BAXTER MATH PROBLEM SOLVER

Aditya Batheja, Somanshu Singh

CS 5335: Robotics Science and Systems

Northeastern University

ABSTRACT

"I prefer the pen. There is something elemental about the glide and flow of nib and ink on paper."

— James Robertson

Writing with a pen is an organic and tactile experience. Humans have been writing using a pen for centuries. We have come to a point that we hardly ever think about the dexterity required to write beautifully. Through this project we try to get Baxter to write just like we do. It recognizes the math equation written on a whiteboard. The digits in the equation placed in front of it are recognized using deep learning and the answer is calculated based on the equation. Baxter then writes the answer to the problem on the whiteboard.

INTRODUCTION

While coming up with a project idea we were convinced we wanted to build something that was exciting and emulated a task we do routinely enough, we can do with our eyes closed. It was when we were looking for a pen to write down the prospective ideas that it struck us. We would use Baxter to see, calculate and write the answers to basic math equations. The project was a good fit considering the course used manipulator robots extensively as a means of explaining robotics concepts and gave us the opportunity to use concepts learnt in other courses as well.

What we intend to achieve from this project: Use the mounted head camera to take images. Make sense of the pixels by processing the the pixel values to isolate the relevant details. Calculate the result for the given expression. Use inverse kinematics using the data points provided for digit representation to draw them on the whiteboard in front. Implementation of this idea presents interesting challenges and opportunities. Using Baxter we have built a better understanding of sensory data, control systems and image processing.

DESIGN CHOICES

- We chose to use ROS(Robot Operating System) because its officially supported by Baxter and has many libraries to interact with Baxter.
- We chose Gazebo to run simulations because that's the default Baxter simulator and custom models for numbers could be built and placed in the environment
- Moveit is used for motion planning and Forward and Inverse Kinematics calculations. Moveit is a better choice over the Ikservices provided by ROS as it allows for more control and is more robust.
- Convolutional Neural Networks are used for equation recognition in general. CNN gives a high accuracy in image recognition tasks.

IMPLEMENTATION

Gazebo does not have a model that simulates a pen or any kind of writing material. For this purpose models(.sdf files) representing handwritten numbers were custom built on sketchup. Sketchup is a 3d modelling platform that can be used to build custom files.

After setting up the environment and the equation on the table the next step is taking a picture. This is done by interfacing with baxter using ROS. A publisher subscriber model is used to get the image and opencv is used to convert the ros message to jpeg file.

The image is then sent for processing to the image processing module. This module is used to process the image of the equation that we get from the headcam of baxter. As we get the image of the whole whiteboard on which the equation is written, we need to extract the equation from it. To achieve this we first convert the image to grayscale. This reduces the redundancy of color feature in our case. Different colors should not interfere with how the expression is classified by CNN. And then crop it to the height and width of the equation.

To solve the equation we also need to retrieve the individual digits and the operations that are part of the equation.To obtain these images, image segmentation based on the background colors is used where we compare the pixel color of the characters to the background color to get images of each character.

The images are also resized so that they can be used by our maths symbol recognizer to recognise the characters. This is done using the results section contains the **data** collected during experimentation. The results section is the heart of a scientific paper. In this section, much of the important information may be in the form of tables or graphs. When reading this section, do not readily accept an author's statements about the results. Rather, carefully analyze the raw data in tables and figures to draw your own conclusions.

Next we want to classify the data. We have used convolutional neural network for this because it has very good accuracy for image recognition tasks. The layers and the parameters used for our model is described below.

Layers	Filter Size	Num of filters	Stride Size
Conv1	5 x 5	6	1 x 1
Conv2	5 x 5	16	1 x 1

Convolutional Layers

Layers	Inputs	Outputs
Full1	1296	400
Full2	400	120
Full3	120	84
Logits(Softmax Layer)	84	13

Fully Connected Layers

All the above mentioned layers other than Logits layer used the Relu activation function. Outputs from Conv1 and Conv2 layers are squashed using max pooling layer of ksize of 2 x 2 and stride of 2 x 2. Fully connected layers use dropouts as a method for regularization, where dropout rate of 0.75 is used. Adam optimizer was used for the optimization of the network.

Model Training:

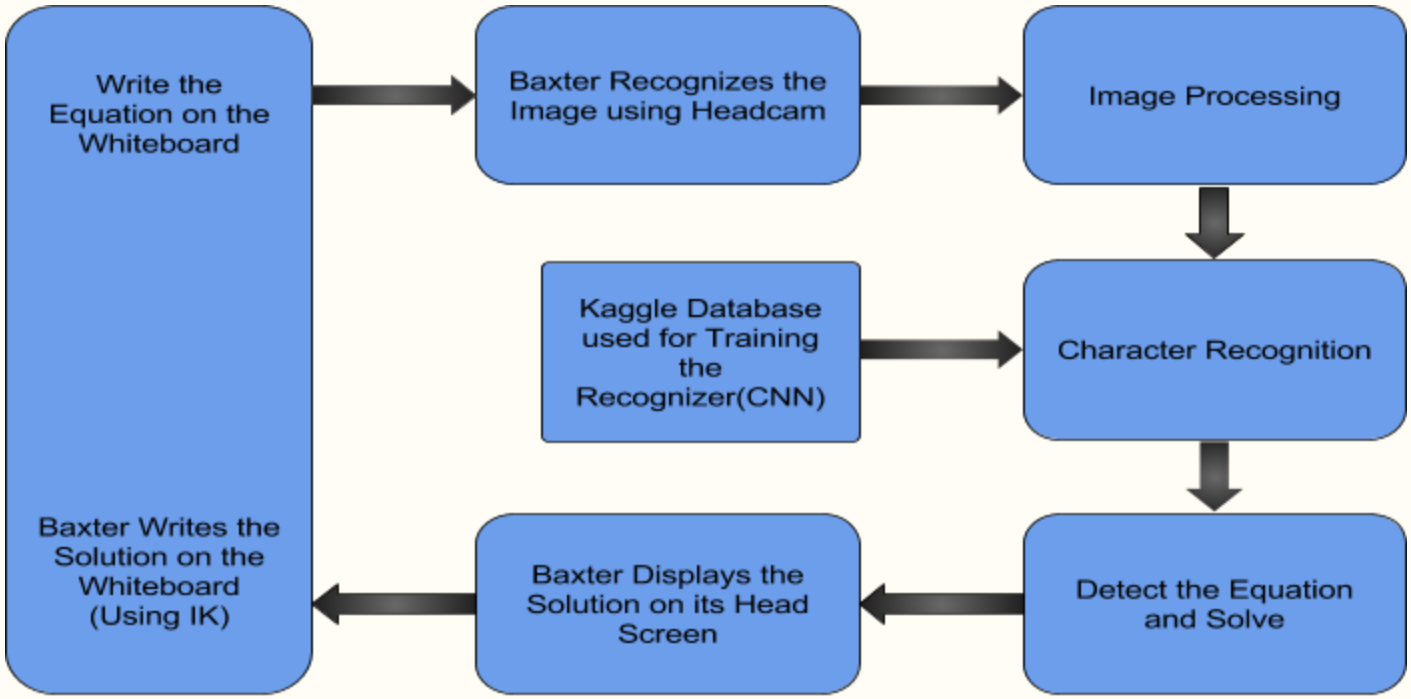
Total number of epochs - 30.
Batch size - 128.
Learning rate - 0.001

All the above hyper-parameters were chosen iteratively and the the parameters which provided maximum accuracy and a faster time was chosen for the network.

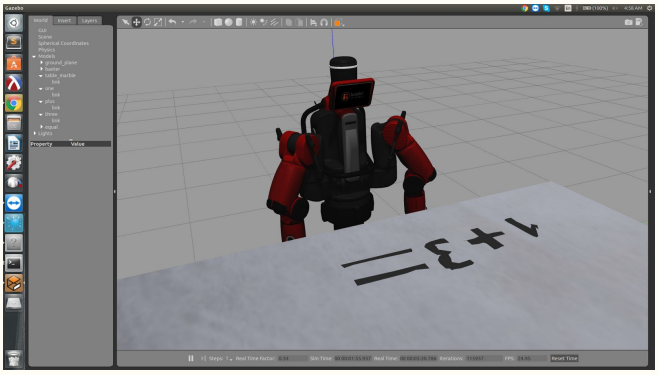
The answer is calculated on the basis of the results of image processing module. It is then displayed on baxter's screen. This process is done using a subscriber publisher model. Same as was used before when using the headcam.

Then desired joint angles are passed to baxter to get it in the correct position to start writing. This is done using ros forward kinematics tools. This places baxter's right arm in a hovering position over the table.

Then the end effector position and orientation is fetched from baxter interface. The path points are fetched for the answer returned from the image processing module and they are used to guide baxter's arm through all points to write the character onto the table. MoveIt's planning and execution abilities are leveraged for this purpose. A constraint is added on the orientation of the end effector to limit possible solutions and paths while also making sure the end effector faces down throughout. .



Data Flow and Architecture



Baxter Initialized and equations placed

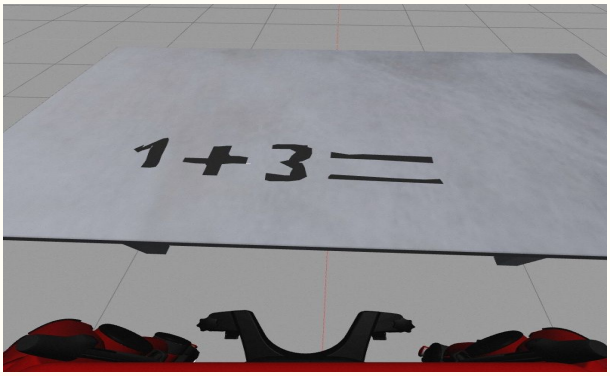
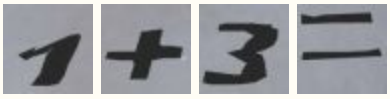
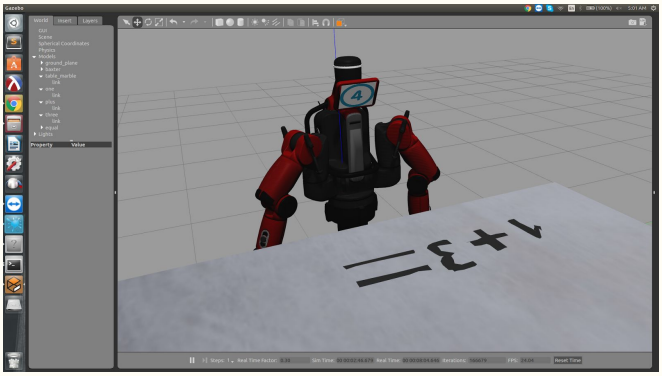


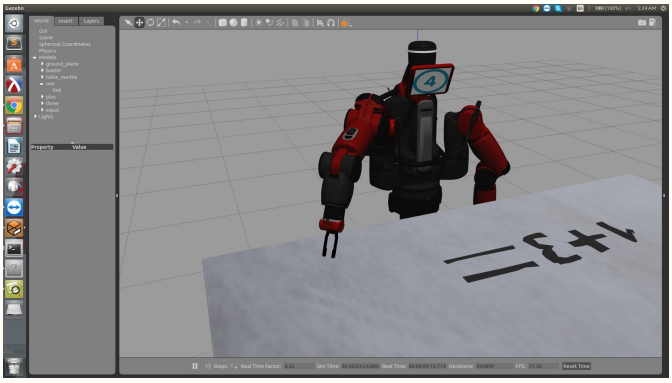
Photo taken by Baxter headcam



Images post segmentation



Baxter displaying solution on screen



Baxter writing solution on board

DISCUSSION

While doing this project, we faced an abundance of difficulties, far more than we expected. Many of which can be attributed to our unfamiliarity with ROS and its APIs. There were numerous technical problems during setup and implementation as detailed below:

Setup:

- We faced numerous issues while installing ROS on ubuntu because of the many dependencies and specific files that needed to be changed.
- Since we implemented this project using gazebo simulation, the hand writing part had to be simulated using gazebo models. We had to create our own models for the character due to our specific needs.

Implementation:

- The major issue during implementation was our limited knowledge of ROS and baxter. Due to this we faced many difficulties while using services related to Inverse Kinematics(IK), Head Cam, and Head Display.
- Since we used MoveIt for handling the IK requests during the writing of the solution, we had to go through all details of how to call functions of MoveIt module. Also, since Moveit is primarily written in C++, it was difficult for us to get proper tutorials of it.
- Since we simulated our project in gazebo, there were many limitations that we faced such as baxter writing not visible, improper placement of character models under baxter. These problems were accentuated due to visualization of 3D space in 2D.

RESULT

Baxter is able to write appropriate answers to basic arithmetic problems. The sensors work as expected and store and display images. The digit recognition works with a high level of accuracy. Baxter is able to move its end effector according to the given path points. Since the movements are done in a simulator without an actual pen. The answers are best seen when simulated in the real world. Till then a vivid imagination is a mandate.

SCOPE FOR IMPROVEMENT

- Character recognition is presently limited to addition and subtraction because simulating characters is a tedious task. The present implementation can easily be expanded to include any character.
- Presently the drawings made by Baxter are not perfect because of fewer path points for individual digits as well as the inaccuracies in Baxter's movements. Increasing the number of points will result in clearer "handwriting".

LEARNING

- Running Robot simulations in Gazebo.
- Understanding of ROS concepts and operations.
- Model creation for Gazebo.
- Image processing.
- Convolutional Neural Networks.
- Robot Control and Systems.

BIBLIOGRAPHY

- Sdk.rethinkrobotics.com. (2017). *Baxter Setup - sdk-wiki*. [online] Available at: http://sdk.rethinkrobotics.com/wiki/Baxter_Setup [Accessed 9 Nov. 2017].
- Wiki.ros.org. (2017). *kinetic/Installation - ROS Wiki*. [online] Available at: http://wiki.ros.org/kinetic/Installation [Accessed 14 Nov. 2017].
- Kaggle.com. (2017). *Handwritten math symbols dataset | Kaggle*. [online] Available at: https://www.kaggle.com/xainano/handwrittenmathsymbols/data [Accessed 23 Nov. 2017].
- Moveit.ros.org. (2017). *Plugins | MoveIt!*. [online] Available at: http://moveit.ros.org/documentation/plugins/#movegroupcapability [Accessed 2 Dec. 2017].