Cost Oriented Automation Approach to Upper Body Humanoid Robot

J. Baltes * **C. T. Cheng** ** **M.C. Lau** *** **J. Anderson** ****

* *Autonomous Agent Lab, University of Manitoba, Winnipeg, Manitoba R3T 2N2, Canada (e-mail: jacky@cs.umanitoba.ca)*
** *(e-mail: tkuggt@gmail.com)*
*** *(e-mail: laumc@cs.umanitoba.ca)*
**** *(e-mail: andersj@cs.umanitoba.ca)*

Abstract: To develop an efficient robotic system is always a challenge, in particular if the cost of the system is also an important factor. This paper presents an overview of development of our 10 degree of freedom humanoid, Betty. Reducing the cost of the system requires optimization of all aspects to retain its flexibility, reliability and performance at minimum cost. During the design and development of Betty, we only use low cost hardware and open source software to address both cost and performance issues. We develop a real-time kernel optimized to control servo positions and read back servo data. Parameters of this kernel are controlled by a PID controller resulting in an adaptive real-time kernel. After solving the forward and inverse kinematics of our robot, we implemented portrait drawing as a sample application showing the performance of our system.

Keywords: Real-time Embedded System, inverse and forward kinematics, simulation, portrait drawing, PID.

1. INTRODUCTION

Designing and implementing a cost-oriented automation (COA) system (e.g., a humanoid robot) is a crucial tasks in the development of future service and entertainment robots. Generally humanoid robots like ASIMO, HOAP and QRIO are too expensive to be viable, commercially successful consumer products. They are only affordable to some well-funded research organisations. In recent years there were several research publications discussing drawing robot which usually required expensive equipments, Calinon et al. (2005); Lu et al. (2009); Lin et al. (2009) with high precision position and force feedback. So one of our motivations in the research is to design and develop an affordable humanoid robot system which could overcomes this problem.

By implementing forward kinematics, the length of each link and the angle of each joint is measured so we can calculate its end effector position by using the Denavit-Hartenberg (DH), Spong and Vidyasagar (2006) convention. In inverse kinematics, the length of each link and position of the end effector is given and we have to calculate the individual joint angles. Currently we use a kinematic diagram to solve the inverse kinematics problem. Finally, we test the kinematics control by running it on different applications: simulation, drumming, and portrait drawing. The simulation is developed in OpenGL to provide a 3D model view. We use OpenCV in Betty's visual servoing system to output a sketch-like portrait from the camera so it could be drawn. Visual servoing is using computer vision data in the servo loop to control the motion of a robot, Chaumette and Hutchinson (2006, 2007).

The paper is organized as follows. In section 2 and 3 we will look into hardware and software of Betty. Section 4 introduces forward kinematics and inverse kinematics of Betty.

2. HARDWARE

The Upper body of Betty consists of a ten revolute joints with ten degrees of freedom (DOF). Its head has four DOFs which give pan, tilt, swing and one DOF for the mouth. Each of its arms provides three DOFs, shoulders allow lateral and frontal motions, and and elbows give lateral motion. These joints are constructed by four Dynamixel AX-12 servos in the head and three Dynamixel RX-64 servos for each of its arms as shows in Fig. 2. The main reason we chose RX-64 to construct Betty's arms is it has higher final maximum holding torque, 64.4∼77.2 kgf.cm compared to only 12∼16.5 kgf.cm for the AX-12 Robotis (2007, 2006). It will allow a RX-64 servo to generate sufficient torque to support the weight of the arm.

In order to control the servos, we use Dynamixel's dedicated controller, CM-2+ from Robotis as the central control unit with its `AVR ATmega128` microcontroller. The real-time OS establishes communication with the servos. The connections on the CM-2+ is illustrated in Fig. 1, Fig. 2 and Fig. 3 show the overview of Betty's upper body. For Betty's vision, it has two Logitech QuickCam Orbit MP, 1.3 Megapixel, pan and tilt motorized webcam. The total cost of the hardware is less than USD 3000.

Fig. 1. Overview of Betty's upper body

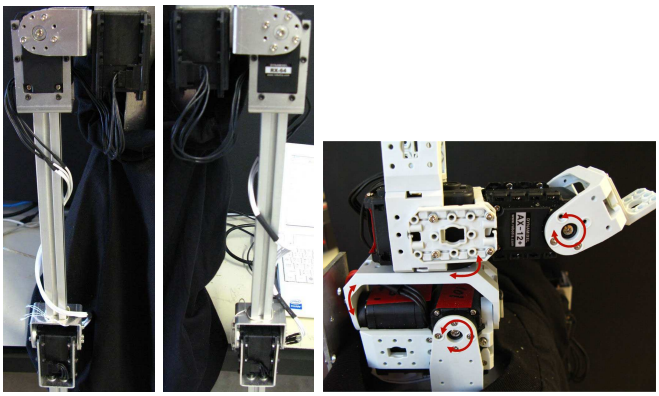

Fig. 2. Betty's joints construction

Fig. 3. CM-2+ Connection

3. SOFTWARE

3.1 Real-time Embedded System

The real-time embedded control system is needed enable the communication between a controller program and the robot's servos. We develop a pre-emptive multi-tasking kernel in the Control Program to handle several tasks. Currently our kernel supports four tasks, Lau and Baltes (2010). The first task is an idle task to toggle a LED which is always ready to run to ensure that at least one thread is running and the task switcher is working properly. The second task is the PC thread that handles serial port communication between the Motion Editor program and the CM-2+. The third task is the servo thread which will prepare commands to send from the CM-2+ to the RX-64 and AX-12 servos. Finally, the last task is the torque thread that reads the current torque of each servo and sends it through the PC thread. The pre-emptive multi-tasking kernel uses a timer interrupt to switch between different threads at an initial timer frequency of 10Hz. To execute task switching, the kernel will save the complete task state on the stack and then store the stack pointer in the task control block (TCB).

Table 1. Communication Protocols

Protocol	Command Structure
Absolute Position	FF011982828282828282828282E5 FF: Header 01: SyncWrite command 19: Speed 82: Position E5: Checksum = lower byte of ∼(01+19+82+82+82+82+82+82+82+82+82+82)

We implement interrupt linear queue as the data structure to handle communication data. Queue is a data structure where item insertions are made at rear and item retrievals or deletions are made at front of the queue. Because the first item added to the queue will be the first one to be removed so it is commonly refer as first-in-first-out (FIFO) data structures, Jeff (2010). We also implemented circular queue solution and the experimental results are discussed in our other publication, Lau and Baltes (2010).

3.2 Motion Controller

Motion Controller is a program running in a PC to control the motion based on the vision input. Currently we use Absolute Position as the communication protocol which has advantage of simplicity in implementation compare to other protocols. The different protocols were discussed in another paper, Lau and Baltes (2010). Based on this protocol, the Motion Controller will send instructions to the CM-2+'s Control Program. For instance, we use a `SyncWrite` command to move the servos to specific positions synchronously. Table. 1 shows the structures of the Absolute Position protocol where `SyncWrite` commands were sent to move all servos to move from position 512 to 520 at the speed of 100. We divided the positions and speed by 4 so each of them can be wrapped into one byte in hexadecimal. According to Table. 1, each `SyncWrite` command of the Absolute Position protocol will send 14 bytes of instruction in one packet.

3.3 PID Controller

Proportional-Integral-Derivative, PID controller is the most popular industrial feedback control algorithm. Implementation of the PID controller in our Control Program is based on the feedback of latency and jitter from torque responses which will modify its context switching frequency. We need to ensure that our pre-emptive kernel is robust enough so that the Control Program can modify

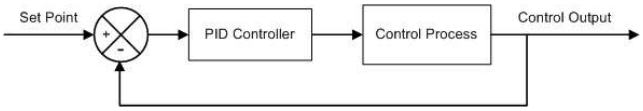

Fig. 4. Block diagram of PID controller

its context switching frequency in real-time. Fig. 4 shows the theory of the PID controller.

In the PID control loop, the control output (co) is calculated based on the following equations:

$$e(t) = sp - pv \tag{1}$$
$$P_{out} = K_P e(t) \tag{2}$$
$$I_{out} = K_I \sum_{t=0}^{n} (e(t)) \tag{3}$$
$$D_{out} = K_D[e(t) - e(t-1)] \tag{4}$$
$$co = P_{out} + I_{out} + D_{out} \tag{5}$$

Firstly, the error, $e(t)$ is calculated in equation (1) by subtraction of the set point, sp and the process variable, pv which is the latency of current run, t where $t = 0, 1, 2...n$. Then the proportional, integral, and derivative terms of output are determined by equations (2), (3) and (4). Finally, all PID outputs are summed to calculate co as equation (5). K_P, K_I and K_D are the constants gain of proportional, integral and derivative respectively. The PID controller will find the context switching time which has the lowest control output and it will be adjusted to reach the desired set point based on the latencies and jitters measurements.

4. IMPLEMENTATION

By solving Betty's kinematics problem with the Denavit-Hartenberg (DH) Convention in forward kinematics and geometric approach in inverse kinematics; it allows Betty to perform in three applications which are Wii drum kit, portrait drawing and simulation. Fig. 5 explains the coordinate frame system attached and assigned to each arm.

4.1 Forward Kinematics: DH-Convention

In the interest to calculate the global position of Betty's hands, we have to compute the forward kinematics. Although, a simple two-joint planar manipulator analysis is possible to solve with some simple transformation matrices. But the accumulated kinematics analysis of multiple-joint manipulator can be extremely complicated. Denavit and Hartenberg used screw theory in the 1950's to show that the most compact representation of a general transformation between two robot joints required four parameters. These are now known as the Denavit and Hartenberg convention (D-H convention) and they are the de facto standard for describing a robot's geometry, Hooper (2010a). Thus, this project applies DH-convention to solve the forward kinematics analysis problem because it introduces a simplified presentation to calculate the position and the orientation of the end effectors which is widely used in robotics and animation. In this convention, each homogeneous transformation, D_i is represented as a product of

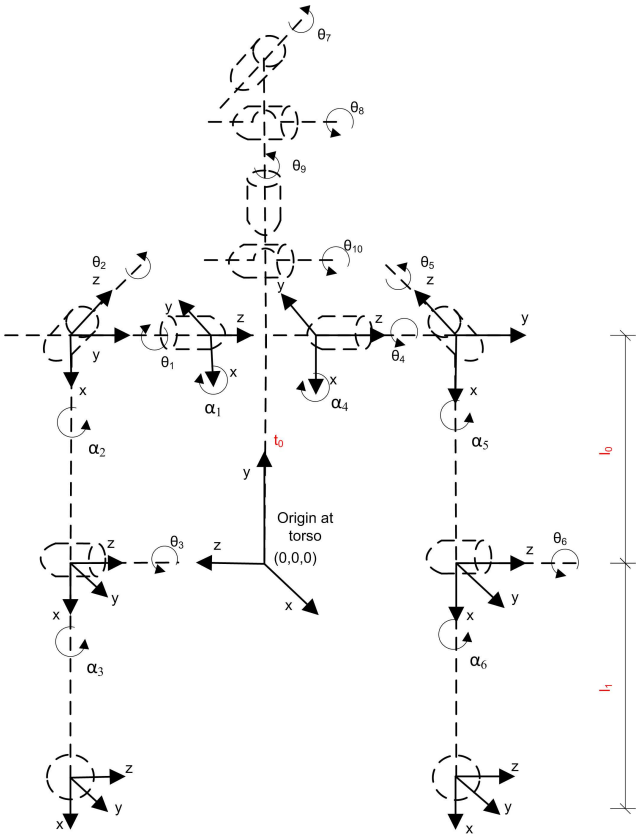

Fig. 5. Kinematic diagram of Betty's coordinate frame system

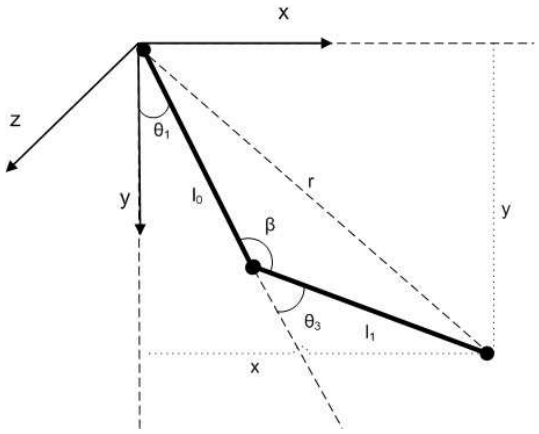

Fig. 6. XY-plane projection of Betty's right arm

four basic transformations matrices get the global position of any particular point at joint i, Spong and Vidyasagar (2006).

$$\begin{aligned} D_i &= R_{z,\theta_i} T_{z,d_i} T_{x,a_i} R_{x,\alpha_i} \\ &= \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & 0 \\ sin\theta_i & cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{i1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha_i & -sin\alpha_i & 0 \\ 0 & sin\alpha_i & cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{6}$$

Table 2. The DH-parameters for both arms

Joint,i	a_i	α_i	d_i	θ_i
1	0	α_1 =90	0	θ_1
2	l_0	α_2 =90	0	θ_2
3	l_1	α_3 =0	0	θ_3
4	0	α_4 =90	0	θ_4
5	l_0	α_5 =90	0	θ_5
6	l_1	α_6 =0	0	θ_6

In equation (6), R_{z,θ_i} is rotation matrix for angle $theta_i$ by z-axis, R_{x,α_i} is rotation matrix for angle $alpha_i$ by x-axis, T_{z,d_i} is transformation matrix for length di by z-axis and T_{x,a_i} , is transformation matrix for length a_i by x-axis. The detail descriptions of the DH-parameters are:
a_i - the distance between two joint axes measured along x-axis.
α_i - the relative twist between two joint axes measured about x-axis.
d_i - the distance between the two perpendiculars measured along the joint axis, z-axis
θ_i - joint angle about the z-axis

According to the coordinate frame system in Fig. 5, the DH-parameters are shown in Table 2.

To establish the homogeneous coordinate between origin and arms, both shoulders' coordinate frames are translated to the origin at Betty's torso with translation matrices, T_L and T_R for left and right shoulder respectively.

$$\text{Where, } T_L = \begin{bmatrix} 1 & 0 & 0 & t_0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -t_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and } T_R = \begin{bmatrix} 1 & 0 & 0 & t_0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Once the DH-parameters and relative frames are determined, we will combine the transformation matrices as the product of translation and DH-Convention matrices. The example shown in equation (7) and (8) is the calculation of the right hand's global position which denoted as P_R; based on the rotation angles of θ_1, θ_2 and θ_3 from the origin O with the transformation matrix M_R.

$$\begin{aligned} M_R &= T_R D_1 D_2 D_3 \\ &= \begin{bmatrix} 1 & 0 & 0 & t_0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta_2 & 0 & -sin\theta_2 & l_0 cos\theta_2 \\ sin\theta_2 & 0 & cos\theta_2 & l_0 sin\theta_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\quad \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & l_1 cos\theta_3 \\ sin\theta_3 & cos\theta_3 & 0 & l_1 sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{7}$$

$$P_R(x, y, z) = M_R O \tag{8}$$

4.2 Inverse Kinematics

Previously we discussed how to calculate the global position with DH-Convention based on the assigned servos' rotation angles. This section is concerned with the inverse method of finding each of the joints' angle with a given global position. For the implementation of Betty's hand positioning, we use the geometric approach for solving the inverse kinematics problem. The reason of using geometric approach is because the design of Betty's arms is geometrically simple. Therefore, it is easier to solve compare to general inverse kinematics problem, Spong and Vidyasagar (2006). Generally there are two solutions to a inverse kinematics problem, which could be ± 45 degrees and this multiple solutions scenario adds to the challenge of the inverse kinematics problem. However we can identify the correct solution with a trigonometric function called $atan2$ that will find the proper quadrant when given both the X and Y in a $theta = atan(Y/X)$ argument, Hooper (2010b). Based on Betty's coordinate frames system in Fig. 5 and Fig. 6 , the equations to calculate the servos' rotation angles for Betty's right arm are shown as:

$$\theta_2 = atan2(Z/Y) \tag{9}$$

$$r^2 = X^2 + Y^2 \tag{10}$$

$$\beta = acos((l_1^2 + l_2^2 - r^2)/(2l_1 l_2))...(Law\ of\ Cosines) \tag{11}$$

$$\theta_3 = \pi - \beta \tag{12}$$

$$\theta_1 = atan2(X/Y) - asin(l_2 * sin(beta)/\sqrt{r^2}) \tag{13}$$

4.3 Simulation

Simulation is a fundamental tool in humanoid robotics, fulfilling many important roles. For example, it is used to validate controllers with respect to safety and performance considerations; the controller development cycle of programming, testing and improvement is made more convenient through simulation, which should be easier, safer and quicker to execute than experiments on a real robot, Joshua et al. (2008).

This program illustrates the simulation model of Betty's upper body with interactive moving joints which will rotate according to nine joint angles. One of the main objectives of this project is to illustrate various features in OpenGL libraries which will demonstrate model rendering, environment setting, visual effects and interactive control capabilities. Another objective of this simulation program is to avoid any miss-behaviour and collision of the robot in the physical world control which will cause damage to the servos. Therefore, this project implements DH-convention, a forward kinematics analysis method to calculate the final position of the end effectors, which refer to Betty's hands. This approach will enable the collision detection between hands and obstacle based on the global positioning of hands and objects. Where is demonstrated with Betty's right hand and a wall. Figure 7 illustrate basic simulation model of Betty and warning detection when Betty's right hand collides with the wall by moving shoulder joint.

4.4 Drumming

This is a simple application to play Wii drum kit with Hydrogen (2010a) and Joy2Key (2010). Hydrogen is an advanced drum machine created by Alessandro Cominu.

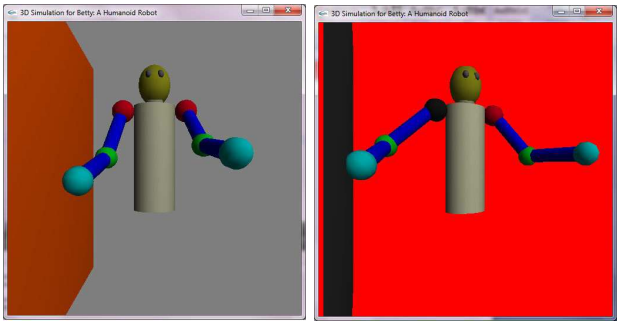

Fig. 7. Simulation with wall and hand collision warning

Fig. 8. Flow diagram of Betty's OpenCV image processing module.

Its main goal is to provide professional yet simple and intuitive pattern-based drum programming, Hydrogen (2010b). Joy2Key is a keyboard emulator for joysticks or gamepad. In this project it translates the drum pad input from Wii drum kit into equivalent keyboard input in Hydrogen. The output of this project was performing a series of drumming motion with Betty's inverse kinematics control on a Wii drum kit.

4.5 Portrait Drawing

The objective of this project is to recognise human faces and convert them to line art images which will be sketched by using Betty's inverse kinematics control. We convert the 2D drawing into a set of joint angles which run on the Betty's right arm. Flow diagram in Fig. 8 shows the general algorithm of image processing module in the Motion Controller. In this module, we implement the Haar Cascade Classifier (HCC) in OpenCV to perform face detection. This classifier uses an XML file in OpenCV root directory to classify the image. By default, OpenCV provides several pre-trained classifiers to serve different profiles such as frontal face, full body, lower body and upper body. This project only interested in face detection, so we use `haarcascade_frontalface_default.xml`, one of the four provided frontal face classifier in OpenCV, Gary and Adrian (2008). Based on one second detection interval, if three consecutive faces are detected, a grayscale image will be saved with `cvHaarDetectObjects` flag variable and `CV_HAAR_DO_CANNY_PRUNING` option.

`CV_HAAR_DO_CANNY_PRUNING` will enable HCC to skip image regions that are unlikely to contain a face, reducing computational overhead and possibly eliminating some false detection as discussed by Robin (2007). Then the module will run the Canny edge detection algorithm in OpenCV to convert a selected grayscale image to line art image by adjusting the low and high thresholds as shown in Fig. 9.

After the Canny image is generated successfully, each pixel will be mapped into 2D drawing space which is based on the XZ-plane of Betty's right arm coordinate frame. Fig. 10 illustrate the mapping of pixels from left to right and top to bottom. The mapping conversion is shown in equations (14) and (15). Where (x, z) and (x', z') devote

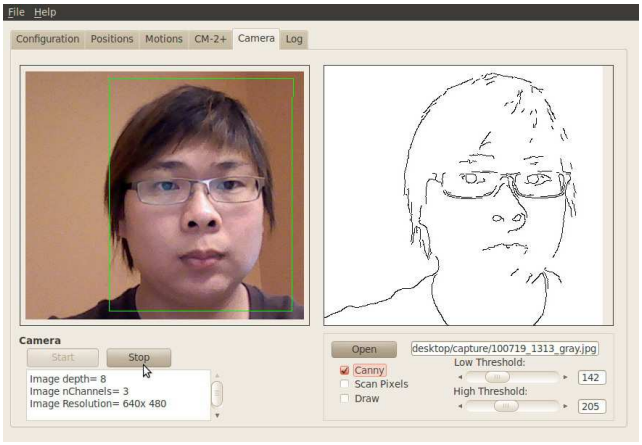

Fig. 9. OpenCV image processing module

Table 3. Number of pixels in pixel mapping images after pixel reduction

Pixel distance	0	2	5	10
Number of pixels	1502	1005	838	744

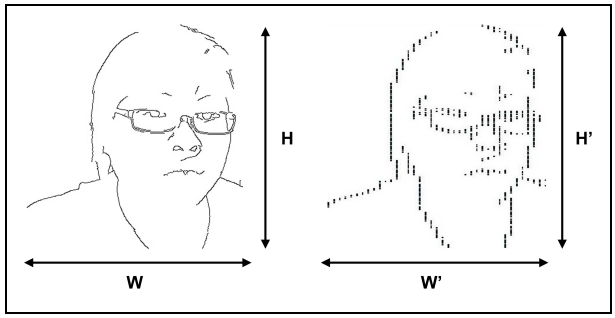

Fig. 10. Pixel mapping of drawing space image

respective coordinate on grayscale image and drawing space; x'_{start} is the starting of W' and z'_{end} is the ending of H'.

$$x' = \frac{x \times W'}{W} + x'_{start} \tag{14}$$

$$z' = \frac{z \times H'}{H} - z'_{end} \tag{15}$$

Initially the total number of pixels in a drawing space image is too large and it will take a great amount of time to sketch the portrait in a dot matrix approach. So we use a simple pixel reduction algorithm to reduce the number of pixels by considering the distance between pixels from left to right and top to bottom which represent by PD. Fig. 11 explains the algorithm which will measure the distance between pixel where it will omit not more than two pixels in consecutive pixels as seen in fourth row. Table 3 shows the differences between total number of pixels according to different PD values. Fig. 12 shows the comparison between original Canny image, pixel mapping images with different PD and actual drawing output when $PD = 10$ is selected.

5. CONCLUSION AND FUTURE RESEARCH

In this paper we demonstrate a cost-effective humanoid robot based on its design and development of hardware

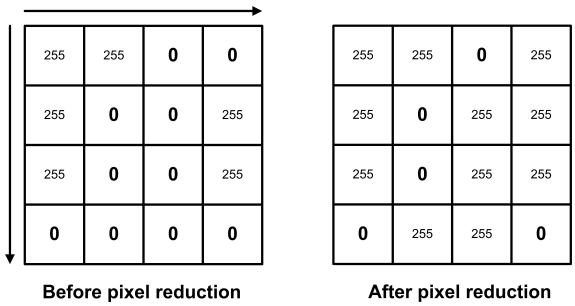

Fig. 11. Pixel reduction algorithm

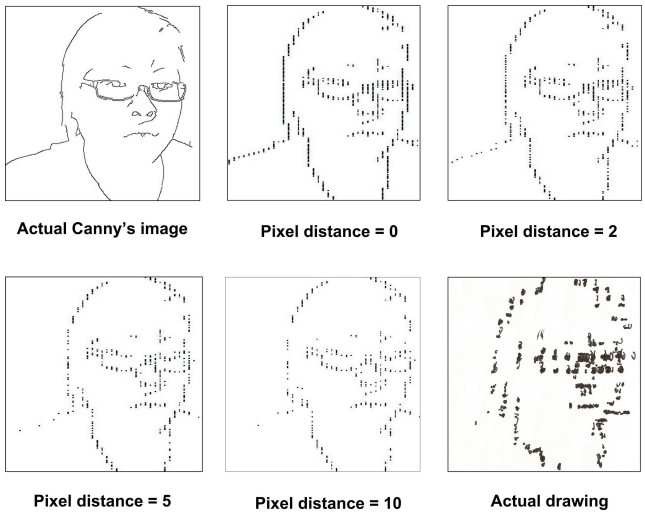

Fig. 12. Comparison between Canny image, different pixel distance and actual drawing output

and software. We also discussed three projects that implementing forward, inverse kinematics, PID and vision system. In future, we will work closely on active stereo vision processing, human-robot interaction and speech synthesis. We also plan to re-design Betty's arms by adding wrists which will significantly improve Betty's motion control and flexibility.

REFERENCES

Calinon, S., Epiney, J., and Billard, A. (2005). A humanoid robot drawing human portraits. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (HUMANOID 2005)*. IEEE-RAS, Tsukuba, Japan.

Chaumette, F. and Hutchinson, S. (2006). Visual servo control part i: Basic approach. volume 13, 82–90. IEEE, Brisbane, Queensland Australia.

Chaumette, F. and Hutchinson, S. (2007). Visual servo control part ii: Advanced approach. volume 14, 109 – 11. IEEE, Brisbane, Queensland Australia.

Gary, B. and Adrian, K. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., CA.

Hooper, R. (2010a). Learn about robots: Robot forward kinematics. `http://www.learnaboutrobots.com/forwardKinematics.htm`.

Hooper, R. (2010b). Learn about robots: Robot inverse kinematics. `http://www.learnaboutrobots.com/inverseKinematics.htm`.

Hydrogen (2010a). Advanced drum machine for gnu/linux. `http://www.hydrogen-music.org/?p=main`.

Hydrogen (2010b). Hydrogen (software). `http://en.wikipedia.org/wiki/Hydrogen_(software)`.

Jeff, F. (2010). Data structures and algorithms, part 2. `http://www.javaworld.com/javaworld/jw-06-2003/jw-0613-java101.html?page=7`.

Joshua, G.H., Benjamin, H., and Eduardo, M.M. (2008). Robot simulation, collisions and contacts. In *Proceedings of the 2008 International Conference on Intelligent Robots and Systems*. IEEE/RSJ, Nice, France.

Joy2Key (2010). Application: Joy2key. `http://www.linux.org/apps/AppId_1463.html`.

Lau, M.C. and Baltes, J. (2010). The real-time embedded system for a humanoid: Betty. In *Proceedings of the 13th FIRA Robot World Congress*, volume 103 of *Communications in Computer and Information Science*. Springer-Verlag.

Lin, C.Y., Chuang, L.W., and Mac, T.T. (2009). Human portrait generation system for robot arm drawing. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 1757–1762. IEEE, Singapore.

Lu, Y., Lam, J.H.M., and Yam, Y. (2009). Preliminary study on vision-based pen-and-ink drawing by a robotic manipulator. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 578–583. IEEE, Singapore.

Robin, W. (2007). Seeing with opencv. T & L Publications, Inc., CA.

Robotis (2006). Robotis: User's manual dynamixel ax-12.

Robotis (2007). Robotis: User's manual dynamixel rx-64.

Spong, M.W. and Vidyasagar, M. (2006). *Robot Dynamics And Control*. John Wiley Sons, Inc., New York.