16-662 Robot Autonomy
Lecture 11
Jacobian Control and Probabilistic Roadmaps
Lecture by: Siddhartha Srinivasa

Alex Bunch
abunch@andrew.cmu.edu

Adwait Gandhe
agandhe@andrew.cmu.edu

Feb 25 2013

Abstract

In the previous lecture, we studied the theory about forward and inverse kinematics. We looked at some search algorithms in continuous space and discussed how collision checking is implemented. We also studied the matrix theory behind Singular Value Decomposition (SVD). In this class, we first discuss the theory of SVD in the context of the Jacobian Control. We present a simple example to understand how the three matrices obtained by implementing SVD on the Jacobian can be estimated by carefully analyzing the configuration of the body. We then discuss how Roadmap Methods are used to convert a problem of continuous search to discrete search. This is followed by a discussion on the Vertical Cell Decomposition technique and Visibility Graphs and we conclude this lecture by a comparison between these two techniques.

Contents

1 Jacobian Control

1.1 Overview of the Jacobian

The Jacobian matrix is the matrix of all first-order partial derivatives of a vector or scalar-valued function with respect to another vector. In the context of manipulators, the Jacobian is defined as the change in linear velocity of the end effector when angular velocity of one joint angle is one radian per second. The Jacobian matrix acts as a transform from the configura-

$$J = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}.$$

tion space to the task space. $\frac{dx}{dt}$ = J*$\frac{dq}{dt}$ An introduction to the Jacobian is provided in the previous lecture [2]

1.2 Overview of SVD of the Jacobian

An introduction to Singular Value Decomposition is also provided in the appendix of the previous lecture [2]. To briefly summarize, a Singular Value Decomposition decomposes a matrix into three simple transformation matrices; a rotation matrix V*, a scaling transform $\sum$ along the rotated axis and then another rotation transform U. The Jacobian can now be decomposed using SVD to yield the three matrices.

J = U $\sum$ V^T

We can identify several properties of the system by examining the SVD of the Jacobian. Firstly, the system has a Null Space if the $\sum$ is partly filled. Recall that the $\sum$ is a diagonal matrix of the singular values. The $\sum$ is partly filled implies that the singular values
$\sigma_1 \geq \sigma_2 \geq \sigma_3$ ... σ_r, $\sigma_r + 1 = 0$, ... σ_m =0

Then, we have $v_i = \sigma_i \ u_i$

Clearly, for i = r+1, $u_r + 1 = 0$

This implies that there is a configuration of joint motions that does not produce any effective motion in the task space. Hence, such a motion is a null space motion. Figure 1 illustrates this concept of a null space further. The matrix U_1 is called as the range. We see that the N-R rows of V_2 pro-

duce no net motion in the task space. Hence these rows correspond to the null space.

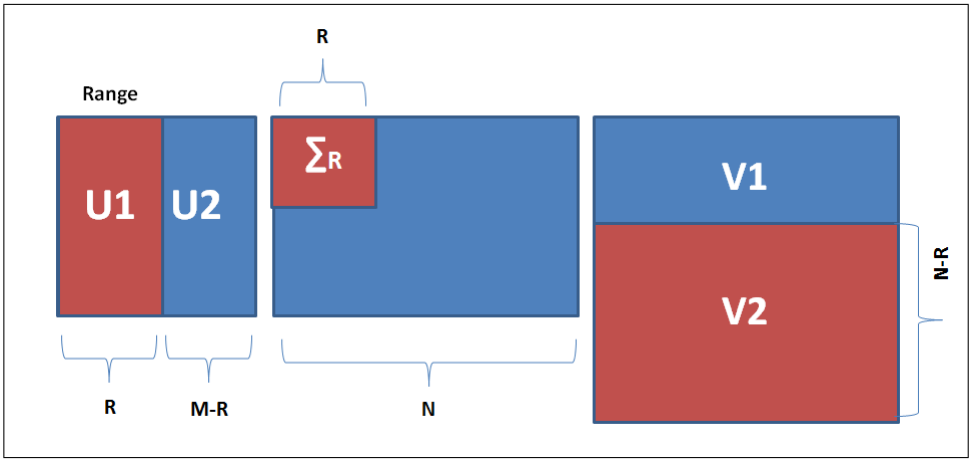

Figure 1: Singular Value Decomposition of the Jacobian

Figure 2 shows a configuration where the motion of the joints does not cause any motion in the task space. Now before we move to our example

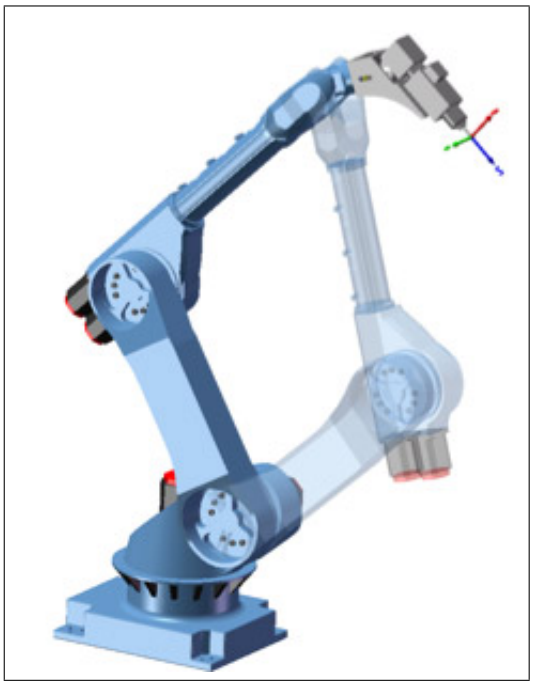

Figure 2: Example of a Null Space Motion

of estimating the three decomposed matrices of a Jacobian at a particular configuration, let us look at what is happening when the robot cannot move in a particular direction at a particular configuration. Figure 3 shows the span of the robot in the task space. We have seen previously that the optimal solution is the point $q*$ which is nearest to the origin. The robot cannot move to any point that is not in the span. Hence, all the motions on the span map to the origin of the task space. This is now clear from the previous discussion on Jacobians, SVD of the Jacobian and the null space. We have

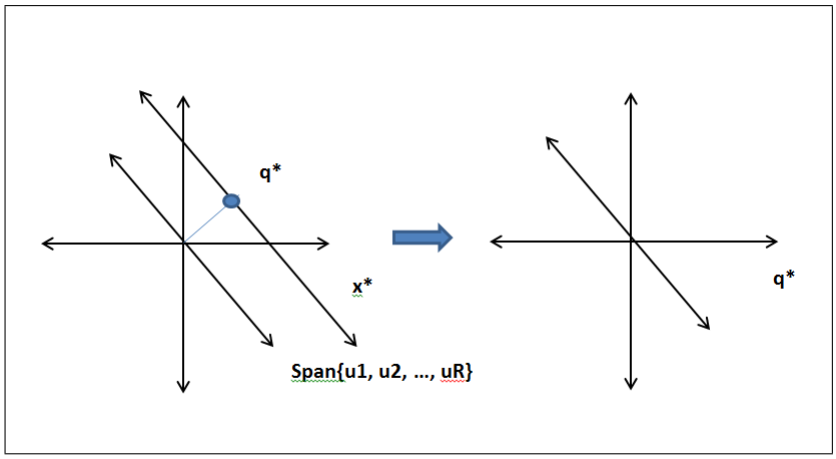

Figure 3: Span of the task space

seen previously that

$\frac{dx}{dt} = \mathrm{J}*\frac{dq}{dt}$

We can write this as

$\frac{dx}{dt} = \mathrm{U} \sum V^T * \frac{dq}{dt}$

Hence, we have

$\frac{dq}{dt} = \mathrm{V} \sum^{-} 1 \, U^T * \frac{dx}{dt}$

The Singular Value Decomposition hence gives us the closest solution. Note here that the inverse of the diagonal matrix will be the diagonal matrix with the values $\sigma_1^-1 ... \sigma_r^-1$

1.3 Example of estimating the Jacobian from a given configuration

Now that we have revised the SVD of the Jacobian and understood some key observations, we can take an example of a particular configuration of the manipulator and estimate the decomposed matrices of the Jacobian. Figure 4 shows the configuration of a 2 DOF arm. We will now determine the matrices U $\sum$ and V^T by observing this configuration and using the theory that we studied in the last sections. By definition, the Jacobian is defined as the change in linear velocity of the end effector when angular velocity of one joint angle is one radian per second. Let us use this to construct the

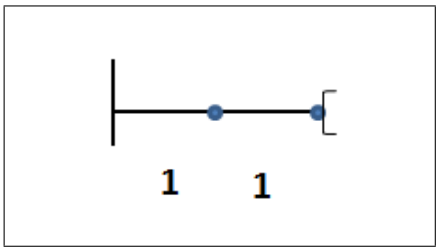

Figure 4: 2 DOF Robot Arm Configuration

Jacobian for this configuration. Assume that the joint near the manipulator is stationary and the other joint is rotating at an angular velocity of 1rad/s. We know that:

$\vec{v} = \vec{\omega}$ x $\vec{r}$

Hence, the end effector will move at 2 m/s in the positive y direction and at 0 m/s in the positive x direction. Similarly, when the first joint is held stationary and the second joint is rotated at 1 rad/s, the end effector will move at 1 m/s in the positive y direction and at 0 m/s in the positive x direction. Hence, the Jacobian of this configuration will be:

$$J = \begin{bmatrix} 0 & 0 \\ 2 & 1 \end{bmatrix}$$

Consider first the range U1 of the U matrix. The range determines the configuration in the task space that the robot can actually navigate to. In our example, it is clear that the robot can only navigate along the y-axis instantaneously. Hence, the range will be the multiple of the U1 matrix:

$$U1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Now can we think of a configuration in the null space ? Recall that the null space configuration will not cause any net movement in the task space.

Imagine that the first of the joints is rotating with an angular velocity of -1 rad / s and the second joint is rotating at 2 rad / s. The net velocity at the manipulator will then 2 + (-2) = 0 m/s. Hence, this configuration will impart no effective motion to the end effector. Hence, this configuration belongs to the null space. Hence we have

$$V2 = \begin{bmatrix} -1 & 2 \end{bmatrix}$$

So far, we have estimated the following elements of the three matrices:

$$J = \begin{bmatrix} 0 & - \\ 1 & - \end{bmatrix} \begin{bmatrix} - & - \\ - & - \end{bmatrix} \begin{bmatrix} - & - \\ -1 & 2 \end{bmatrix}$$

Now, the number of non zero elements in the $\sum$ matrix tells us about the number of degrees of freedom of the system. We also know that the $\sum$ matrix is a diagonal matrix. The matrices U and V are rotation matrices. Hence, these matrices are orthonormal. The singular values in the matrix decomposition are the multipliers or the scaling parameters. Hence after taking these points into consideration, we have:

$$J = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{5} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix} \frac{1}{\sqrt{5}}$$

Hence we have estimated the decomposed matrices and the Jacobian by observing the configuration

1.4 Key points

In the previous section, we saw how we can estimate the Jacobian and the decomposed matrices from the configuration. In this example, we were assuming that the exact configuration is achievable. Let us now consider what would happen if the configuration is really close to the assumed configuration, but is not that exact configuration. If we consider the same case as our previous example, a 0 in the transform implies that the circle in the q-space is a line in the x-space. However, if the configuration is not exact, this line will be a very thin ellipse. If we move away from the configuration under consideration, then the ellipsoid will get fatter. Hence, the skinnier the ellipsoid, the more demand on the angular velocity. The practical implication of this is that when the configuration is not exact, then there may be a sudden large demand for an angular velocity to reach that configuration. Hence it is important to implement checks to ensure that such sudden movement is not executed or attempted by the robot.

2 Roadmap Methods

2.1 Overview

When we have a path planning problem that is in a continuous space it is impossible to enumerate every single path to attempt to get from an initial state to a goal state.
In order to avoid this problem and reduce it into a simpler one that is easier to solve we use 'Roadmaps' to convert continuous problems to discrete. The main idea of 'Roadmaps' is that we break our problems into three parts:

- Get to the highway
- Traverse highway
- Get off highway and go to destination

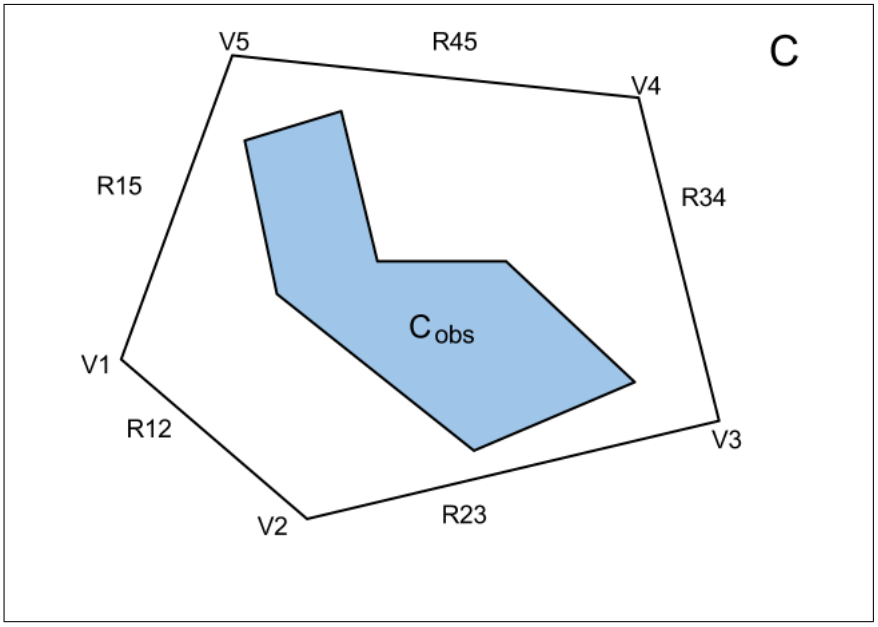

Figure 5: Example Highway

2.2 Definition

We define our problem thusly:

$$Vi \in C_{free}$$

$$GraphG(V, E)$$

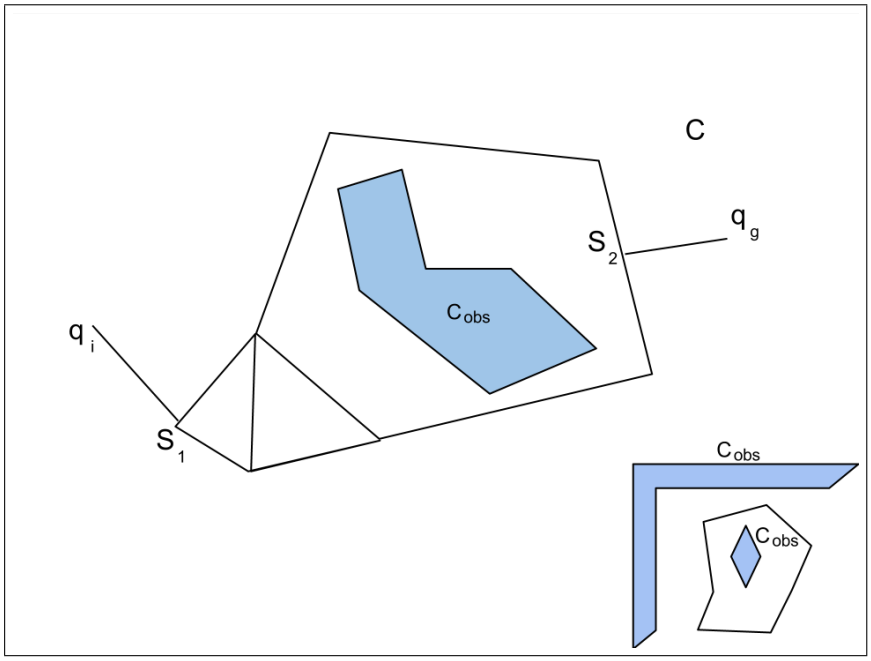

Figure 6: Example Swath connecting q_i and q_g

$$e_{i,j} \equiv \tau : [0,1] \rightarrow C_{free}$$

We define a Swath S to be

$$S = \bigcup_{e \in E} \tau([0,1])$$

which can be thought of as any point on the highway

Properties

1. **Accessability**
   From any $q \in C_{free}$ it is simple and efficient to compute a path to a Roadmap

2. **Connectivity Preserving**
   Given $q_i, q_g \in C_{free}$ we can find corresponding s_1, s_2 s.t. s_1 is the closest point to q_i on S and s_2 is the closest point to q_2 on S

Note: A new roadmap is needed for each island (Isolated area of C_{free})

2.3 Key Ideas

1. Turns a continuous problem into graph search

2. Creates a complex path through a concatenation of simple paths

In the case that q_i is close to q_g will find that the closest point to the highway will be the same for both & we will not use the highway.

Additionally, we should note that paths derived from the Roadmap method are suboptimal but are easy to compute.

2.4 Intuition

The roadmap method is similar to how a person would decide how to get from Pittsburgh to Chicago. One solution would be to look at all the possible paths using streets, backroads, thoroughfares, turnpikes, highways, and bike paths, which would be computationally intensive. Instead, we just look at the highways that are present between the two cities and figure out how to get to the highway from Pittsburgh, which highways to take to get near Chicago, then figure out how to get to your destination from the highway once arriving in Chicago.

3 Vertical Cell Decomposition

3.1 Overview

Vertical Cell Decomposition provides a simple and deterministic way to create roadmaps by exploiting the geometry of the problem.

3.2 Definition

To create our VCD we use the following definition Given:
C_{obs} = polygonal in $\mathbb{R}^2$
P = Set of all vertices
For each $p \in P$ extend rays $\uparrow$ & $\downarrow$
This creates two types of cells

- 1-cell: Line Segments
- 2-cell: Triangles or Trapezoids

Note: There is no limit to the number of obstacles in C_{obs}

Now we must define a new vertex for each cell, such as the midpoint of the 1-cells and 2-cells. Connecting them creates our roadmap and this roadmap is guaranteed to exist since triangles and trapezoids are convex and any 2 points can be joined by a straight line

$$q_i \to s_1 \rightsquigarrow s_2 \to q_g$$

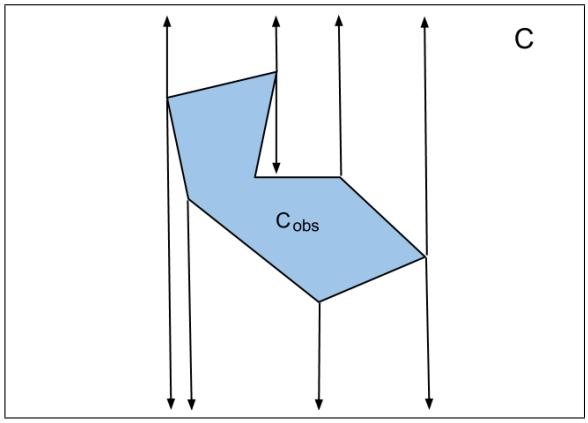

Figure 7: Extension of vertical rays from all verices

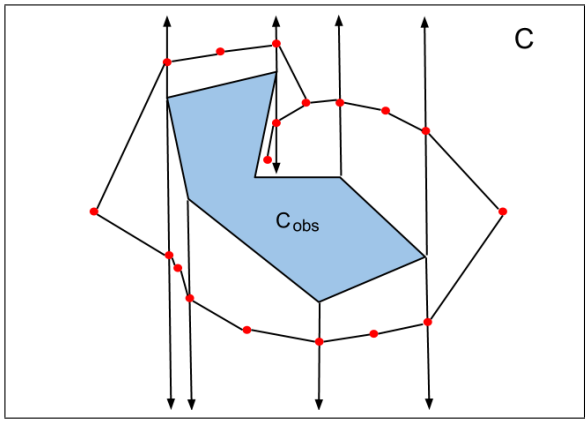

Figure 8: Vertices of cells marked and roadmap created

3.3 Extendability

Unfortunately this algorithm doesn't scale well in higher dimensions, it is possible to do in 3D but to do so we are simply making a stack of 2D planes and solving..

3.4 Intuition

Larger Roadmaps:

$\Longrightarrow$ create more optimal paths

$\Longrightarrow$ Planning goes back to discrete search speed

Taking the idea from our travelling from Pittsburgh to Chicago example, it is possible to create more Highways to get between the two to make the

path closer to ideal, but if we make every road a highway we end up back at our original problem of having to search too many paths.

4 Visibility Graphs

4.1 Overview

Instead of attempting to create a new roadmap and going out of our way to avoid obstacles, we can hug them more tightly in order to go around them to get from our initial state to our goal and the path will be closer to optimal.

4.2 Definition

Given:
C_{obs} = polygonal in $\mathbb{R}^2$
Vertices are all the vertices of obstacles in C_{obs} and are connected if:

- They are already connected by an edge in an obstacle
- The line segment joining them is in C_{free} [1]

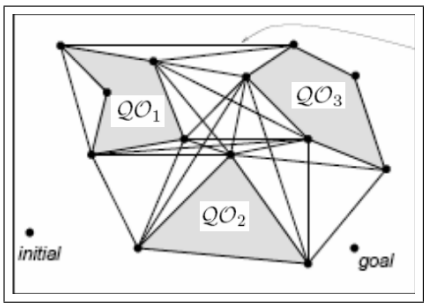

Figure 9: Example Visibility graph[1]

References

[1] Howie Choset. Robotic motion planning: Roadmap methods.

[2] Poornima and Anusha. Search in continuous space. *Scribing Lecture 10 16662 Robot Autonomy.*