Description of Project3

Group Charlie: Object Tracking
Andrew Case, Scott Lebow, Peishi Yuan

1 Goal

Our goal for this project is to have Charlie detect an object, orient itself so that it is facing the object directly, then walk towards the object. We decided to have Charlie turn to correct direction firstly and then walk forward to simplify our code. It also gives Charlie the ability to be more predictable. If being used as a security robot, it would be easier to tell where Charlie last saw an object.

2 Approach

We used the Pixy Camera for our object detection. The Pixy Camera, on its own, is able to detect an object it sees existence, size and location. It then relays that information to the Raspberry Pi 2. A program on the Raspberry Pi 2 then tells Charlie scan its head for the object. When Charlie sees and object, the programs instructs it to turn to the left or the right until the object is in the center of its vision. When the object is in the center of the vision, it then walks towards the object, then stops to look for the object again.

This was done using a state machine similar to that used by the mario bots. The robot starts in an idle state where it crouches for a bit before standing up to begin its search. First it checks its center field of vision for the object it is looking for. It does this by calling the pixy camera api function getBlocks(). If it doesn't find the object it looks to the left and checks for the object again. If it doesn't see the object on its left it turns its head right. If the object is still not found it goes back to the idle state to start the process again. Changing poses between states was accomplished by playing pages in the robot motion editor. If an object is found it will turn left, turn right, or walk a little bit forward depending on where the object was found.

3 Problems Addressed

We had issues trying to compile C++ code with both the Pixy and RME APIs running together. We solved this by switching to Python as our language. Additionally we had to move the api used to control the robot into the same directory as that used by a python wrapper of the pixy camera. Most of the problems we encountered were making sure all files and modules needed could be found at execution.

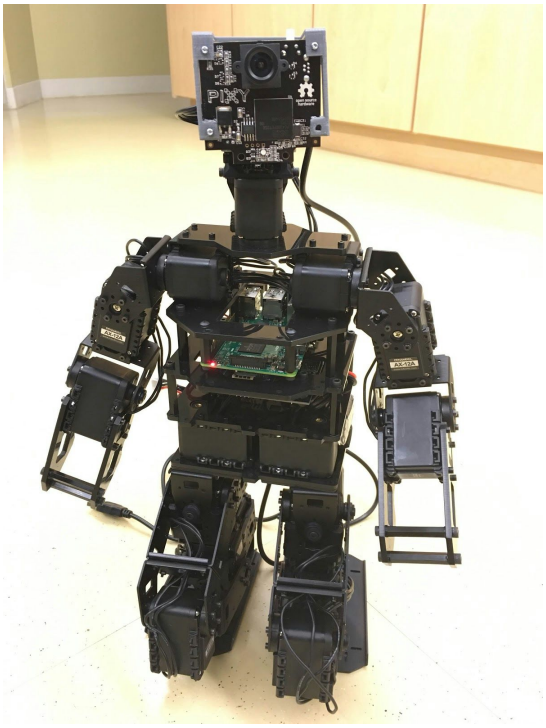
4 Problems Not Addressed

The Pixy Camera is very particular about lighting levels in a room when trying to see objects so we had to pick specific objects to track. This could be solved with color codes, but we did not implement this. The RME is also finicky about when it will work. Another thing that would have made our implementation better would been tuning the walk cycle of the robot. The robot tends to fall down often especially when walking or transitioning between standing and walking.

If we were to make a more developed version of a security bot, we would make the transitions between states more fluid and have the robot adjust its walking trajectory more often without pausing while it can still see the object. The reason we didn't do that this time was because we wanted to be safe and avoid have the robot fall by walking unpredictably. This could be accomplished by processing the data returned by the pixy camera more often and paying more attention to the angle that the object is in regards to the robot. If walking was tuned to be less unstable this approach would make the robot appear more human when following the object.

5 Steps to Install

To get this project working you will need to clone the git repository https://github.com/Anmar1985/Human_Robots_Interaction_Fall15/wiki and follow the instructions in order to get an python api wrapper for the human robot functions. After this you should install the pixy camera software and follow the tutorial that they provide in order to get a basic demo running. This will allow you to verify the camera is installed properly. Next you need to clone the following git repository in order to get the pixy api calls accessible in python. The url for the repository is https://github.com/charmedlabs/pixy/tree/master/src/host/libpixyusb_swig . This is actually not the url you want to use when cloning but the url does show the directory that contains the pixy api. The file get_blocks.py shows how to use the api. Next move the human robot api that was created by following the steps in human robot git repository wiki into the directory shown above. This will allow you to call methods from both apis. The code we used to run the security bot is shown below.



6 Code

(1) helloRobot.py

```
#!/usr/bin/python3
import ctypes
import api
import os
import time
import sys
import struct
#test
from pixy import *
from ctypes import *
from StateMachine1 import *

def Main():
    #api.ServoShutdown()
    try:
        if api.Initialize():
            print("Initalized")
        else:
            print("Intialization failed")
        #api.ServoShutdown()
        api.PlayAction(15)
        #api.PlayAction(8)
        #value = int(input("Turn head to"))
        #api.SetMotorValue(20, value)
        print("Pixy Python -- Get Blocks")
        pixy_init()
        #Run()
        CurrentState = State.Idle
        while(True):
            #proceed = int(input("1 for next step"))
            CurrentState = ManageState(CurrentState)
            #pass
    except (KeyboardInterrupt):
        api.ServoShutdown()
        sys.exit()
    except():
        api.ServoShutdown()
        sys.exit()

def Run():
    command=1
    #api.Walk(True)
    #print("Running...")
    #move foward
    if(command == 1):
        api.PlayAction(15)
        api.PlayAction(8)
        #api.WalkMove(0)
```

```
            #api.Walk(True)
            #api.WalkMove(20)
        #move back
        elif(command == 7):
            print("supposed to move backwards")
        #move right
        elif(command == 8):
            api.WalkMove(0)
            api.WalkTurn(20)
        #move left
        elif(command == 9):
            api.WalkMove(0)
            api.WalkTurn(-20)
        elif(command == 5):
            api.WalkMove(0)
        Run()

if __name__ == "__main__":
  Main()
```

(2) StateMachine1.py

```
#!/usr/bin/python3
import ctypes
import api
import os
import time
import sys
import struct
#from enum import Enum
from pixy import *
from ctypes import *

class State:
        Idle,TurningLeft,TurningRight,MovingForward,BeginSearch,LookLeft,LookRight = range(7)

class SearchResult:
        ObjectLeft,ObjectCenter,ObjectRight,NoObject = range(4)

class Blocks (Structure):
  _fields_ = [ ("type", c_uint),
               ("signature", c_uint),
               ("x", c_uint),
               ("y", c_uint),
               ("width", c_uint),
               ("height", c_uint),
               ("angle", c_uint) ]

blocks = BlockArray(100)

#system configuration parameters
```

```
leftBound = 120
rightBound = 180

def delayMS(delayAmmount):
	time.sleep(delayAmmount/1000)

def checkObject():
	#search for objects and if there is an object
	count = pixy_get_blocks(100,blocks)
	if count > 0:
		return True
	return False

def checkLeftRightMiddle():
	#check to see if object is in left right or middle if
	count = pixy_get_blocks(100,blocks)
	maxIndex= 0 #index of maximum sized object
	max = 0
	if count == 0:
		return SearchResult.NoObject
	for index in range (0, count):
		area = blocks[index].width * blocks[index].height
		if area > max:
			max = area
			maxIndex = index
	print("X Reading: " + str(blocks[maxIndex].x))
	if (blocks[maxIndex].x < leftBound):
		return SearchResult.ObjectLeft
	elif (blocks[maxIndex].x < rightBound):
		return SearchResult.ObjectCenter
	else:
		return SearchResult.ObjectRight


#performs actions needed by state machine
#returns the Updated state
def ManageState(CurrentState):
	if(int(CurrentState) == State.Idle): #MOSTLY COMPLETE(add wait later)
		api.PlayAction(15)
		print('Sit');
		#delay for a set ammount of time before beginning next search
		#either use RME or delay function
		delayMS(2000)
		api.PlayAction(8)
		print('Standing to start search');
		return State.BeginSearch
	elif(int(CurrentState) == State.TurningLeft):  #INCOMPLETE (Delay Required)
		#move to the left a bit
		api.Walk(True)
		api.WalkMove(0)
```

```
            api.WalkTurn(20)
            delayMS(1000)
            print("Turn Left")
            return State.BeginSearch #for now go back to being search
        elif(int(CurrentState) == State.TurningRight): #INCOMPLETE (Delay Required)
            #move to the right a bit
            api.Walk(True)
            api.WalkMove(0)
            api.WalkTurn(-20)
            delayMS(1000)
            print("Turn Right")
            return State.BeginSearch #for now go back to being search
        elif(int(CurrentState) == State.MovingForward): #INCOMPLETE (Delay Required)
            #move forward a little bit
            print("Moving Forward")
            api.Walk(True)
            api.WalkMove(20)
            api.WalkTurn(0)
            delayMS(1000)
            print("Moving Forward")
            return State.BeginSearch #for now go back to being search
        elif(int(CurrentState) == State.BeginSearch):  #This state will cause robot to begin moving or move its
head to look more   #FINISHED
            api.WalkMove(0)
            api.WalkTurn(0)
            api.Walk(False) #turn of Walking before beginning search
            #check the camera for object using search criteria
            delayMS(500)
            print("Checking Front View")
            Result = checkLeftRightMiddle()
            if Result == SearchResult.NoObject:
                return State.LookLeft
            if Result == SearchResult.ObjectLeft:
                return State.TurningLeft
            elif Result == SearchResult.ObjectCenter:
                return State.MovingForward
            else:
                return State.TurningRight

        elif(int(CurrentState) == State.LookLeft): #INCOMPLETE
            #play action where left arm raised

            #play action where head looks left
            print("Checking Left")
            api.PlayAction(44)
            #scan for objects
            objectFound = checkObject()
            #play standing action to return the head to center
            api.PlayAction(8)
            #if object found move head back and go to state turningLeft else go to LookRight
```

```
        if(objectFound):
            return State.TurningLeft
        else:
            return State.LookRight

    elif(int(CurrentState) == State.LookRight): #INCOMPLETE
        #play action where right arm raised

        #play action where head looks right
        print("Checking Right")
        api.PlayAction(45)
        #scan for objects
        objectFound = checkObject()
        #play standing action to return the head to center
        api.PlayAction(8)
        #if object found move head back and go to state TurningRight else go to Idle state
        if(objectFound):
            return State.TurningRight
        else:
            return State.Idle
    else:
        pass  #This line should never be hit
```