Exploring the Motion Manifold for an Articulated Arm

Avikalp Kumar Gupta & Amitabha Mukerjee

November 18, 2015

Abstract

This project is a step towards the extension of visual motion planning algorithms (specifically Debojyoti Dey's M.Tech. thesis work [Dey (2015)]) to 3 dimensions. In this project, we try to generate the *motion manifold* of an articulated arm using only *visual input*. The articulated arm considered in this project has 3 revolute joints, and hence has 3 degrees of freedom.

We use a simulation environment with 3 cameras (vision sensors). Hence, each individual configuration of the robot is expressed as a set of 3 images. We call this set of images, or the configuration it represents, as a *pose* of the robot. A graph is generated over these poses by connecting *k-nearest neighbours* for each pose. The distance between two poses is measured as the Euclidean distance between the pixel values. This graph is supposed to represent the motion manifold of the robot.

When the graph is constructed over the joint angles, it results in a motion manifold (a thick hollow cylinder, $\mathbb{R}^2 \times S^1$) as expected. The visual manifold is corrupted owing to similarity between distant poses, and results in *short-circuits*. The phenomena are discussed and possible approaches to overcoming it are proposed.

1 Introduction

Our ultimate aim is to plan paths of multiple robots. Debojyoti had developed a roadmap composition technique in multi-robot environment. Key traits of the algorithm:

- *decoupled* motion planning (much more efficient than *centralized* motion planning in terms of time)
- oblivious to configuration parameters (The algorithm is implemented on a visual configuration space, and hence is a *generic* method)
- probabilistically resolution *complete*
- Space optimality: *neighbourhood product lattice* used so that Cartesian products of individual roadmaps are dynamically generated.

This algorithm was able to plan paths for 2-D robots.

"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. This includes the complete configuration of the robot. The same is not true in 3 dimensions.

The need of images of the environment from multiple views is obvious. But the number of cameras, that will be sufficient to capture the *complete* information about the environment, depends on the environment itself because *occlusion*, which depends on the geometries of the robots and the obstacles, can give rise to incompleteness. Consider the figure 1. 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. Algorithms, which can generate the a 3D image using a scan through the environment Bhartiya and Mukerjee (2015), can be used to solve this problem in the future.

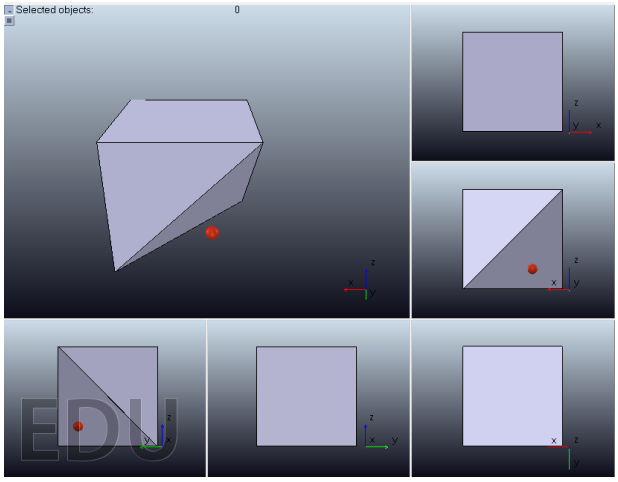


Figure 1: (left-top)A small sphere near a truncated cube; (others) all 5 cameras (placed at orthogonal positions) detect "false" collision *(image has been generated using v-rep software)*

1.1 3-D Articulated Arm

Since we wanted our method to operate on only visual input, hence we moved forward to work on an algorithm, which might not always be able to find a path, if there exists one, but will only output *correct and collision-free* paths. We started with a 3-D articulated arm with 3 revolute joints (3R) as shown in the figure 2. We installed 3 cameras at orthogonal positions in the environment, which will capture the visual input for our algorithm.

This project uses *v-rep* for object modelling and simulation in 3 dimensions. The arm has been created in the *v-rep* environment,

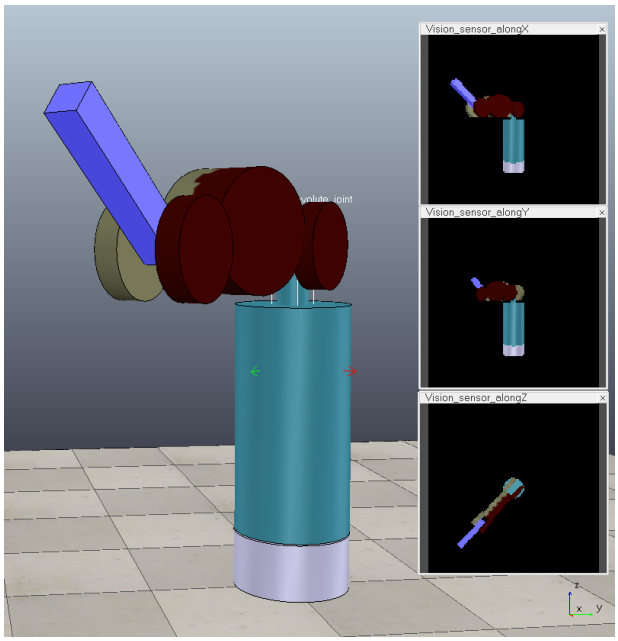


Figure 2: Robotic arm with 3 revolute joints - Created in v-rep. The 3 smaller windows show the image captured by the vision sensors installed in the scene *(image has been generated using v-rep software)*

and all data, including joint angles' information used for validation, is extracted using the *remote API* functions provided by v-rep. The code for this project is written in *Python* using multiple libraries, including some created by M.S.Ramiah, IIT Kanpur. Stanford's MATLAB code for ISOMAP was used for visualizing the lower dimensional manifold of the images.

1.2 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

2 Graph Generation

For the extension of Debojyoti's algorithm, the following steps will have to be followed prior to the creation of the roadmaps. The requirements of these steps will be the basis of all the work done in this paper.

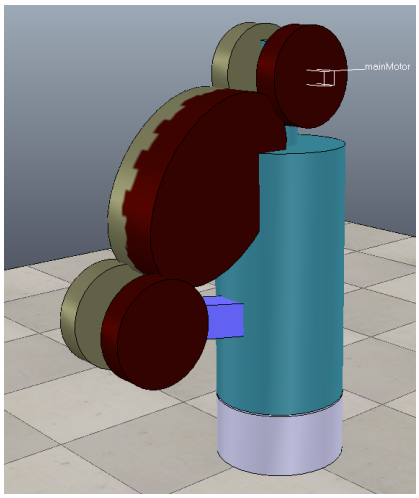


Figure 3: An impossible pose generated by assigning random angles to all the joints

1. A single set of images for the static obstacles will be captured.
2. Each robot will be simulated in an independent environment. Random *target angles* for all joints will be given during the simulation, and images will be captured when the robot becomes stable. Simulation will be done to avoid *impossible poses* (eg. figure 3).
   Drawback: Reduction in the uniformity of the randomness as the boundary configurations become more probable than other configurations (figure 4).
3. Poses, in which the robot "seems to be colliding" with some obstacle in **all** the views, will be rejected. A robot "seems to be colliding" with some obstacle in a view if for some pixel, the robot's image and the obstacles' image, both have non-zero values in that view.

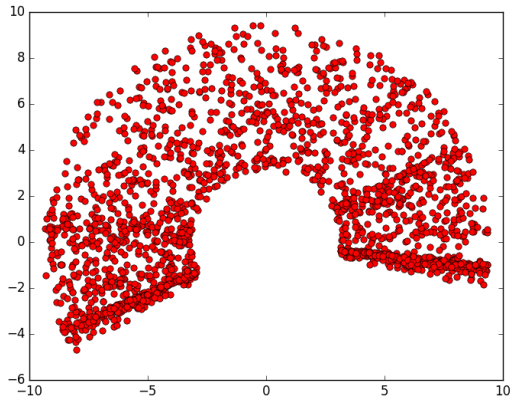


Figure 4: Plot showing accumulation of data points on the boundaries (*Details: Radial distance* $= 2\pi + \theta_2$*, and angle* $= \theta_0$)

4. A *completely connected* graph over all these remaining poses will be created. Each edge will have a weight, based on some metric (eg. distance).

5. Invalid edges (which lead to collision with obstacles during transition) will be dropped using *local planner*.

6. K-nearest neighbours for each vertex (pose) will be retained, and rest of the edges will be dropped.

This will generate the graph which represents the manifold over which the robot can move.

3 Current Progress

We have used *v-rep software and API* to generate images [see 1.2]. An external python script was written using v-rep remote API which would:

- Start the simulation of the currently active scene on the v-rep window
- Give random values to the target positions of all the joints
- Wait for the robot to reach a stable configuration
- Capture the images from each vision sensor, and store the motion parameters for that pose

Now this script could be used to generate the raw data for any robot [see section 3.1 and 3.2].

3.1 Image generation for the Articulated Arm

The robot in figure 2 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 (by angle θ_2), one shoulder joint (θ_1) and one elbow joint (θ_0), both having axes 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 R1-R1 articulate arm explained in [Dey (2015)]. Hence we have sufficient knowledge and understanding of its behaviour.

3.1.1 Manifold

As we know, the C-space of the system on top of the base cylinder would be a toroid (T^2), if there were no restrictions on the angles which were achievable by the joints. But since both of the top 2 revolute joints cannot rotate 360°, hence the configuration space is homeomorphic to $\mathbb{R}^2$ space, i.e. a rectangle.

The base revolute joint can rotate 360°, and hence has an S^1 configuration space. Hence we expect the C-space of the complete robot to be a ring-like structure (see figure 5, 6).

Unfortunately, this is not what we get in our first attempt. Figure 7 illustrates the

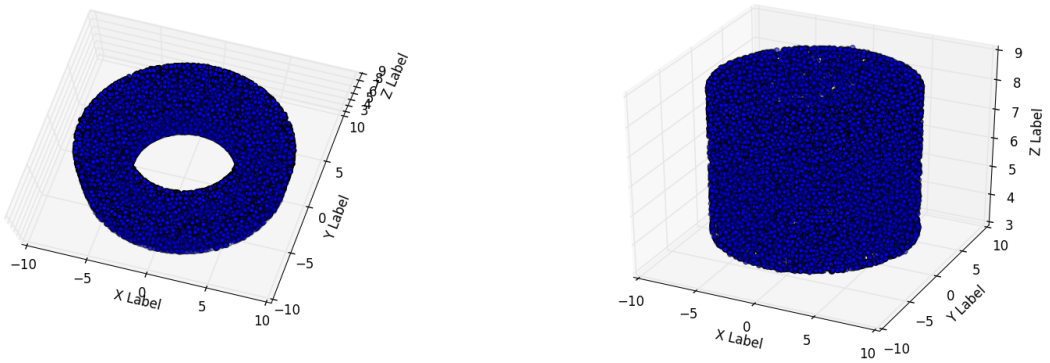
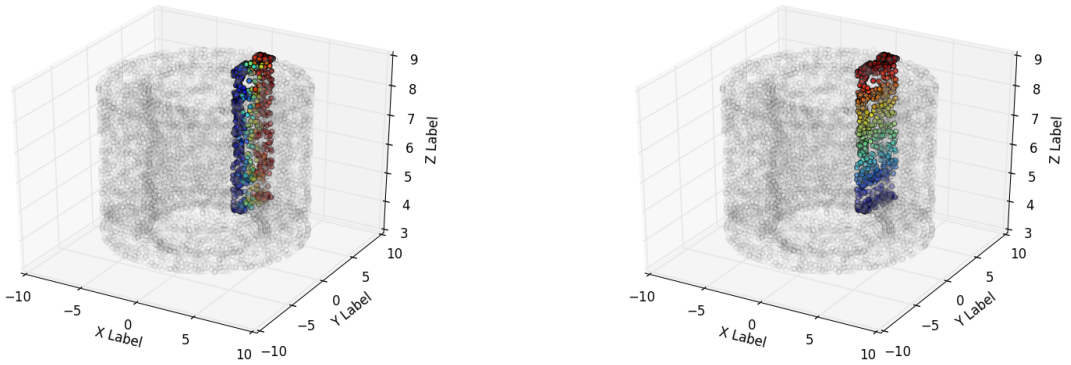


Figure 5: 3-D plot of θ_0, θ_1 and θ_2 is a hollow cylinder (*Details:* $2\pi + \theta_0$ = *distance from z-axis,* $2\pi + \theta_1$ = *distance from x-y plane &* θ_2 = *angular displacement along z-axis*)



(a) color changes as θ_0 increases

(b) color changes as θ_1 increases

Figure 6: 3-D plots explaining the structure of the hollow cylinder in figure 5 (*Details: The colored points only correspond to configurations which have* $0 < \theta_2 < 0.1$ *rad)*

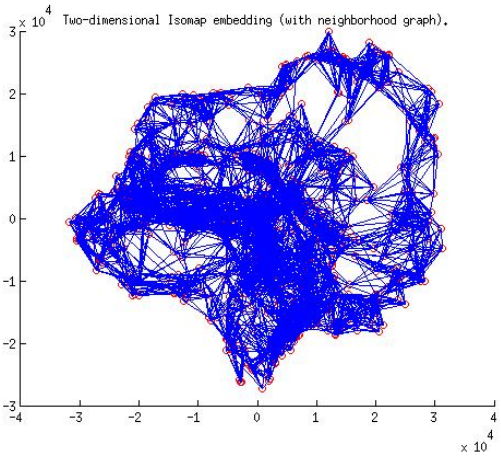


Figure 7: 2-D projection of the configuration-space of the robot, as deduced by dimensionality reduction using ISOMAP

motion manifold obtained on the graph generated on the images. In the 3D projection of the same, you can observe the presence of a cylinderical structure. In this graph as well, the expected graph with short-circuits can be figured out.

3.1.2 Difficulties

To understand these short-circuits, we generated the neighbours of a random set of poses in the k-nearest neighbours' graph. Two of the most significant such poses with their neighbours are shown in figure 8 and 9. These figures contain images from all three vision sensors, describing a particular pose, in a the same vertical. From these images, we figured out that the following were the major reasons of these short-circuits:

- a jump when 2nd link and the base cylinder are in the same spatial region (Figure 8).
- the opposite faces of the middle link get inter-changed (θ_1 change by 180^o, and θ_2 and θ_3 change signs) (9).

3.1.3 Proposed solutions

The solution of the first problem is to distinguish the different ends of the 3^{rd} link using different colors (figure 10).
The 2nd problem was a fore-seen problem, and it was expected that if we use different colors for the 2 different sides of the middle link, we will be able increase the distance between them. But it turns out that this distance did not increase by much. Hence we plan on *changing the distance metric* for the images to handle this. We will use track-points (called Ideal Tracked Points [ITP] in Ramaiah et al. (2015)) on the robot and measure the distance between two poses as the sum of the displacements of all the track-points. In this, our real challenge will be to recognize the track-points in each pose.

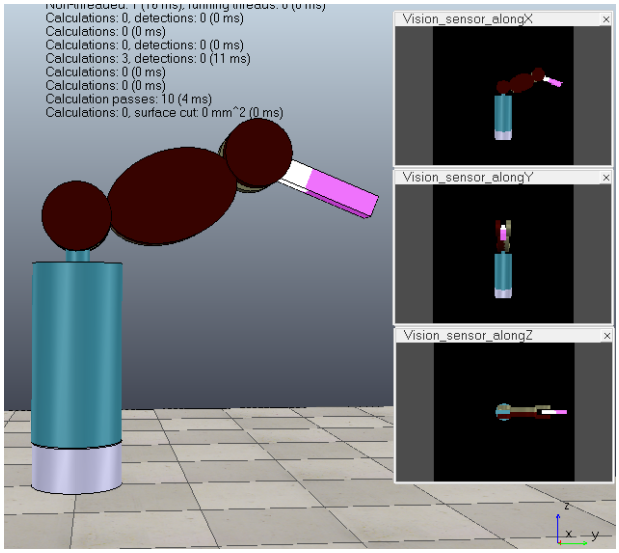
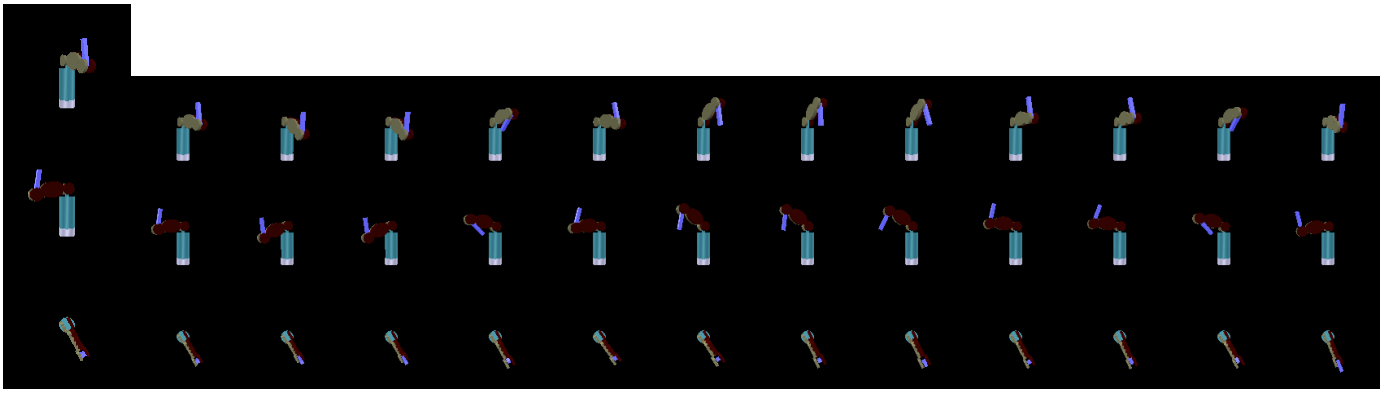


Figure 10: A proposed solution for dealing with short-circuits similar to fig. 8

3.2 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



(a) Pose (b) its neighbors

Figure 8: *Illustration of a short-circuit*: Poses in which the 3rd link is in the same spatial region, but the angles are far off, also become neighbours [$\Delta\theta_2 \sim 0$; $\Delta\theta_1 \sim 60^\circ - 100^\circ$; $\Delta\theta_0 \sim 180^\circ$]

(a) Pose (b) its neighbors

Figure 9: *Illustration of a short-circuit*: Poses which occupy approximately the same space, but are far off in the actual manifold become neighbours [$\Delta\theta_2 = 180^\circ$, $\Delta\theta_1 = -2\theta_1$ & $\Delta\theta_0 = -2\theta_0$]

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 11. You can see that they belong to the same configuration of the robot.

But due to the huge degrees of freedom, such an analysis on Baxter will require a huge set of images. To get an idea of the scale, the approximate number of poses that were required for the analysis of :

- planar articulated arm with 2-dofs: $\sim$ 5000 poses
- 3D articulated arm considered above (3-dofs): $\sim$ 22000 poses

And baxter has 7-dofs for each arm, and more joints in its torso as well.

4 Future Work

The first step in the future is the implementation of the proposals of this paper. Different data might be needed for getting the positions of the track-points.

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.

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.

This project would not have been possible without their help.

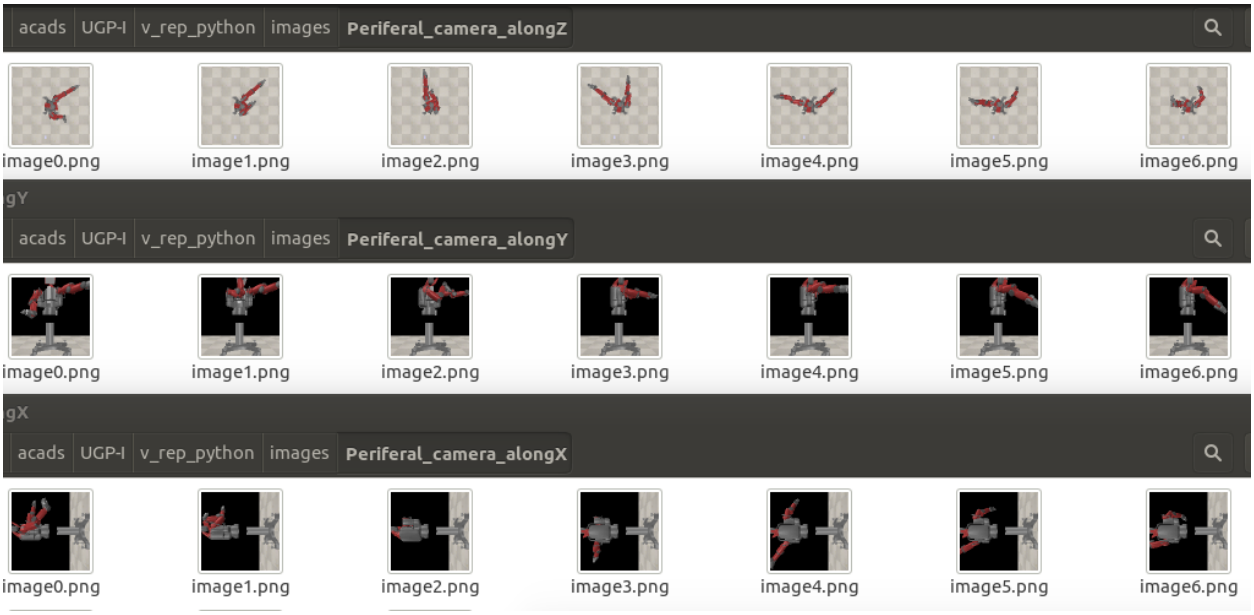


Figure 11: An external python script (which uses v-rep remote API) captures images from various peripheral cameras. Images with the same name belong to the same pose.

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.

Ramaiah, M. S., Mukerjee, A., Chakraborty, A., and Sharma, S. (2015). Visual generalized coordinates. *CoRR*, abs/1509.05636.

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