roBob the builder

EE106A Final Project Proposal

Fall 2020

1 Contact Information

Name	SID	Email (@berkeley.edu)
Aatif Jiwani	3032638026	aatifjiwani
Bhavna Sud	3033974090	bhavnasud
Brian Yu	3034022372	bri25yu
Nicholas Figueira	3033669292	nfigueira

2 Abstract

Our project aims to tackle the task of a robot using blocks in its environment to build a structure. Given a number of images depicting a structure made of cubic blocks, TIAGo will attempt to use its current environment and itself to move blocks in a manner that will try to reproduce this structure. This task heavily relies on planning and sensing over actuation as the meat of this project will come down to using cameras and lasers to model the current world and some planning algorithm to tell the robot how it should move around the world and orient its arm to pick and place blocks. The actuation aspect of this robot will come down to moving the agent to a desired location and moving its joints to a position such that the gripper can pick up and place a block.

The robot we plan to use is the TIAGo robot that has navigation capabilities and a singular arm.

3 Project Description

3.1 Project goals

1. Be able to reconstruct a model of a structure based on images of the structure from different angles
2. Have TIAGo be able to identify where the blocks are in its world and move around the world to pick and place blocks, while avoiding obstacles
3. Have TIAGo recreate the original structure by placing blocks layer by layer

3.2 Project design requirements

The project essentially has a TIAGo robot reconstruct a structure made of cubes, based on images of the structure. The design criteria the project must meet are:

1. Create a structure made of cubes in Gazebo simulation, and take images from different angles of the structure
2. Correctly identify the bottom left coordinates of each of the blocks in the structure
3. Create a world with TIAGo and cubes placed around the world

4. Correctly build a map of the world using SLAM, and identify the positions of all of the cubes lying around in this world
5. Have TIAGo be able to take the position of a cube and plan a path to that cube, pick it up, and plan a path back to the structure it is building
6. Orient TIAGo's arm correctly so it can place the cube, and place cube correctly on structure
7. End up with the same structure as was originally built in simulation
8. Perform this process in a reasonable amount of time

3.3 Why is your project interesting?

Our project is interesting because it involves the very challenging image processing component of determining the locations of each cube in a structure based on images of the structure. This is difficult because we must recreate a 3d model based on just images, and after recreating the 3d model we have to separate the model into cubes and determine the position of each cube. The planning of how to best pick up and place the cube is also challenging, as we don't want to drop the cube and we don't want to push other cubes in the structure when we place a cube. Also, since TIAGo doesn't originally know where the cubes are in its world, our project also involves exploring the world to find the cubes using SLAM.

3.4 Sensing, planning, and actuation complexity requirements

3.4.1 Sensing

1. Taking images/processing images to determine position of cubes in structure
2. Uses laser data to determine TIAGo's position in world and position of cubes lying around TIAGo's world
3. Uses laser/potentially camera data to avoid obstacles (blocks lying around) when moving around world

3.4.2 Planning

1. Planning for original loop around world to determine positions of cubes (where to explore next while avoiding obstacles)
2. Planning for how to move around world and go towards blocks/towards structure (feedback controller), while avoiding obstacles (using laser data)
3. Planning for how to pick up/place blocks using MoveIt Controller (how to orient arm to pick up and place a block, when to close/open gripper)

3.4.3 Actuation

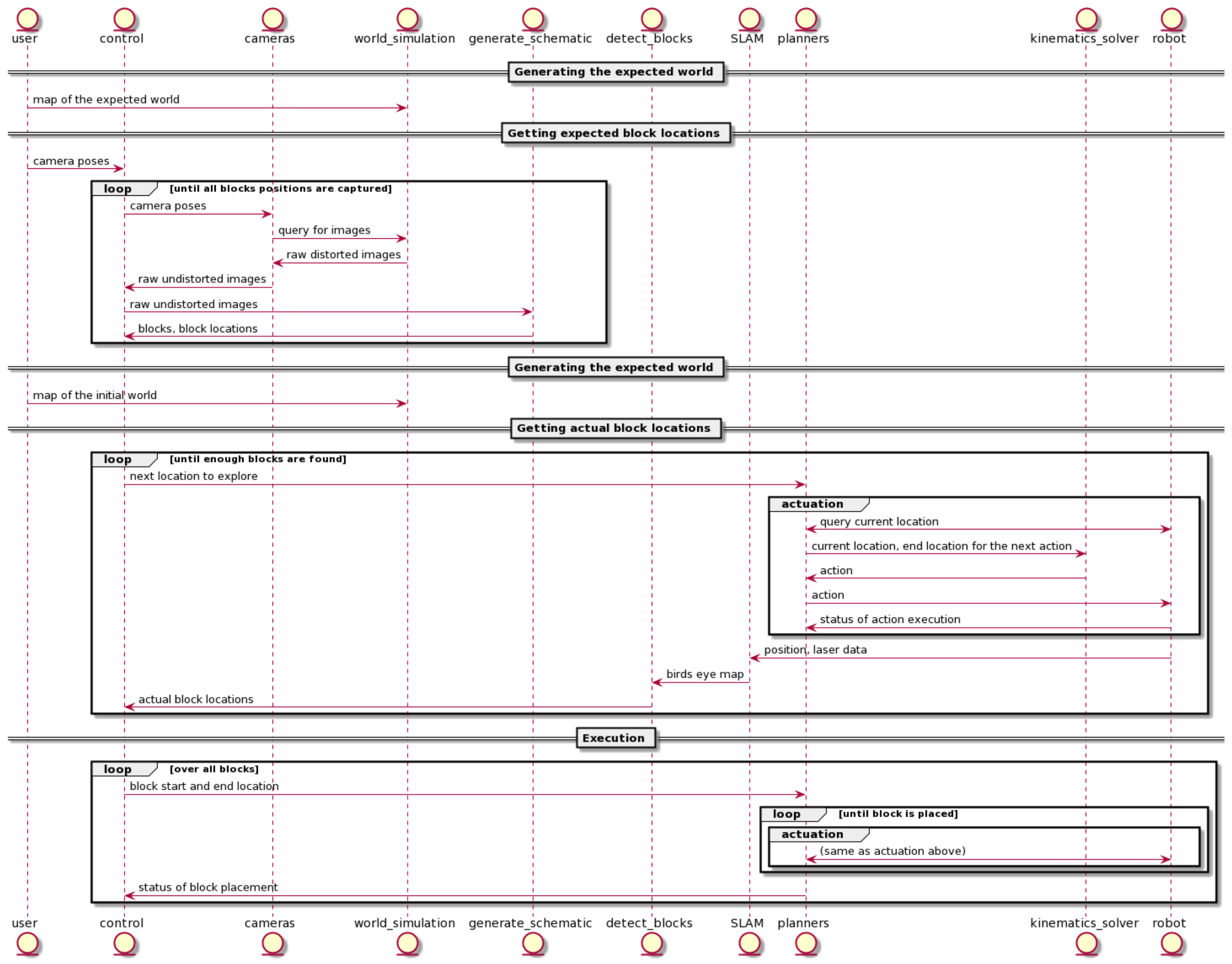
1. Move TIAGo to desired location by setting desired velocity every few timesteps
2. Move TIAGo's joints to a position such that it can pick up/place a block
3. Open/close gripper

3.5 What similar work have other groups done before? How is your work different?

Amay and Ron worked on a similar project when they took 106A. Our work is different because our project is completely in simulation, so we take our images in simulation and move TIAGo in simulation. This adds the extra task of identifying the positions of cubes in TIAGo's world. We also are using only cubes, while they used a block with a circle at the top. This will make the task of picking up the block slightly different. The main difference is that we are using a different robot, TIAGo.

4 Design

Sequence diagram



5 Tasks

For each of the following tasks, we make the following assumptions:

- Our robot doesn't care about box color, but the boxes are different colors in the image
- All boxes will be cubes of the same size, and we don't care if the cubes we use to build our structure are the same size as the cubes from the original structure
- Our world has no walls
- Blocks in structure are adjacent

1. Build simulation world in Gazebo
   (a) We will have 2 separate worlds: one for the ideal world which will contain the expected structure, and another for the TIAGo robot to interact with the environment to recreate the expected structure. (Lab 4)
   (b) World 1 (the *expected* world)
      i. Map with the ideal structure we want to recreate

   ii. Set up two gazebo depth cameras (See here)
   iii. Take two images of structure from different angles (angles and positions of cameras will be passed in)

(c) World 2 (the *actual* world)
   i. Map of cubes randomly placed in a new world
   ii. TIAGo robot with one of two grippers (a rectangular gripper similar to Baxter's and a hand)
      A. We will be toying around with the block size and grippers to find an optimal pair of configurations

2. Process images for expected box positions in world 1
   (a) Case 1: structure has no height (e.g. a pathway)
      i. Capture a head-on image of structure
      ii. Use Canny edge detection to identify edges of boxes in images (assuming all boxes are different colors) (Lab 6)
      iii. Return 3d coordinates of bottom left corner of all boxes (use x and y coordinates from image, and assign all blocks the same z coordinate)
      iv. This is a base goal
   (b) Case 2: structure has height (e.g. a house)
      i. Take 2 images from 2 different locations, perturbing in all 3 directions
      ii. We know camera positions and angles (these were inputted to the program), so we can compute essential matrix
      iii. Use Canny edge detection and Harris corner detection to identify edges/corners of boxes in images (assuming all boxes are different colors) (Lab 6), and find corresponding points in the other image (Lab 7). See here
      iv. For each edge and corner, identify its location in 3d space using triangulation technique from lab 7
      v. Can get camera projection matrix using camera API
      vi. Return 3D coordinates of bottom left corner of all boxes (with respect to first camera frame)
      vii. Note: we might miss some blocks if we can't see them from two images
      viii. This is a goal we hope to reach (not a stretch goal)
   (c) Case 3: structure is 3D, multiple images:
      i. With multiple images from a bunch of different angles, we won't miss any blocks, but we're not sure how to stitch multiple images together yet. This is a stretch goal.

3. Find actual block positions in World 2
   (a) Use SLAM to map out where the current blocks are in the world
      i. To map out the entire world, we need to explore to ensure we can find all blocks
      ii. Controller uses the current SLAM-generated map to find a new target position to go to and moves towards it
      iii. We can use a feedback controller to get to the position
      iv. Controller identifies blocks in its current view by using edge detection on SLAM map (SLAM returns matrix of 1s and 0s)
   (b) Stop once we've found enough blocks for our structure

4. Code execution of structure building in World 2 (repeat in loop)
   (a) Determine a box's location in the current world and where we want to put it based on World 1
      i. We can do some bookkeeping by keeping track of which blocks have been placed, and place new blocks in the remaining spots

(b) Use the SLAM generated map to plan and move to a box

    i. ROS has many supported 3rd-party libraries depending on the robot we use (see here)

    ii. If we do not use 3rd-party software, we can try using a controller similar to the feedback controller from Lab 4 section 7.2

        A. If we see that we are getting very close to an obstacle, then we need to move in the opposite direction with respect to the obstacle

(c) When the robot arrives at the desired location, pick up box

    i. Use Moveit/PathPlanner from lab to plan out path to pick up box

    ii. Possibly will need to use the hand gripper or restrict the size of the box to fit inside the rectangular gripper

(d) Determine the desired location of the box we picked up

    i. May be right near the robot or on the other side of the map

(e) If we the desired location is farther than the current reachable space, then plan path to that location using the same methods in step B or C

(f) Once we are at the destination position, place the box

    i. Use MoveIt to plan out the correct configuration of the robot's joints

    ii. For boxes next to each other, position end effector 1 block above where it needs to be so that it doesn't get in the way, then open gripper to release box

6 Milestones

Week of	Milestone
Oct 30	• Final project proposal finished
Nov 6	• Create world simulation class, build a few simple worlds for sanity testing • Test out integrating with SLAM, MoveIt, PathPlanner, TIAGo, etc
Nov 13	• Generating structure schematic on 2D inputs • Leveraging SLAM to find block positions
Nov 20	• Generating structure schematic on 3D inputs (initial) • Control planning (initial)
Nov 27	• Improve generating structure schematic on 3D inputs (finished) • Control planning (finished)
Dec 4	• Buffer
Dec 11	• Final presentation finished • Demo finished
Dec 18	• Final report finished • Video finished

7 Assessment

7.1 Goals

- We are able to create a structure in 3D space made of blocks
- Given a structure (specified through yaml file), we are able to rebuild a model through images of it
- Given a world with blocks in it (specified through a yaml file), the robot is able to map out its environment and find the location of blocks
- Given a map of its environment, the robot is able to pick up and move blocks to where it wants to build the structure
- Given a map of the environment and a model of the structure, the robot is able to build the structure
- The structure can be 3d, that is, it doesn't have to be just blocks in a line

7.2 Reach goals

- Have boxes in the structure and the environment be different sizes and shapes
- Have the robot build the structure matching the color of the structure
- Add unmovable obstacles to the environment and have the robot be able to work around those and still build the structure

8 Team Member Roles

Brian and Bhavna will be in charge of all of tasks 1 and 2. They will first work on task 1 (so tasks 3 and 4 can continue speedily), then will work on and improve task 2 (it will probably take a lot of iterations). We do not want to further split up the tasks within this group, since we'll be working together and pair programming a lot.

Aatif and Nicholas will take the lead on Task 3 and Task 4. They will first start with finding the packages necessary, and then use dummy data to stand in for task 2 to start with programming. We are both EECS, have backgrounds in software, and have dabbled in other CS domains.

Code: go brrrr

Aatif, Bhavna, Brian, and Nicholas: brrrr

9 Bill of Materials

Since we plan to execute this project virtually, we likely will not need physical materials.

10 Appendix

10.1 External packages

- Tiago simulation
- Tiago MoveIt!
- Tiago gmapping
- OpenCV
- rosnumpy

10.2 External links

- roBot the builder Github repo