

Team Elvis - 4x4 Tic Tac Toe

Phong Nguyen Rakhee Bhojakar

Portland State University Robotics II

## Table of Contents:

Objective Before we start ROS Intro What is ROS?

Setup ROS Kinetic packages Block Diagram:

ROS Networking Nodes

Text to Speech/Speech to Text Computer Vision

Robot Arm Control (Servo control) Artificial Intelligence:

Challenges Media Links

Code Reference Team Contribution

# Objective

1. Robot must have speech capabilities: word or sentence recognition and text to speech.
2. Robot must use some type of Machine Learning, Evolutionary algorithm, neural net or fuzzy logic. Take it from previous projects and integrate.

# Before we start

Throughout the project, we have learned and worked with the ROS programming to control many features.

# ROS Intro

I recommend reading at least the first five pages of Gentle introduction to ROS, digital version available [here](http://wiki.ros.org/ROS/Installation). Also, check out the ROS wiki [here](http://wiki.ros.org/ROS/Introduction#What_is_ROS.3F). Just remember to not worry yet if the next two sections mean nothing to you right now; I will try to explain why they are relevant in the following sections. Also, they will be covered again in class along with other topics in greater detail. If you need more help you can also use the reference boot left in the lab.

# What is ROS?

From the ROS wiki: ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly used functionality, message passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server. These are explained in greater detail in our Conceptual Overview.

# Setup ROS Kinetic packages

First, We are going to install ROS kinetic so we will be using Ubuntu 16.04 LTS. This version of ubuntu is compatible with ROS kinetic, anything newer or older version of Ubuntu may not work.

$ git clone git clon[e](https://github.com/Phasor2/ROS_and_Turtlebot2_PKG/) <https://github.com/Phasor2/ROS_and_Turtlebot2_PKG/>

$ cd ROS\_and\_Turtlebot2\_PKG

$ chmod +x ros\_turtlebot2\_pkg.sh

$ ./ros\_turtlebot2\_pkg.sh

and follow the instruction in the shell. It took around 15 minutes on virtual machine.

1. To install ROS Kinetic on Ubuntu 16.04 run command:

$ sudo apt-get install ros-kinetic-desktop-full

Rosdep for dependencies sudo rosdep init

rosdep update

$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc

$ source ~/.bashrc

1. When we are done installing all of the required packages for ROS, we start creating packages.

Create a workspace

$ mkdir -p ~/catkin\_ws/src

$ cd ~/catkin\_ws/

$ catkin\_make

Make sure ROS\_PACKAGE\_PATH environment variable includes the directory you're in.

$ echo $ROS\_PACKAGE\_PATH Create a package

$ cd ~/catkin\_ws/src

$ catkin\_create\_pkg beginner\_tutorials std\_msgs rospy roscpp

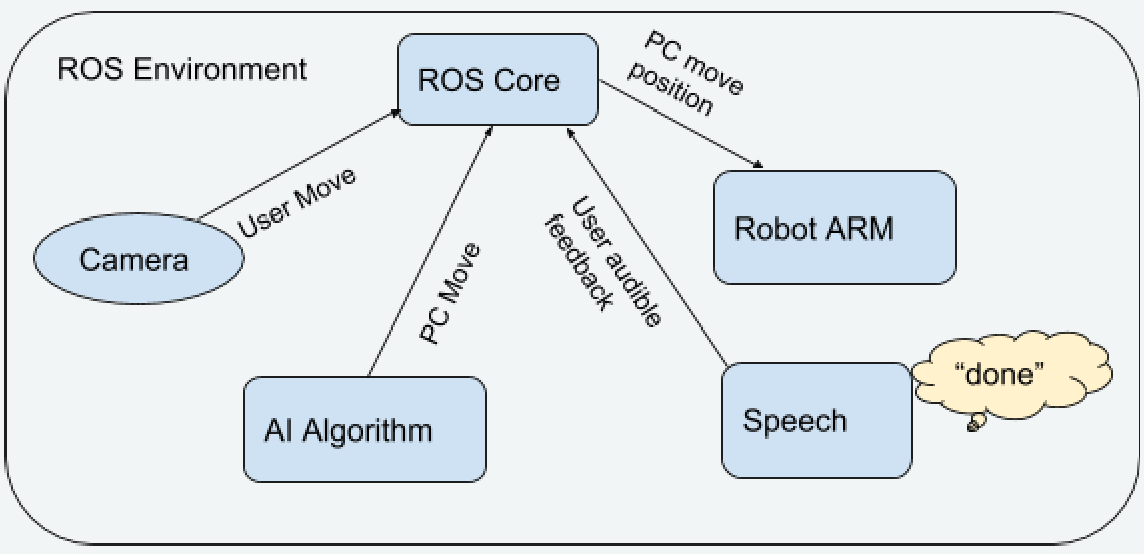
$ cd ~/catkin\_ws

$ catkin\_make

To add the workspace to your ROS environment you need to source the generated setup file:

$ ~/catkin\_ws/devel/setup.bash

### Block Diagram:



**System Algorithm:**

Speech\_recognition node:

Init node speech\_recognition Listen to “done” from user Publish 1 in to speech\_done topic

Later: Listen to win topic play win sound

Camera node:

Init node Object\_detect Listen to speech\_done If speech\_done is 1:

then start to update status of board and coordinate. Publish board\_status topic to AI node

Publish coordinate topic to arm node

AI node:

Init node AI\_TIC\_TAC Listen to board\_status

Do computation output the next pos topic Publish pos topic to arm node

Publish win topic

Arm node:

Init node Arm

Listen to coordinate topic where to pick up Listen to pos topic where to drop off Publish Arm\_done topic.

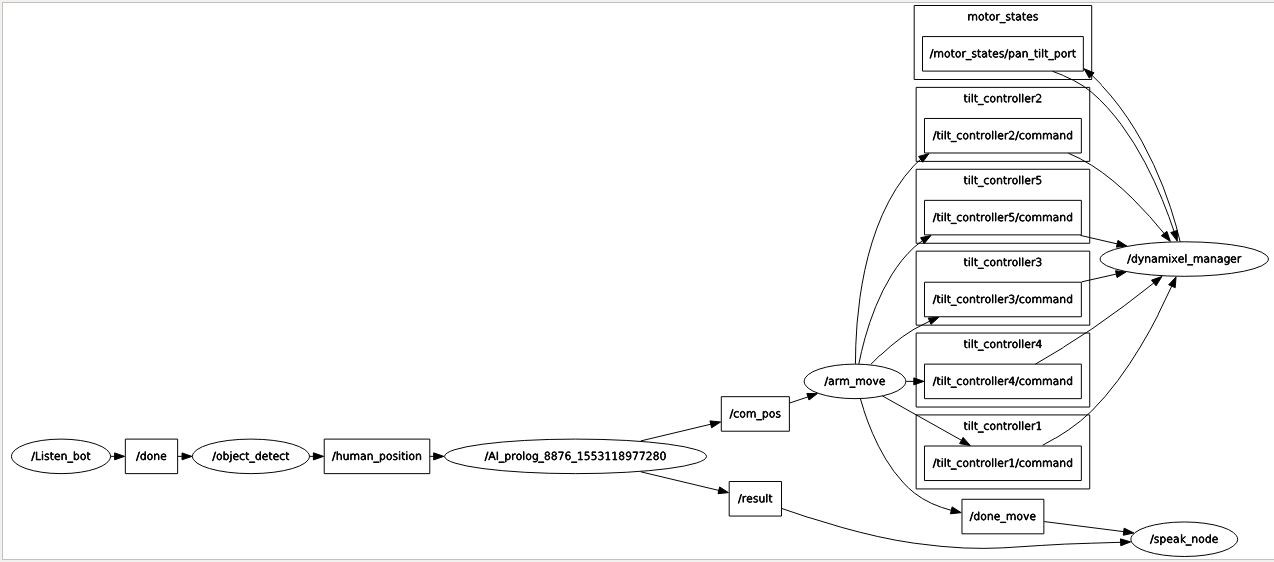
Speak node:

Init speak node

Play an audio file when arm node is done

Play ending audio file when result from AI is winning or losing

### ROS Networking

Nodes

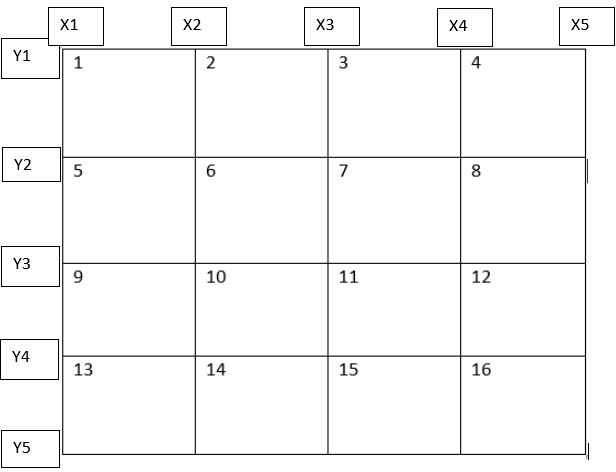
### Text to Speech/Speech to Text

Listen\_bot: just simply listen to keyword ‘done’ and sending 1 to /done topic and subscriber is object\_detect.

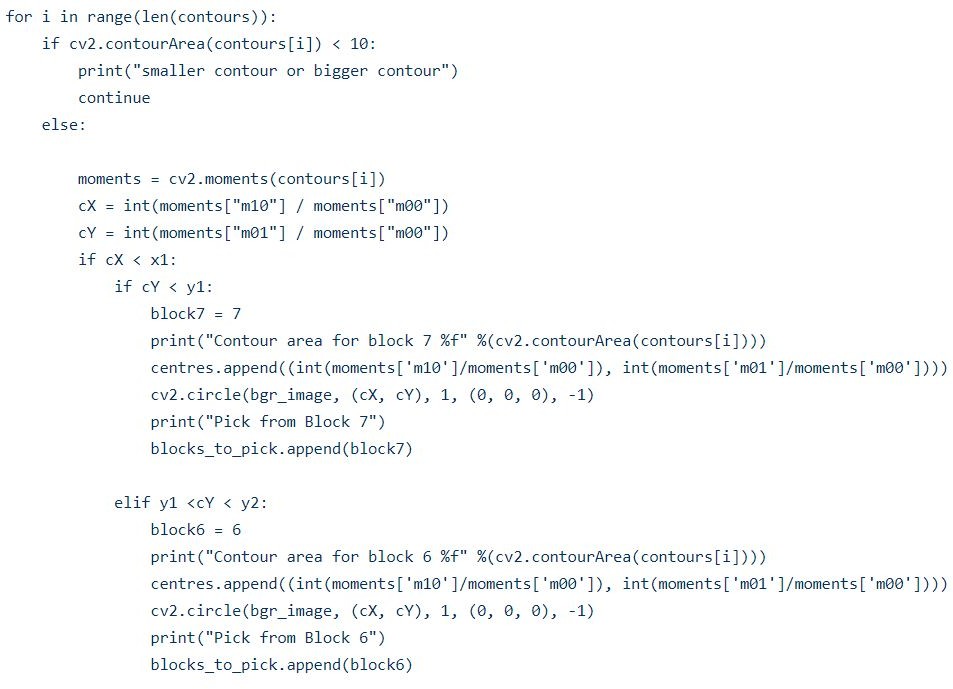
Speak\_node: play an random audio file after the arm is moving or result win, lose, and tie from AI node.

### Computer Vision

Opencv uses color masking and morphology to detect mask (color) positions on the image of 4x4 tic tac toe board. It uses gaussian blur preprocessing and color based detection algorithm. After capturing image it divides it into 16 region of interest and set 16 flags to detect green colored pawn on particular region. At a time it detects only one region out of 16 and sends the block number ranging from 1 to 16. It finds the center of detected contour and checks whether it lies between the coordinates of blocks.



Above figure shows how tic tac toe board is divided into 16 regions. And depending on where the green colored pawn is detected algorithm outputs the block number by comparing the center of detection with x and y coordinates. Below code snippet shows how this algorithm works:



If moments of shape detected cY occurs between y1 and y2 and cX is less than x1, algorithm outputs block number 7. Similar way it sets 16 flags for 16 blocks by comparing centers to x and y coordinates as shown is above figure.

Algorithm:

* 1. Divide tic tac toe board into 16 blocks
  2. Set x1,x2,x3,x4,x5 and y1,y2,y3,y4,y5 coordinates for comparison of center of moments of detected green pawns
  3. There are 16 blocks and 16 comparisons for get block number.
  4. If cX lies between x1 and x2 and cY lies between y1 and y2 the resulting block number will be 1.
  5. Green color detection is applied on the image and block number is selected and published on human position node.

During this process the lighting problem occurred every time we start the game due to variable lighting condition. So we set the fixed color and lighting condition while capturing the image.

### Robot Arm Control (Servo control)

There are 5 servos for the arm. The board has a total of 16 blocks. The arm can reach any of these block.

Arm\_move node: node that controls servos. The node consists of functions that create motions for the arms. Such functions are drop, pick up, default,etc… these functions are sending a map values servo to individual servos topics. After the AI to sending the coordinate through /com\_pos

,Arm\_move subscribe to /com\_pos, the arm automatically picks adjacent pieces from its tray to make a drop to that coordinate.

### Artificial Intelligence:

Artificial Intelligence is a computer program that intelligently responds to the user move.

Algorithm:

# Step 1: Wait for User Move

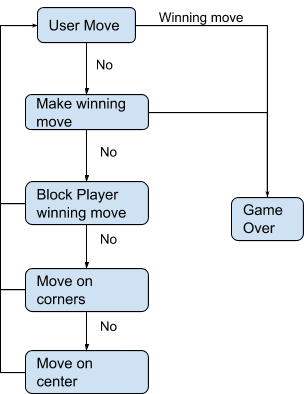
# Step 2:Check is there is free space on board.

#Step 3: Check if comp has winning move → yes → Winning move

# Step 4:If no → check if user has a winning move → yes → Block the user winning

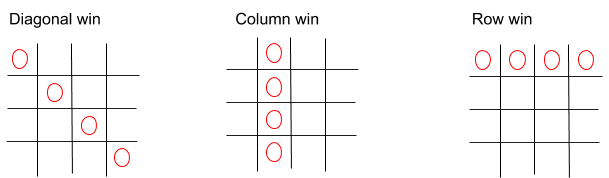
move

#Step 5: If no → generate a random position from the free space. # Repeat Step 1.



### Expert system Explanation:

1. Rules/predicates for winning:

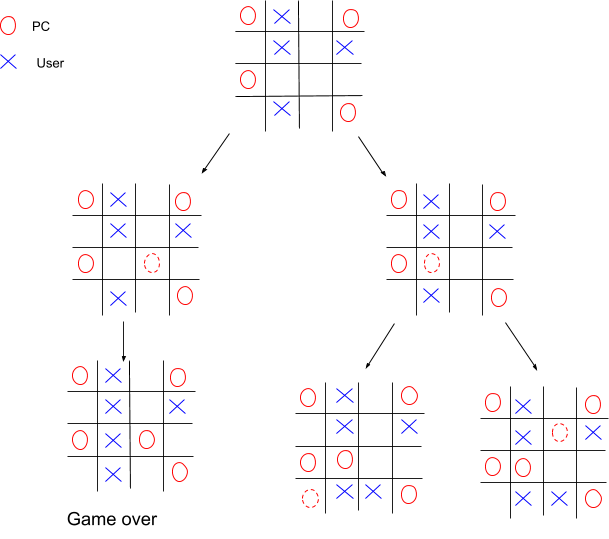


move.

Strategy behind checking all the possible moves to decide which is the next best

The search strategy used is **Breadth-First Search**, starts by considering all possible

moves (positions) that PC can take on the tic tac toe board. The below example demonstrates the breadth first tree search (expert system of prediction of next move) for a 4\*4 Tic Tac toe



### Challenges

* Parsing values between PROLOG and Python code:
  + Support for the pyswip module has not been updated in last 5 years and the present module available has many open bugs. The major issue was passing the values between python and the prolog. All the available examples were of 1 time interaction between prolog and python. Since our game is a user interactive session - user input - pc move - user input, we have to have a stream of data that passes between the user and the pc i.e between prolog and python and online available resources for this was very limited.
* Imitating human like hand movements for robot
* Camera vision performance in various light setting
* Lighting of camera has to adjust everytime we set the game so the code to mitigate the lighting effect of camera.
* Instead of manual setting of coordinates we can implement outer square detection and divide square into 16 blocks and calculate the block numbers for green pawn detection.

### Media Links

<https://github.com/Phasor2/Robot-4x4-Tic-tac-toe>

### Code Reference

<https://inventwithpython.com/chapter10.html> <https://codereview.stackexchange.com/questions/108738/python-tic-tac-toe-game> <https://github.com/aroques/numerical-tic-tac-toe>

[https://www.learnopencv.com/invisibility-cloak-using-color-detection-and-segmentation-wit](https://www.learnopencv.com/invisibility-cloak-using-color-detection-and-segmentation-with-opencv/) [h-opencv/](https://www.learnopencv.com/invisibility-cloak-using-color-detection-and-segmentation-with-opencv/)

<https://www.pyimagesearch.com/2014/08/04/opencv-python-color-detection/> <http://wiki.ros.org/rosserial_arduino/Tutorials/Servo%20Controller>

### Team Contribution

Phong - Speech and Arm Servo nodes, AI for the game node

Rakhee - Vision and Image processing node