Visual Motion Planning of 3-D Robots

Avikalp Kumar Gupta & Amitabha Mukerjee

November 13, 2015

Abstract

The aim of this project is to plan paths of multiple robots. This project is an extension of the work done by Debojyoti Dey in his M. Tech. Thesis [Dey (2015)] to 3 dimensions.
Debojyoti developed a roadmap composition technique in multi-robot environment. The algorithm is implemented on a visual configuration space. Key traits of the algorithm:

- *decoupled* motion planning (time efficiency)
- oblivious to configuration parameters (*generic* method)
- probabilistically resolution complete
- Space optimality: *neighbourhood product lattice* used so that Cartesian products of individual roadmaps are dynamically generated.

Debojyoti's algorithm was able to plan paths for 2-D robots. This project uses *v-rep* for simulation in 3 dimensions.
For a particular simulation environment with n cameras (vision sensors), each individual pose of the robot will be expressed as a set of n images, instead of just one. This project is hence a real-world implementation of the algorithm.
Since, for fixed cameras, occlusion can give rise to incompleteness, hence various ways of using body-fixed cameras were being thought about as the next step in the project.

1 Introduction

"Traditionally, the problem has been handled using a configuration space defined in terms of ***motion parameters*** *of the robots. An example of such motion parameters are joint angles of an articulated arm. In this work we propose a novel approach based on* ***visual input*** *alone. Such a model works on a random sample set of images of the robots without knowing the motion parameters concerned."*[Dey (2015)]
Even though the vision based model makes the algorithm more generic with respect to robots, it raises issues regarding the information captured. Debojyoti implemented the model for 2-D robots. In 2 dimensions, all the instantaneous information about all the objects in the environment can be captured by a single photograph. The same is not true in 3 dimensions.

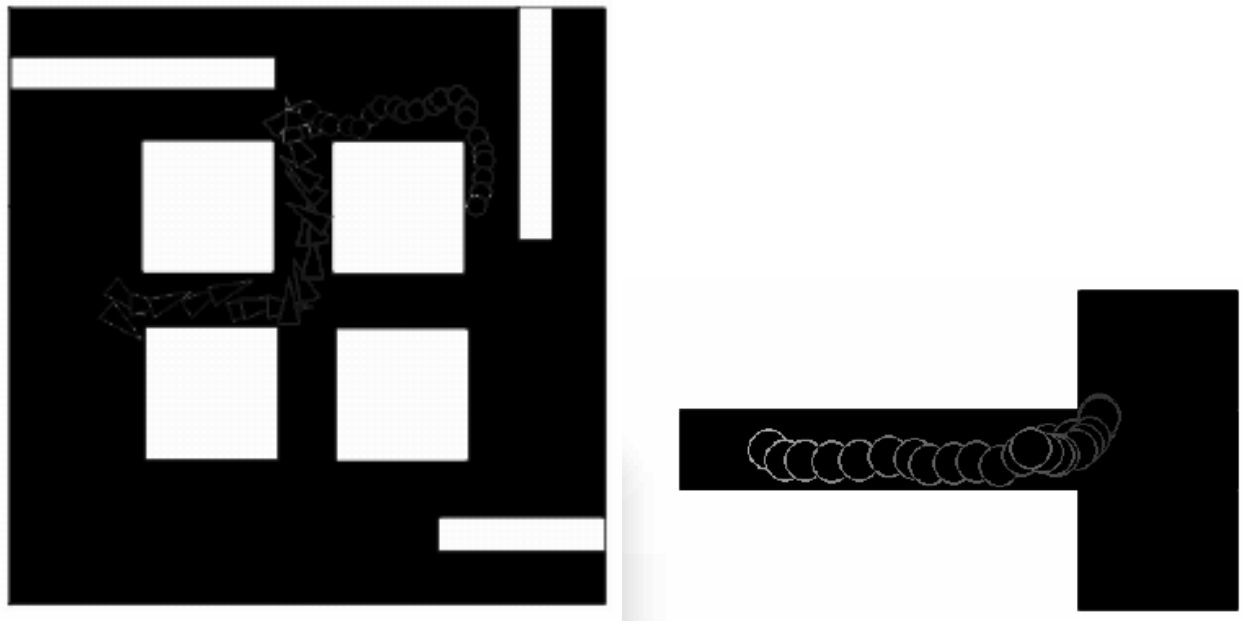

Figure 1: Examples of arenas for Debojyoti's code: Black implies free area and white means obstacle. *Image Source: Grayscaled (to highlight the arena) versions of images from [Dey (2015)]*

1.1 Problem: Data Generation

One of the problems, to begin with, is the data generation. In debojyoti's work, the images, which were used to generate the roadmaps for each robot, were generated in the following manner:

1. An image file, similar to the ones illustrated in figure 1, represented the arena.
2. A position on the arena is chosen at random (and an orientation is also chosen at random) to place the robot objects.
3. Images are validated on the basis of collision with obstacles. Hence, an image, in which any pixel of the robot coincides with any white pixel, will be rejected.
4. Each of the remaining image represents a valid pose of the robot in consideration.

1.2 Problem: Image Validation

We have to capture images of the environment using cameras and use the information from *all the images* to extract information. While generating the set of valid poses, for example, if we want to check collision, we have to go through all the images: the pose is valid if there is *atleast* one image in which the objects are not colliding.

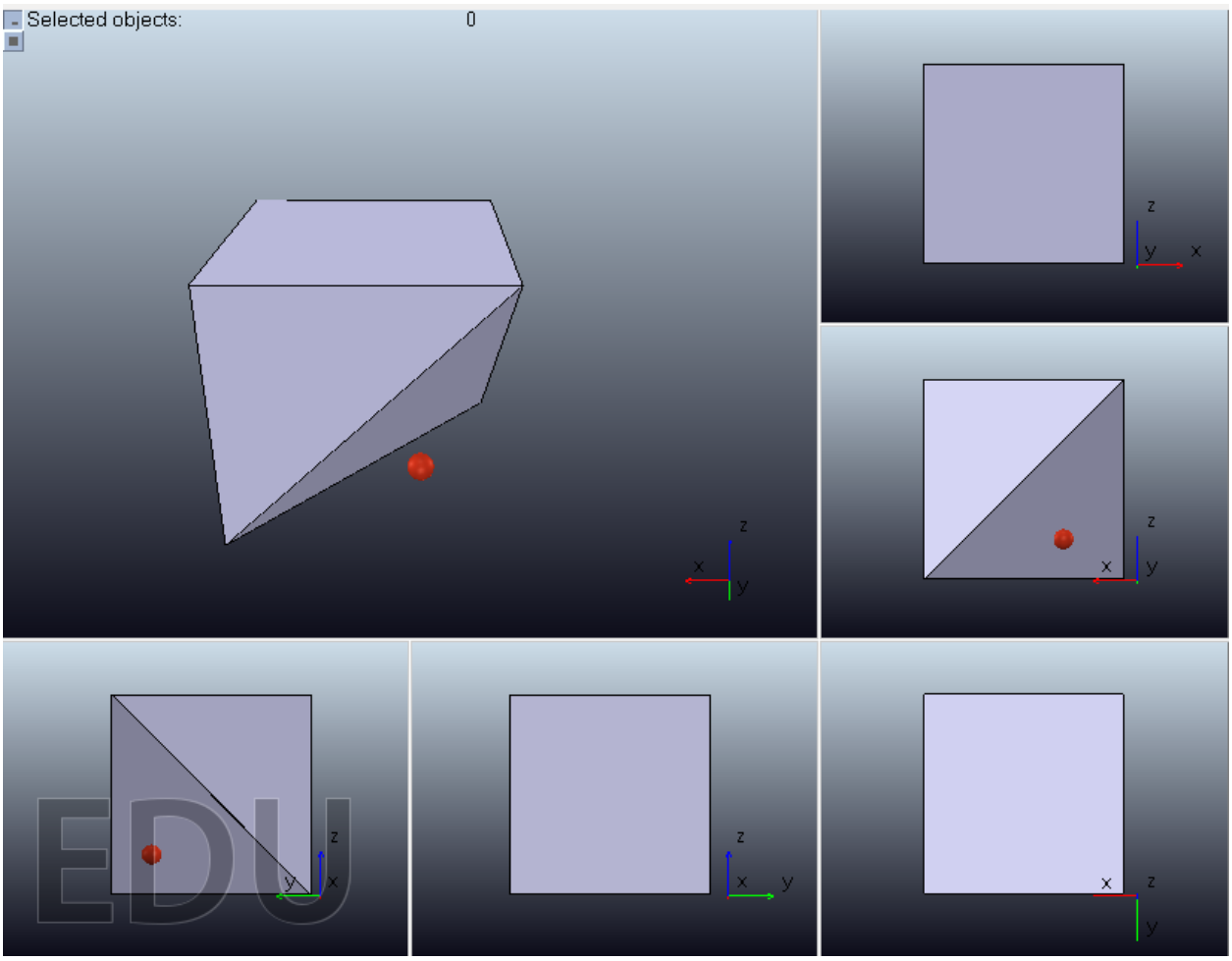

Figure 2: (left-top)A small sphere near a truncated cube; (others) all 5 cameras (placed at orthogonal positions) detect "false" collision *(image has been generated using v-rep software)*

But it turns out that we can never have enough (fixed) cameras. Consider the figure 2. The environment, in this case, consists of 5 cameras, strategically placed at orthogonal positions to capture maximum information about the scene. The environment does not have any collision, but still all the cameras together are unable to classify this pose as valid.

Apart from the genericness, the paper also describes the closeness of the vision based model to humans:
"Human beings do not appear to use global coordinates; instead, motion is encoded (both consciously and implicitly) in terms of relative poses. Mammals navigating in familiar environments acquire "place cells" which indicate their position in space, but these are organized in a system of neighbouring columns in the brain that encode nearby places. Thus, the representation appears to encode space in terms of nearby positions rather than a global coordinate."[Dey (2015)]

2 Idea/Approach

The approach is similar to Debojyoti's work, which includes the following steps:

1. Find a way of generating images for a particular pose of a robot from multiple cameras
2. Create a graph using these poses (described by the set of images corresponding to the same motion parameter values)
3. Ensure the validity of these poses, i.e. handle *"static collision"*
4. Assign weights to the edges of the graph using some metric, preferably distance between the poses
5. Construct a road-map from the initial to the final pose (using K-nearest neighbours)
6. Repeat the above steps for all the robots in the scene
7. Create a composite roadmap using the roadmaps of all the robots (this includes handling *"dynamic collisions"*)

3 Current Progress

From the above listed steps, currently we have implemented steps 1, 2 and 3:

1. : We have used *v-rep software and API* to generate images [figure 3.1].
2. : An external python script was written using v-rep remote API which would give random values to all the joint angles of any robot present in the scene and then capture the images from each vision sensor. These images can be used to generate the road-maps.
3. : Since v-rep only supports collision detection in its internal API, hence a work-around was figured out for this step. Motors in all joints in the environment were made immensely powerful, and instead of assigning random values to the angles, random values are assigned to the *target configuration* of the joints. And this value is changed while the simulation is running. Hence the robot never reaches any impossible configuration. Although, this has a drawback: the uniformity of the randomness gets reduced as the boundary configurations become more probable than the other configurations.

3.1 Choice of a 3-D modelling software

After going through some 3-D simulation softwares, we chose *v-rep* over the others due to the following:

1. Its free version (for educational purposes) supports most of the relevant features

2. It is a light software and supports 3-D rendering
3. 3-D modelling is supported inside the software, as well as 3ds-max/blender/CAD can be imported
4. Objects and simulations can be externally controlled using its remote API for both, C/C++ and Python. (remote APIs for other languages are also supported)
5. Proper documentation is available, and v-rep is also supported by an online forum

3.2 Image generation for a robot with 3 dofs (3 revolute joints)

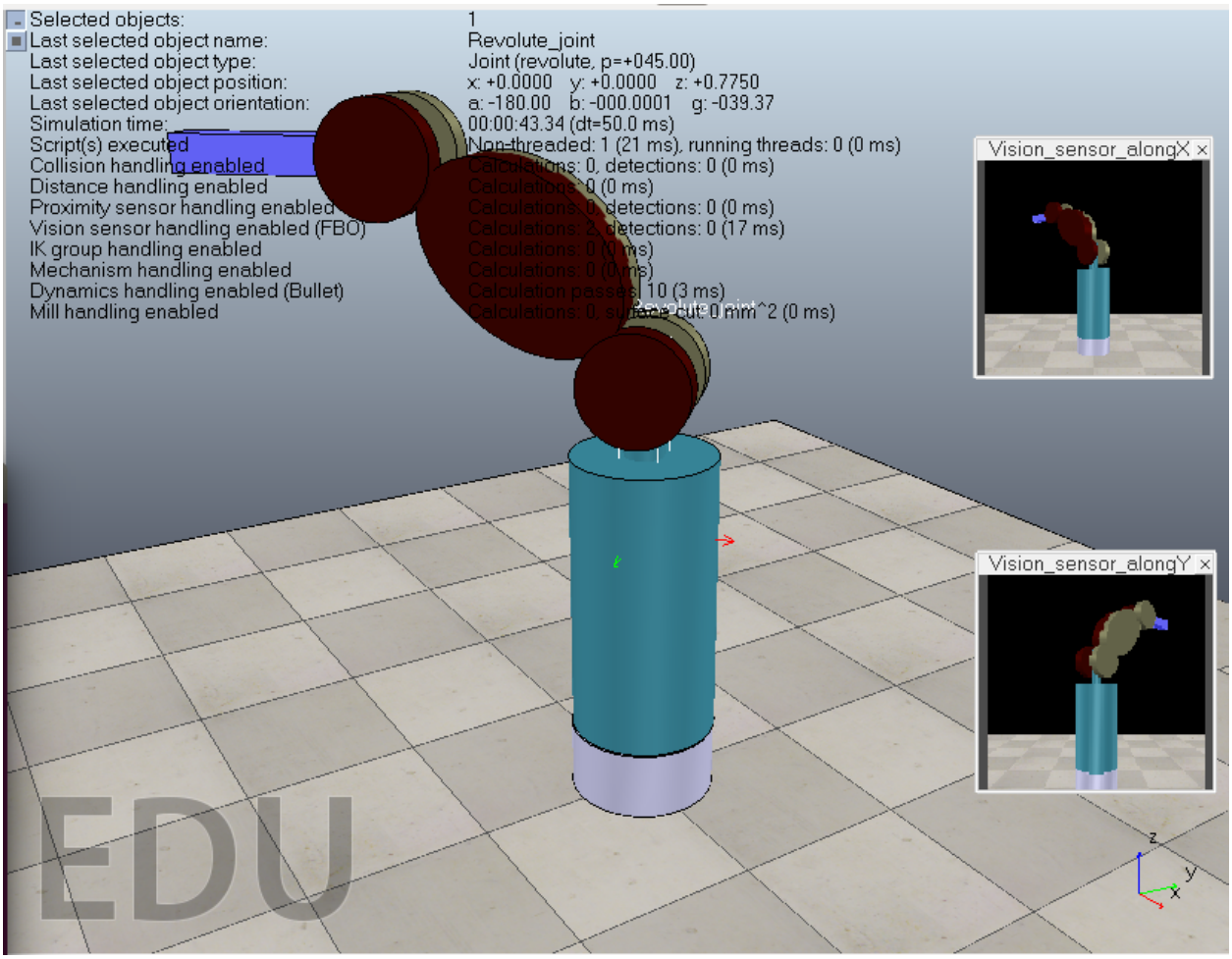

Figure 3: Robotic arm with 3 revolute joints - Created in v-rep. The 2 smaller windows show the image captured by the 2 vision sensors installed in the scene *(image has been generated using v-rep software)*

The robot in figure 3 was made using primitive shapes in v-rep. It has 3 revolute joints: One at the base, which makes whole system rotate about z-axis. And the other 2 have axes

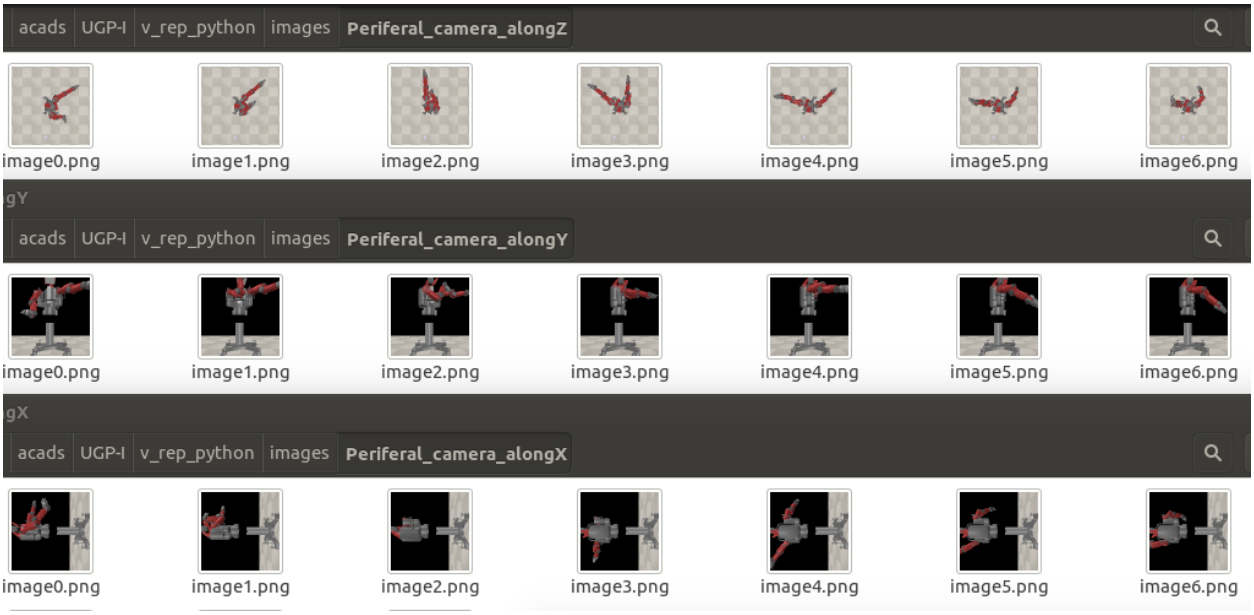

Figure 4: An external python script (which uses v-rep remote API) captures images from various peripheral cameras. Images with the same name belong to the same pose.

parallel to the $x-y$ plane. This was created because if you consider the system installed on top of the rotating base cylinder, it is the same 2R articulate arm explained in [Dey (2015)]. Hence we have sufficient knowledge and understanding of its behaviour.

3.3 Image generation for Baxter

Baxter robot is one of the in-built models in v-rep. It is a 2-armed robot with an animate face [Wikipedia (2015)]. It has 3 vision sensors installed on itself, one on the head and one on each arm. Apart from this, it also has many proximity sensors on itself, which are currently irrelevant to us. Both of baxter's arms have 7 degrees of freedom (and hence are redundant).

Apart from the robot, there are 3 other vision sensors (cameras) in the environment, one on each global axis. The images were generated for all the six cameras, but in the absence of an environment and obstacles, the images from the cameras installed on Baxter were mostly blank. The first few images from the other vision sensors are shown in figure 4. You can see that they belong to the same configuration of the robot.

4 Future Work

For the direct extension of Debojyoti's work, the first thing to do now is to complete the remaining steps.
But it is quite evident that, due to occlusion, the results produced by it will not have all the traits which the planar model had. Particularly, the algorithm will not be probabilistically

resolution-complete.
Hence, now we will proceed by understanding how humans and animals plan their motions. The implementation will be similar to paper by Amitabha Mukerjee and Divyanshu Bhartiya called "A Visual Sense of Space" [Bhartiya and Mukerjee (2015)] in which they have proposed a visual characterization for the complex motions of an unknown mobile system.

Another method could be using body-fixed cameras for capturing information about the environment (example: the case of Baxter) and making the robot learn to find the shortest path. Some study about the various neural network learning algorithms (deep learning and RNNs (including LSTMs)) was also done during the course of this project to proceed in this direction.

References

Bhartiya, D. and Mukerjee, A. (2015). A visual sense of space. *Procedia*.

Choset, H. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementation*. A Bradford book. Prentice Hall of India.

Dey, D. (2015). Visual Motion Planning of Multiple Robots by Composing Roadmaps. Master's thesis, IIT Kanpur, India.

Wikipedia (2015). Baxter (robot) — wikipedia, the free encyclopedia. [Online; accessed 8-November-2015].

5 Acknowledgement

I would like to specially thank Mamidela Seetha Ramaiah, a.k.a. MS Ram who had shared with me some very useful libraries he had developed for processing images and generating graphs over them. He also helped me in understanding concepts related to robotics and manifolds (using Choset's book Choset (2005)).

I would also like to thank Debojyoti Dey for spending some very precious time in explaining his implementation.

Lastly, I would like to thank Prof. Amitabha Mukerjee, who is the superviser of the project, for his guidance and expert advice through the project.