Final Project Report - Stackbot - G1

Anitha Ganesha, Jovita Castelino, Qi Zhiyuan, Sanjay Dmello

March 20, 2014

Contents

1 Abstract

The StackBot is an object-identifying and stacking system modeled along the lines of an assembly line robotic arm stacking device. The system performs image processing functions, conveyor belt speed variance and robotic arm direction in real-time.

2 Objectives

- **Minimum Objective** Identifying objects of different shape moving on a conveyor belt. The objects will be placed on the conveyor belt in succession. Using the camera the shape of the object is determined by performing image processing on the acquired image. The different shapes to be determined could be circular, triangular and rectangular.

- **Target Objective** Stacking the objects according to shape detected. After identifying the shape of the object, the robotic arm moves to pick up the object from the conveyor belt. The robotic arm will have to move in such a way that it takes into account the speed at which the object approaches the arm. After picking up the object, it is placed in a location corresponding to a particular shapes stack.

- **Goal Objective** Objects not correctly aligned on the conveyor belt should also be detected and stacked. The shape of the objects will be determined regardless of the alignment of the object on the conveyor belt. They will then be stacked as per the target objective

3 High Level Design

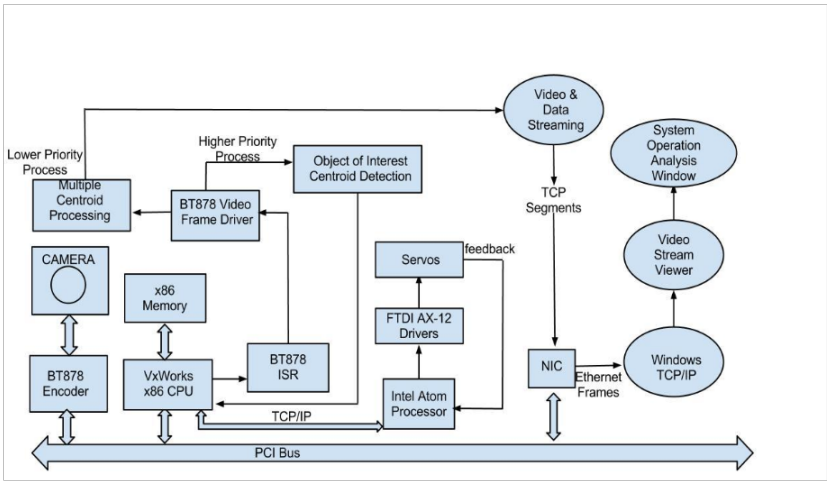

Figure 1: High Level Design

4 Hardware Based Low Level Design

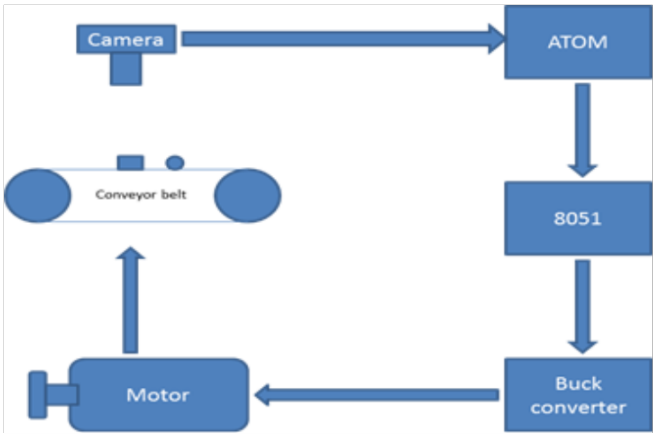

Figure 2: Conveyor Belt and its driving system

4.1 Conveyor Belt

The conveyor belt we are using is actually a homemade one. We use unistrut as the building material for the skeleton. Two cylindrical rollers with bearing served as the motorized wheels. A motor from an electrical screwdriver is used as power supply for the whole conveyor belt system since it can give out extraordinarily huge amount of torque while maintaining a relatively stable speed. Two alignment screws are used in connection with the gliding block, which serve as the load bearing blocks, for the purpose of tuning the belt in a very precise way.

4.2 PWM based voltage regulator

A PWM based Buck converter is used for tuning the speed of the conveyor belt. The PWM is generated from the pin of an 8051 and is controlled via serial signals through RS232 cable. The design of the Buck converter is as shown in thefigure . On the left side is the schematic; while on the right side is the actual picture.

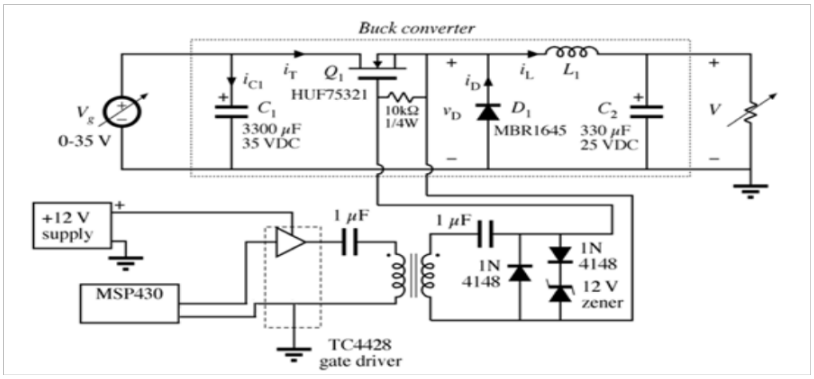

Figure 3: PWM based voltage regulator

-k/8	-k/8	-k/8
-k/8	k+1	-k/8
-k/8	-k/8	-k/8

Table 1: Edge Enhancement Kernel

5 Detailed Design

5.1 Image Processing

5.1.1 RGB to Grayscale

In order to convert the RGB image to grayscale, the luminance of the color image is used to match that of the grayscale image. The camera we used is based on NTSC standard and the luma component is computed as follows. $Y = 0.3R + 0.59G + 0.11B$ To detect the objects in the frame, the grayscale image was further enhanced used a point spread function (PSF). The image array is applied to the edge enhancement kernel, such that each pixel is replaced by sum of -k/8 its neighboring pixel and k+1 times itself.

5.1.2 Filtering and Centroid Detection

In order to find the centroids of the objects in a single frame of the image, first a fixed intensity threshold and a fixed size threshold is set. Pixels with intensity higher than the set threshold will be considered. The centroid for the objects is calculated via the following equations. $X_{centroid} = \frac{\sum x^i}{N}$ where i goes from 1 to N. $Y_{centroid} = \frac{\sum y^i}{N}$

5.2 Multiple Centroid Detection

For multiple centroid detection, we separate the field of view into dozens of strips that perpendicular to the conveyor belt moving direction. The strips with intensity greater than some threshold are considered. If the consecutive strips have value greater than threshold then they are considered as the same object. The intensity of the these consecutive objects are collected, and the centroid calculation is applied to these set of strips. The centroid computation of the new object is not started until a new strip which satisfies threshold is obtained.

5.2.1 Converting Pixels to Centimeters

The co-ordinates of the object detected are obtained in pixels. These are mapped to respective co-ordinates, to correspond to position in terms of the robotic arm movement. 3 steps are taken in order to achieve the same.

- Transformation from pixel to length (inch) $C = \frac{inch}{pixel}$

6 Robotic Arm Control

6.1 Inverse Kinematics

6.1.1 Forward Transformation

$y = l2sin\theta + l3sin(\theta_1 + \theta_2)$
$x = rcos\varphi$
$y = rsin\varphi$
$z = l_1 + l_2cos\theta_1 + l_3cos(\theta_1 + \theta_2)$

6.1.2 Inverse Transformation

$r = \sqrt{x^2 + y^2}$
$\varphi = tan^{-1}(\frac{y}{x})$
Assume $A = rl$
$B = (z - l_1)l_2$
$C = \frac{r_2+(z-l_1)_2+l_2^2-l_3^2}{2}$
$Asin\theta_1 + Bcos\theta_1 = C$

Hence inverse kinematics equations are as follows $\varphi = tan^{-1}(\frac{y}{x})$

$$\theta_1 = cos^{-1}\frac{BC + \sqrt{B^2C^2 - (C^2 - A^2(A^2 + B^2)}}{A^2 + B^2}$$

6.2 Stacking Mechanism

The stacking of the objects is performed by sending the robotic arm a sequence of pre-programmed steps for smooth motion and accurate stacking. The objects which are stacked are either circular or rectangular in shape. They are stacked accordingly in two different stacks.

6.3 Servo Control

The robotic arm consists of seven actuators(servos) which are moved by a controller loaded on the arm. This controller along with the inbuilt driver is referred to as the Dynamixel. A device called the USB2Dynamixel is used to connect the robotic arm to the Atom board. The control software loaded

on the Atom controls the Robotic arm by transmitting data in packets over the serial USB2Dynamixel link.

6.4 Arm Positioning

The arms destination position is calculated using the inverse() function which implements the inverse kinematics equations to calculate the three angles theta1, theta2 and phi. The inputs to this function are the x,y,z co-ordinates of the object which is moving on the conveyor belt. The z co-ordinate is a constant value and is received from the VxWorks target over a socket connection.

7 Results

7.1 Scheduling

Our System has overall 6 tasks with their respective priorities as listed in the below table.

No	Tasks	Priority
1	btvid	10
2	computeCentroid	42
3	sercomm and send2Atom	47
4	streamclient	54
5	recvFrmAtom	55

Btvid task is the highest priority task which is responsible to save the frame to a save buffer. This is pended until the PCI_INTA_ISR releases the semaphore when the frame is captured every 30ms. This then releases the semaphore of the computeCentroid task. The priority of the btvid is set too high so that it is not interrupted by the other system tasks such as tnet task.

The computeCentroid task is the second highest priority task that gets called which performs several image processing tasks. The various functions that gets executed is as listed below.

No	Image processing functions
1	rgbToGray()
2	sharpenGS()
3	multipleCentroid
4	PixtoRobotic

Once these functions are executed, the serial communication semaphore is released.

The next task that gets executed is the serComm, this first checks if the arm is ready to pick a new objects, if yes it then checks if there is a centroid for a new object available with the arms field of view. If available , it then checks if it is a valid centroid value(non negative) and then checks if it is a valid object.If both the conditions satisfied, it then sends these values to the arm and the speed variable is made to slow.If the arm is not ready and if the previous speed is slow, retain its value as slow for 2seconds until the arm picks up the object.The speed value is then later sent to the 8051

board serially. After all the computation it releases the semaphore of the stream_client tasks.

The stream client task transferred the frame with the multiple centroid marker to the host which is displayed via vpipe display.

The recvFrmAtom task is the lowest priority task which gets executed only in the slack time of 30ms after all the task is done. It listens to the acknowledgement from the arm and updates the msgAtom variable which is used in the third task .

The figure below gives an overview of the synchronization mechanism between different tasks using semaphores.

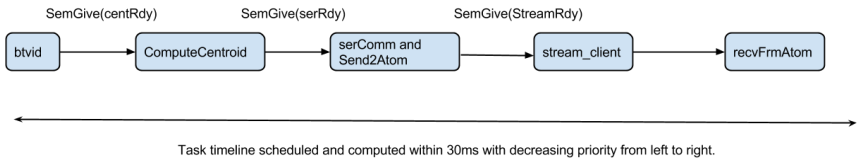

Figure 4: Synchronization of all Tasks

8 Challenges faced

8.1 Multiple centroid Detection

The main challenge here is identify all the objects present in the frame. If the objects are too close to each other then they identified as a single component. This can be solved using connected component labelling algorithm. But this algorithm takes more than 30ms time for processing. Therefore, we used a simpler algorithm to divide the frames to vertical strips and then process.This has a limitation of not identifying objects in the same vertical strip.

8.2 Serial Communication

A serial communication link between the VxWorks target and the 8051 was established with a baud rate of 9600. The 8051 required the serial link to receive signals from the target for conveyor belt speed control. The main problem faced here was in identifying the serial communication tasks priority level as a bottleneck which prevented it from executing on time due to preemption by the system task tNetTask. While the serial communication tasks priority level was lower than that of tNetTask the speed control mechanism would work erroneously.

8.3 Correction mechanism for missed objects

The robotic arm would miss picking objects either due to incorrect picking position prediction of the moving object or by losing its grip on the object while grabbing it. To implement a control mechanism which would let the arm know that it missed the object, the position of the gripper was read and if the grippers value was above a certain threshold, which indicated the absence of an object in between the grippers fingers, a pickup retry command was issued.

9 Conclusion and Future Scope

With the completion of the project the goals set in the initial stages of the project were realized. These goals were shape detection, object stacking, accurate pickups of the object regardless of its alignment on the belt. The ambitious goal of sending the objects at variable speeds to enhance the pickup rate of the robotic arm was also realized. There are many improvements that we would have liked to work on if there wasnt a time constraint. Firstly, the robotic arm control system could be made more robust to pick up objects by placing the gripper at an angle most suited to grabbing relative to the objects alignment, thus reducing the occurrence of slips in grab attempts. The shape detection algorithm could be enhanced to detect more shapes than just two. Also, differently colored objects could be identified and classified for better separation.