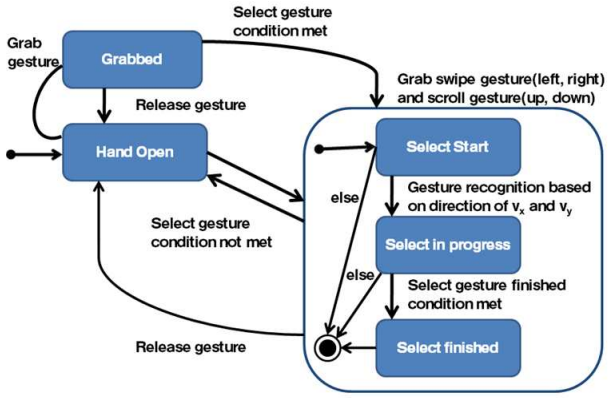

Fig. 3: Finite state machine for gesture recognition

$$d_x^t = d_x^{t-1} + |P_{palm}^t x - P_{palm}^{t-1} x| \tag{8}$$

$$d_y^t = d_y^{t-1} + |P_{palm}^t y - P_{palm}^{t-1} y| \tag{9}$$

where, P_{palm}^t is the position of the palm in current frame and P_{palm}^{t-1} is the position of the palm in previous frame. After this, the displacement $\overrightarrow{v}$ between the palm position of the last stable hand model $Hand_{stable}$ and the recent hand model $Hand_t$ is computed as,

$$\overrightarrow{v} = P_{palm}^t - P_{palm}^{stable} \tag{10}$$

Next, the direction in which $\overrightarrow{v}$ has the maximum absolute value is determined. Only the *swipe* gestures that occur in this dimension are considered. Both *X*-dimension and *Y*-dimension are considered analogous. For a given direction of *swipe*gesture, the absolute sum of the displacement of motion in other two dimensions should not exceed a threshold. If it exceeds, then the counters d_i in the direction $i = x, y, z$ will be reset.

The swipe and scroll gestures are recognized based on the direction of the vector v_x and v_y, i.e

$v_x < 0$, the *swipe* gesture is towards *left*
$v_x > 0$, the *swipe* gesture is towards *right*
$v_y > 0$, the *scroll* gesture is towards *up*
$v_y < 0$, the *scroll* gesture is towards *down*

V. Experimental results and discussion

Minoru3D, an inexpensive 3D stereo camera is used for stereo imaging. Its resolution at 30fps is 320x240 and 640x480 for $<$ 15fps. Although this is at the lower end of our requirements a resolution of 320x240 is sufficient for our experiments. Skin tone detection is implemented prior to depth map generation. Results obtained from multiple thresholding based skin colour detection is shown in Fig.4

Fig. 4: results of skin tone detection

Pictures of a 8x5 chessboard were taken for stereo calibration. A total of 160 different pairs of pictures were taken with the chessboard held in different inclinations. Rotation and translation matrices were calculated for each of these pairs, hence a total of 160 pairs of matrices. `cvStereoCalibrate()` function in OpenCV was used to generate these matrices which were then approximated using Marquardt iterative algorithm into one pair (R,T) with minimum error of chessboard corners for both camera views [12].

An OpenCV function called `cvStereoRectify()` corresponding to Bouguet's algorithm was used. Then the output of the function was fed to another function `cvInitUndistortRectifyMap()` which completes the stereo rectification step on the two images. The result after rectification is applied to one pair (out of the 160 pairs) of images is shown in the Fig.5

Fig. 5: Result images after stereo rectification

`cvFindStereoCorrespondenceBM()`, an OpenCV function based on efficient block matching stereo algorithm is used for getting depth map. In our study, SAD window of size 9 is used.The stereo correspondence result obtained from block matching algorithm is shown as colour depth map in Fig.6. It can be observed that the colour intensity changes as the object gets closer.

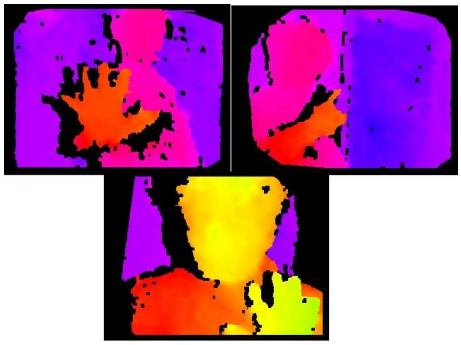

Fig. 6: colour depth map

The performance assessment was done using 20 test cases for each gesture and we achieved overall accuracy of 93.75% within the nominal distance of 1 feet from the Minoru 3D camera. As the Minoru 3D's field of view is 40 degrees, the minimum distance from which the gesture to be made is 9 cm. However, the HGR accuracy reduces as the distance between the user and the camera increases, which is presented in Fig.8.

The breakup of computational loads for different algorithms is given by [19]. They report that most of the computational complexity arises from computation of depth/disparity map (41%) and object segmentation (53%), tasks which can be allocated to two different processors in a pipelined fashion. (Two middle blocks in the proposed architecture shown in

Fig. 7: results of various gestures

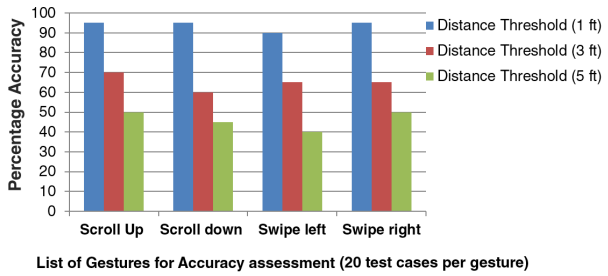

Fig. 8: Performance assessment of various gestures with respect to the distance between user and Minoru 3D webcam

Fig.1). The SIMD capable vision accelerator is ideally suited for calculating disparity maps as it involves parallel processing of large chunks of data and a lot of FIR filters. A DSP with its similar but more extensive capabilities is capable of handling object segmentation while most of the control code can be offloaded to the main CPU which is designed for such tasks. As our software implementation has been built entirely using OpenCV, the task of porting the kernels to the hardware is easier as ports of OpenCV already exist on most SoC platforms.

VI. Conclusion

With 3D stereovision techniques, a depth map is generated along with skin-tone detection to differentiate the hand from the background. With this we could successfully recognize 6 gestures: *palm (open), fist (close), scroll up, scroll down, swipe left* and *swipe right*. With all the results achieved so far, it can be concluded that the proposed method can be used to easily implement simple Gesture Recognition for many pervasive computing applications.

VII. Acknowledgement

We would like to thank Mr Aravindan, Texas Instruments India for his support throughout the project.

References

[1] S. Siddharth and A. Agrawal, "A Vision based Hand Gesture Interface for Controlling VLC Media Player," *International Journal of Computer Applications*, vol. 10, pp. 11–16, 2010.

[2] S. Mitra, "Gesture Recognition: A Survey," *IEEE Trans. on Systems, Man and Cybernetics Pat-C: Applications and Reviews*, vol. 37, pp. 311–324, 2007.

[3] M. Mariappan, X. Guo, and B. Prabhakaran, "Picolife: A Computer Vision-based Gesture Recognition and 3D Gaming System for Android Mobile Devices," in *Proc. of International Symposium on Multimedia*. IEEE Computer Society, 2011, pp. 19–26.

[4] M. Wright, L. Chun-Jung, E. O'Neill, D. Kosker, and P. Johnson, "3D Gesture Recognition: An Evaluation of User and System Performance," in *Proc. of 9th International Conf. on Pervasive Computing 2011*. Vol. 6696 LNCS.Lecture Notes in Computer Science . Springer-Verlag, 2011, pp. 294–313.

[5] Q. Chen, "Real-time Vision-Based Hand Tracking and Gesture Recognition," Ph.D. dissertation, School of Information Technology and Engineering, Ottawa-Carleton Institute for Electrical and Computer Engineering,University of Ottawa, 2008.

[6] T. Starner and A. Pentland, "Real-time Americal Sign Language recognition using desk and wearable computer based video," *IEEE Trans. on Pattern Anal. and Mach. Intell.*, vol. 20, pp. 1371–1375, 1998.

[7] H. Sidenbladh, "Detecting human motion with Support Vector Machines," in *Proc. of the 17th International Conf. on Pattern Recognition*. Vol. 6696 LNCS.Lecture Notes in Computer Science . Springer-Verlag, 2004, pp. 188–191.

[8] J. Guo, "Hand Gesture Recognition and Interaction with 3D Stereo Camera," in *Project report of COMP8740*, 2011.

[9] Y. Chen and K. Tseng, "Multiple-angle Hand Gesture Recognition by Fusing SVM Classifiers," in *CASE*, 2007, pp. 527–530.

[10] Y. Wu, J. Lin, and T. Huang, "Capturing Natural Hand Articulation," in *Proc. of Eighth IEEE International Conf. on Computer Vision, ICCV 2001.*, vol. 2. IEEE, 2001, pp. 426–432.

[11] J. Davis and M. Shah, "Visual gesture recognition," in *IEEE Proc. of Vision, Image and Signal processing*, 1994, pp. 101–106.

[12] G. Bradksy and A. Kaehler, *Learning OpenCV- Computer Vision with OpenCV library*. Newgen Publishing and Data Services.

[13] K. Konolige, "Projected Texture Stereo," in *IEEE International Conf. on Robotics and Automation*, 2010, pp. 148–155.

[14] S. Droppelmann, M. Hueting, S. Latour, and M. van der Veen, "Stereo vision using the opencv library," 2010.

[15] D. Chai and K. Ngan, "Face feature extraction using skin colour map in videophone applications," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 9, pp. 551–564, 1999.

[16] P. P. J. Kovac and F. Solina, "2D versus 3D colour space face detection," in *Fourth EURASIP Conf. on Video/Image processing and Multimedia Communications*, 2003, pp. 449–454.

[17] S. Tsekeridou and I. Pitas, "Facial feature extraction in frontal views using biometric analogies," in *Proc. of the IX European Signal Processing Conference*, vol. 1, 1998, pp. 315–318.

[18] M. Bergh and L. Gool, "Combining RGB and ToF Cameras for Real-time 3D Hand Gesture Interaction," in *Proc. of the IEEE Workshop on Applications of Computer vision (WACV 2011)*, vol. 1, 2011, pp. 66–72.

[19] K. Dong-lk and G. Agarwal, "Gesture recognition: Enabling natural interactions with electronics," *white paper, Texas Instruments*, pp. 1–13, 2012.