B

EECS C106B / 206B Robotic Manipulation and Interaction **Spring 2022**

Lecture 5B: (Chapters 1 and 2 of MaSKS)

Scribes: Han Nguyen, Abhinav Singh, Ryan Adolf

5.1 Recap of Last Lecture and Guiding Principles

We have previously done a lot of robotics dynamics and control theory, but it's important to think about the bigger context in which this is all happening for students who are now near graduation and heading to the industry. It's time to think - **what is the past, current, and probably future for robotics (different methodology, what is common, what is different, etc.)?**

Robotics is a very broad area designed for many different purposes. High-end customers (military, NASA) desire many objectives with robots (precision/safety guarantees/accuracy/robustness). In the recent era, many consumer robots have started to arrive. We no longer need high performance helicopters under military action, but a lot of drones that can be reasonable accurate and safe.

5.2 Current Methods

Last time, we saw the classical system modeling and control design vs. modern techniques (reinforcement learning (RL), etc.) Some things we have learned like parallel parking and motion planning with nonholonomic constraint can be refined with different neural networks, more dissemination, and more testing; but it is hard to expect what you will get. Even from Valmik's lecture slides, we can see that with reinforcement learning, there is more energy consumption than control with sinusoids. From current research, we have found that even with LQR, we don't have an advantage. State-of-the-art RL has found to only be as good as random search. Everything we learn here is to try and reduce the cost.

This begs the question: if the method is brute force (which is what RL is doing), how are we able to solve a problem with a humongous state-space? Where does that efficiency come from? If it's a general problem in that space, there is no hope, since we will have to sample that entire space. Why is it that with reasonable sampling, the sampling required is disproportionately less than the perceived complexity of the problem? This actually comes from the structure of the problem. The structure of the problem should be leveraged to help us find solutions.

The optimal structure actually only depends on a very small fraction of the information of the state. The perceived efficiency should come from the fact that the problems have tremendous structures which can be exploited. The way to solve the problem is by diligence, not machine intelligence.

Innovation these days have become mainly about the terminology, not enough about the methodology. Imitation learning, inverse RL, teacher/student, are just all different names for similar processes (costs become rewards in an inverse problem, etc.) Now, we are looking forward to how to generalize across long term and short term goals in robotic systems. This is done by breaking tasks into different granularity and approach them one by one. These machine learning algorithms work by exploiting low-dimensionality. The machines can learn via imitation and using hierachiral design to solve the problem. However, RL rarely cares

about cost, especially for examples like games where we just care about winning. As long as we converge with quantitative measures (time, energy, precision), we are done. We don't care as much about quantitative things (stability, survivability). Is this the best result?

Much of current practice is not about the methodologies or principles but practice of the principles - leveraging the computational and data resources. The classical controls is very good when you have a good model. Closed form analysis, optimization constraints, and precision can be done explicitly. You can afford a little bit of brute force search or deal with model mismatch and find a policy that works. However, if the underlying dynamics, optimal policy, and rewards are structured, how do we understand them? How many resources do I need to achieve a task to a certain degree of precision? How many samples do I need? When I converge to something, how do I determine if it is good enough?

Another large topic in robotics is in adaptation or generalization. Nowadays, robots can do very good in a laboratory condition, but can fail in a different environment. Robots can emulate humans very well, but how would they perform on their own? What does this mean for robots to be flexible? They can do a very specific task very well, but what about comprehensive decisions and tasks? Will service robots at NASA always have the same jobs?

5.3 Bridge between Principles and Practices

The things we work on in the future will be some combination of both. Controls can benefit from high performance computing, and to do that, we need to bring the power of computation and data resources to practice. **Practice keeps theory honest, and vice versa.**

We can consider the classic loop for control design: *Loop: Perception* $\rightarrow$ *Learning* $\rightarrow$ *Decision-making*

Through this loop, we can test how well we know the environment and the system, as well as if we can predict the behavior and make the right decision given the task/reward. This optimal reaction is where control design comes in. In order for the control design to be effective and get the expected result, we must make sure that our system can output the right data. **Therefore, the bridge between principles and the real world (practices), is computation).**

5.4 An Invitation to 3D Vision

This textbook is linked on Piazza and will be used for this class. There is more comprehensive work in the book, Computer Vision: Algorithms and Applications by Richard Szeliski

Motivation: If a robot is going to pick an object up, it needs to know how big that object is and what it's location is. Autonomous robots don't have someone to tell them this information, so they need to perceive it themselves!

We know a lot about controls theory, but what about feedback? Our controls design will not be effective without proper feedback from vision.

Fundamental Problem - Localization: We perceive 2-D images of the world and are able to navigate and reconstruct what's going on.

- How do we get that information from 2-D to 3-D in robots, despite all the unknowns and uncertainties about our environment?
- How do we calibrate?

- How is it that none of our vision are distorted (everyone agrees that a circle is a circle)?
- How do we survive in our environment without bumping into each other all the time?

For the robot vision problem, the output we want is our pose, position and location. Then, this begs the questions:

- What are the principles that enable a robot to "see" and interact with their environment from just 2-D images?
- How do we represent the 3-D world mathematically, graphically and computationally?

For robotic interaction, the most important thing is to understand the geometry and layout of the environment. There are different methods of representing the real world, such as point clouds, meshes, voxels, and implicit surfaces. They are all ways of representing the real world with different precision and granularity, and are each for different purposes.

5.5 Multi-View Geometry

Localization is an extremely important problem for robotics: the robot needs to know where it is and what's in its way. We humans are very good at this task; we can take a look around and figure out how far we are from the buildings and natural elements in our way. We are also extremely well calibrated in this task. If multiple people see a circle on the ground, they will likely all say it's a perfect circle. No one would think it is an ellipse.

For a robot this task is more difficult. In EECS 106A we studied the matrices that, given the camera's location and parameters, describe the transformations from the real world to the camera. However, we are interested in the inverse problem: From multiple perspective images, we want to know where that camera and its attached robot are located. This technique is called structure from motion or simultaneous localization and mapping (SLAM). It's part of the field of multi-view geometry, which seeks to understand the geometric relationships among different views of the same points, lines, and planes in order to reconstruct the 3-D world.

Formally, we can define this problem as an algorithm with an input and output.

Input: Multiple perspective images, with features
Output: Camera poses, calibration, scene structure representations

Some popular representations of scene structure are illustrated in figure 5.1.

5.5.1 Applications

PATH: In the 1990s, California tried to develop autonomous driving cars. The US lagged behind Japan and Germany when it came to developing autonomous driving. The PATH project detected lane features from images the car took of the road. At the time, technology was very primitive. The first camera used for this task had a resolution of 640x480 pixels, and there were no GPUs used for the vision algorithms. Professor Ma's first paper was on how to design the feedback controller to keep the car inside its own lane. Once it was shown autonomous lane keeping had potential, the project lost its funding, and industry caught up with the research.

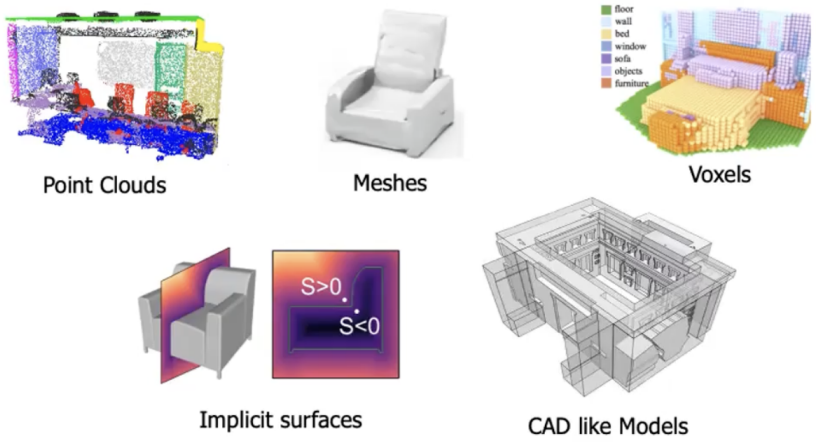


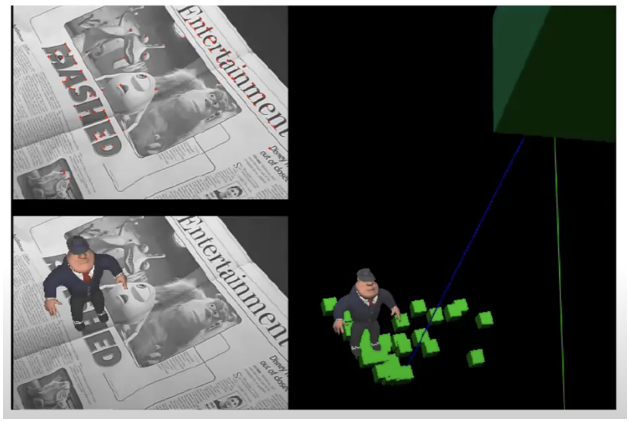
Figure 5.1: Popular representations of scene structure.

Today's autonomous cars: Waymo, Tesla, and others are now working on reconstructing road geometry and writing control algorithms so their cars can autonomously navigate.

Unmanned Aerial Vehicles: Helicopters are much harder to control than planes and present many interesting control challenges with bad weather (and dangers for pilots!) Professor Ma was in charge of the vision part of figuring out the pose of the helicopter relative to the landing pad with an onboard camera and calibrating the camera.

Drones (today's UAVs): Localization is extremeley important for drones, so that they don't crash into buidings or other drones. Today, drones can perform in tightly coordinated light shows and performances due to the precision of their localization.

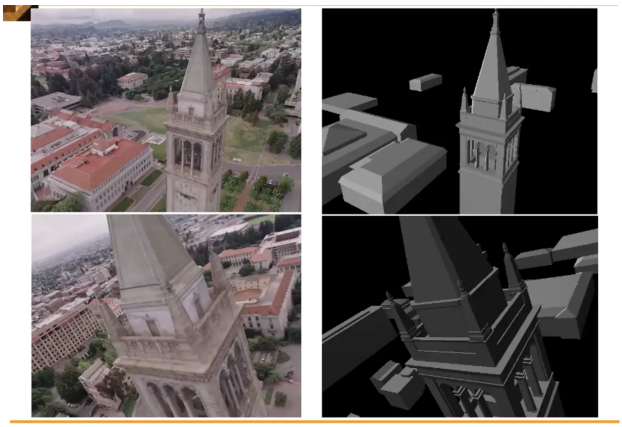
Real-time virtual object insertion: We can insert virtual objects into footage taken from a camera (augmented reality). This requires recovering the post of the camera from the footage. We can also insert virtual advertisements, because money. For some TV football games, the ads on the football field use camera 3D vision to figure out where to place the ads so it looks like they are part of the real field.



Virtual Museums: Multi-camera light stages can capture images of an object at many angles and then render a mesh of that object in 3D. One of these was famously used to scan Obama's face. We can also use consumer AR/VR kits (or phones!) to scan objects.

Virtual performances: Cameras can capture performers and reconstruct them in 3D to deliver the performance virtually.

Image-based modeling and reconstruction In 1996, Paul Debevec was interested in building a 3D model/map of campus (later went on to do models for The Matrix and Obama and win an oscar). He used many images he took of the campanile to reconstruct its geometry.



5.5.2 Calibration

The three most important things in robotics are calibration, calibration, and calibration.

If your camera calibration is incorrect, your vision algorithms will give the wrong results!

5.5.3 General Steps

5.5.3.1 Feature Selection and Correspondence

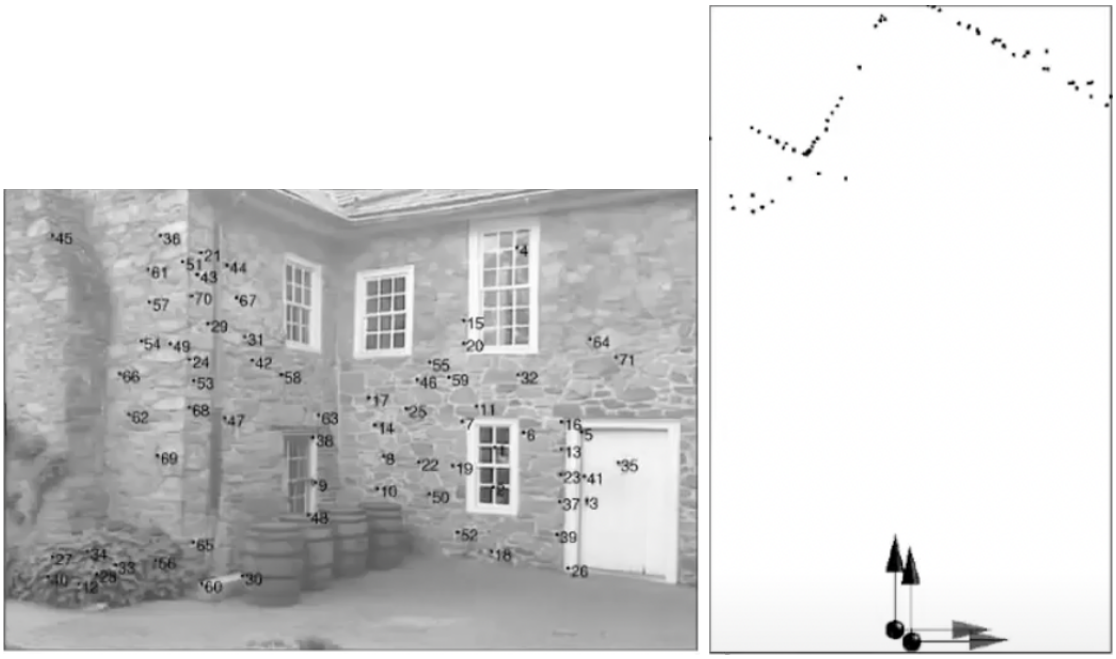
Given all of the above applications, how do we start to do all of this? We first start by having a stream of cameras that track features from multiple angles. In the below example, the corner of a window (boxed) is used as a feature point. Lines can be used as features in addition to points. In addition, it is important to consider the difference between small and large baselines.

5.5.3.2 Structure and Motion Recovery/3-D Surface Model and Rendering

After tracking the feature points across frames, we can use them to build models of the 3D world. By doing this, we can calculate where the camera is relative to the points, and where the points are relative to the camera. This is a very important problem to solve in all computer vision applications.

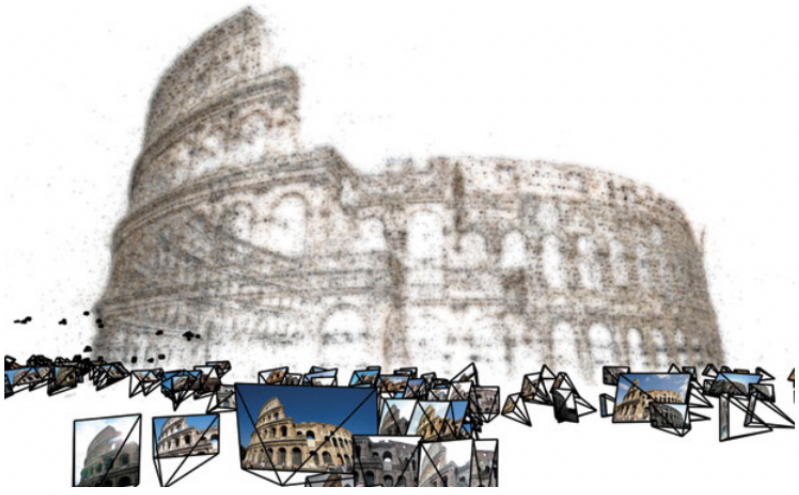
Some variables to consider:

- Two versus multiple views
- Discrete versus continuous motions
- General versus planar scenes
- Calibrated versus uncalibrated cameras
- One motion versus multiple motions



5.5.4 Example: Image-Based 3-D Modeling

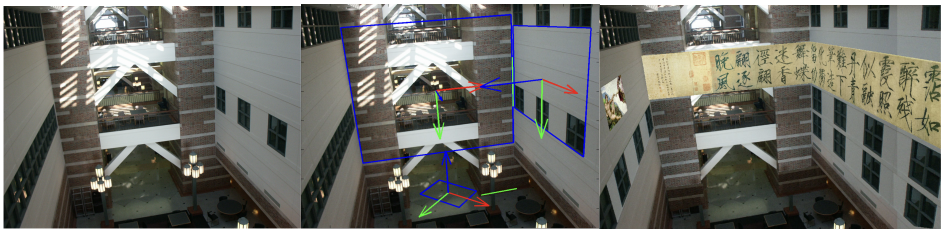
You can then build a 3-D surface model and render the images on the model using point clouds or other methods. Around 2016/17, Microsoft employees working with a professor at the University of Washington attempted to model Rome using tourist images from Flickr. The preparation of the project took a year.



The Colosseum, 2,106 images

5.5.5 Symmetry-Based Modeling and Reconstruction

Humans can still do better. Even within a single image, humans make an almost perfect reconstruction of the 3-D geometry encompassed inside. Once you understand the 3-D geometry inside an image, you can augment it and play around with it by finding the planes inside the image and painting along the planes.



Humans are great at this and machines aren't. Take the example of a classroom with stairs and lots of seats. Things that help a human navigate through the room (lines, symmetry, similar objects, stairs, etc.) are things that compromise the vision of a robot. Our brains can do augmented reality like crazy, but it is a nightmare for machines.

Although right now we are limited in our ability to utilize the geometry of images, the potential is unlimited in the future once we are able to extract the right features. We might even be able to replicate the same precision that humans have.

5.6 Rigid Body Motion and Geometry of Image Formation

These are the fundamental mathematical principles of recovering the geometry/3-D information from 2-D images. We will see how to do it and under what conditions we can do it.

5.6.1 3-D Euclidean Space

5.6.1.1 Cartesian Coordinate Frame

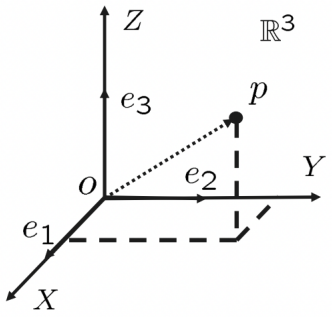
We will start by introducing the notation of rigid body motion. Given a XYZ coordinate frame, we can express its standard basis vectors as:

$$e_1 = \begin{bmatrix}1\\0\\0\end{bmatrix} \; e_2 = \begin{bmatrix}0\\1\\0\end{bmatrix} \; e_3 = \begin{bmatrix}0\\0\\1\end{bmatrix} \tag{5.1}$$

The coordinates of a point p in this space is given by:

$$\mathbf{X}_p = \begin{bmatrix}X\\Y\\Z\end{bmatrix} \in \mathbb{R}^3 \tag{5.2}$$

A frame with some arbitrary point p and the standard basis vectors is shown below:



5.6.1.2 Vectors

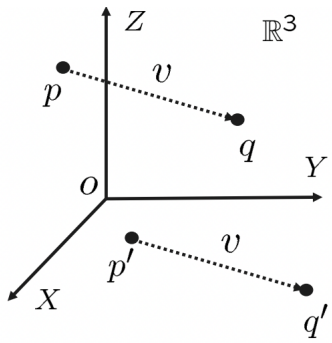
We define a "free" vector as a pair of points (p, q):

$$\mathbf{X}_p = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} \in \mathbb{R}^3, \mathbf{X}_q = \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} \in \mathbb{R}^3 \tag{5.3}$$

The coordinates of this vector v is the difference between these two points:

$$v = \mathbf{X}_p - \mathbf{X}_q = \begin{bmatrix} X_1 - X_2 \\ Y_1 - Y_2 \\ Z_1 - Z_2 \end{bmatrix} \in \mathbb{R}^3 \tag{5.4}$$

A vector v is "free" because it is not confined to any specific point in the space, as shown below. It essentially encodes the direction from one point to another. Vectors and points are two different physical entities.



5.6.1.3 Inner Product and Cross Product

We can use vectors to measure distances and angles in Euclidean space. If we have two vectors u and v defined as

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{5.5}$$

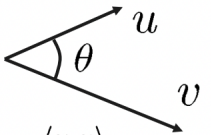
then the inner product between the two vectors is given as

$$\langle u, v \rangle = u^T v = u_1 v_1 + u_2 v_2 + u_3 v_3 \tag{5.6}$$

We can use the inner product for a multitude of things. One is the norm, or magnitude of a vector, given as

$$||u|| = \sqrt{\langle u, u \rangle} = \sqrt{u^T u} = \sqrt{u_1^2 + u_2^2 + u_3^2} \tag{5.7}$$

We can also use it to measure the angle between two vectors. Given vectors u and v and some angle θ as shown below



we find that

$$cos(\theta) = \frac{\langle u, v \rangle}{||u||||v||} \tag{5.8}$$

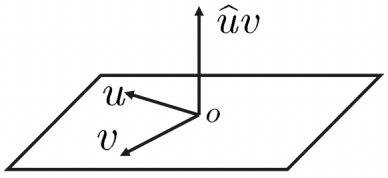
We also define the cross product between two vectors u and v as

$$u \times v = \hat{u}v \quad u, v \in \mathbb{R}^3 \tag{5.9}$$

where $\hat{u}$ is the skew-symmetric matrix associated with u:

$$\hat{u} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3} \tag{5.10}$$

The result of the cross product $\hat{u}v$ is a vector that is orthogonal to the plane spanned by u and v

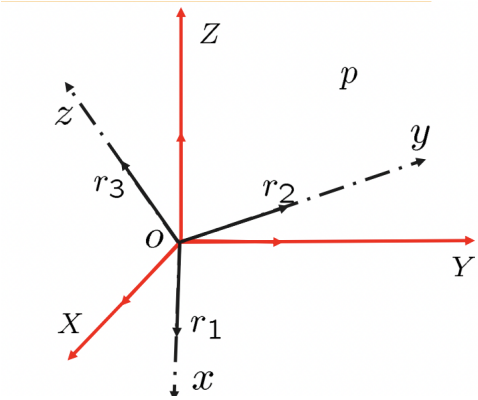


5.6.1.4 Rotation

We can also describe rotations in Euclidean space via the use of rotation matrices. Given two frames, a world frame and a camera frame, we can relate the coordinates of a point in the world frame to that same point in the camera frame via the rotation matrix. A rotation matrix R is defined as:

$$R = [r_1, r_2, r_3] \in \mathbb{R}^{3\times 3} \tag{5.11}$$

where r_1, r_2, r_3 are the standard basis vectors of the camera frame expressed in the world frame as shown below.



The coordinates of a point in the world frame versus the camera frame are related by

$$\mathbf{X}_c = R\mathbf{X}_w \tag{5.12}$$

A rotation matrix must satisfy certain properties. It must be an orthogonal matrix whose determinant is 1:

$$R^T R = I, det(R) = +1 \tag{5.13}$$

Matrices of this form belong to the Special Orthogonal group ($SO(3)$) defined as follows

$$SO(3) = \{R \in \mathbb{R}^{3\times 3} | R^T R = RR^T = I, \det R = 1\} \tag{5.14}$$