# 6.801/6.866: Machine Vision, Quiz 2 Review

Professor Berthold Horn, Ryan Sander, Tadayuki Yoshitake
MIT Department of Electrical Engineering and Computer Science
Fall 2020

These lecture summaries are designed to be a review of the lecture. Though I do my best to include all main topics from the lecture, the lectures will have more elaborated explanations than these notes. Therefore, if you're looking for the most rigorous review and treatment of these topics, we encourage you to rewatch the lecture videos. With that said, we hope these summaries are beneficial for your learning. If you have any feedback for these lecture summaries, please submit it **here**.

Here you will find a high-level review of some of the topics covered since Quiz 1. These are as follows in the section notes:

1. **Relevant Mathematics Review** - Rayleigh Quotients, Groups, Levenberg-Marquadt, Bezout's Theorem
2. **Systems/Patents** - PatQuick, PatMAx, Fast Convolutions, Hough Transforms
3. **Signals** - Sampling, Spline Interpolation, Integral Images, Repeated Block Averaging
4. **Photogrammetry** - Photogrammetry Problems, Absolute orientation, Relative Orientation, Interior Orientation, Exterior Orientation, Critical Surfaces, Radial and Tangential Distortion
5. **Rotations/Orientation** - Gibb's vector, Rodrigues Formula, Quaternions, Orthonormal Matrices
6. **3D Recognition** - Extended Gaussian Image, Solids of Revolution, Polyhedral Objects, Desirable Properties

# 1 Relevant Mathematics Review

We'll start with a review of some of the relevant mathematical tools we have relied on in the second part of the course.

## 1.1 Rayleigh Quotients

We saw from our lectures with absolute orientation that this is a simpler way (relative to Lagrange Multipliers) to take constraints into account:

The intuitive idea behind them: How do I prevent my parameters from becoming too large) positive or negative) or too small (zero)? We can accomplish this by dividing our objective by our parameters, in this case our constraint. In this case, with the Rayleigh Quotient taken into account, our objective becomes:

$$\min_{\overset{o}{q},\overset{o}{q}\cdot\overset{o}{q}=1} \overset{o}{q}^T N \overset{o}{q} \longrightarrow \min_{\overset{o}{q}} \frac{\overset{o}{q}^T N \overset{o}{q}}{\overset{o}{q}^T \overset{o}{q}}$$

## 1.2 (Optional) Groups

Though we don't cover them specifically in this course, **groups** can be helpful for better understanding rotations. "In mathematics, a group is a set equipped with a binary operation that combines any two elements to form a third element in such a way that four conditions called group axioms are satisfied, namely closure, associativity, identity and invertibility." [1]. We won't focus too much on the mathematical details, but being aware of the following groups may be helpful when reading machine vision, robotics, control, or vision papers:

- Orthonormal Rotation Matrices $\rightarrow \mathbf{SO}(3) \in \mathbb{R}^{3\times3}$, **Special Orthogonal Group**. Note that the "3" refers to the number of dimensions, which may vary depending on the domain.
- Poses (Translation + Rotation Matrices) $\rightarrow \mathbf{SE}(3) \in \mathbb{R}^{4\times4}$, **Special Euclidean Group**.

Since these groups do not span standard Euclidean spaces (it turns out these groups are both manifolds - though this is not generally true with groups), we cannot apply standard calculus-based optimization techniques to solve for them, as we have seen to be the case in our photogrammetry lectures.

### 1.3 (Optional) Levenberg-Marquadt and Gauss-Newton Nonlinear Optimization

Recall this was a nonlinear optimization technique we saw to solve photogrammetry problems such as Bundle Adjustment (BA).

Levenberg-Marquadt (LM) and Gauss-Newton (GN) are two nonlinear optimization procedures used for deriving solutions to nonlinear least squares problems. These two approaches are largely the same, except that LM uses an additional **regularization term** to ensure that a solution exists by making the closed-form matrix to invert in the normal equations positive semidefinite. The normal equations, which derive the closed-form solutions for GN and LM, are given by:

1. **GN**: $(J(\theta)^T J(\theta))^{-1}\theta = J(\theta)^T e(\theta) \implies \theta = (J(\theta)^T J(\theta))^{-1} J(\theta)^T e(\theta)$
2. **LM**: $(J(\theta)^T J(\theta) + \lambda I)^{-1}\theta = J(\theta)^T e(\theta) \implies \theta = (J(\theta)^T J(\theta) + \lambda I)^{-1} J(\theta)^T e(\theta)$

Where:

- $\theta$ is the vector of parameters and our solution point to this nonlinear optimization problem.
- $J(\theta)$ is the Jacobian of the nonlinear objective we seek to optimize.
- $e(\theta)$ is the residual function of the objective evaluated with the current set of parameters.

Note the $\lambda I$, or regularization term, in Levenberg-Marquadt. If you're familiar with ridge regression, LM is effectively ridge regression/regression with L2 regularization for nonlinear optimization problems. Often, these approaches are solved iteratively using gradient descent:

1. **GN**: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha(J(\theta^{(\mathbf{t})})^T J(\theta^{(\mathbf{t})}))^{-1} J(\theta^{(\mathbf{t})})^T e(\theta^{(\mathbf{t})})$
2. **LM**: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha(J(\theta^{(\mathbf{t})})^T J(\theta^{(\mathbf{t})}) + \lambda I)^{-1} J(\theta^{(\mathbf{t})})^T e(\theta^{(\mathbf{t})})$

Where $\alpha$ is the step size, which dictates how quickly the estimates of our approaches update.

### 1.4 Bezout's Theorem

Though you're probably well-versed with this thereom by now, its importance is paramount for understanding the number of solutions we are faced with when we solve our systems:

Theorem: *The maximum number of solutions is the product of the polynomial order of each equation in the system of equations*:

$$\text{number of solutions} = \prod_{e=1}^{E} o_e$$

## 2 Systems

In this section, we'll review some of the systems we covered in this course through patents, namely PatQuick, PatMAx, and Fast Convolutions. A block diagram showing how we can cascade the edge detection systems we studied in this class can be found below:

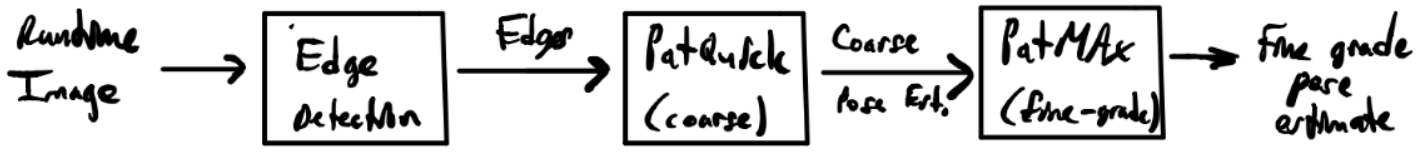


Figure 1: An overview of how the patents we have looked at for object inspection fit together.

### 2.1 PatQuick

There were three main "objects" in this model:

- **Training/template image**. This produces a model consisting of probe points.
- **Model**, containing probe points.

- **Probe points**, which encode evidence for where to make gradient comparisons, i.e. to determine how good matches between the template image and the runtime image under the current pose configuration.

Once we have the model from the training step, we can summarize the process for generating matches as:

1. Loop over/sample from configurations of the pose space (which is determined and parameterized by our degrees of freedom), and modify the runtime image according to the current pose configuration.
2. Using the probe points of the model, compare the gradient direction (or magnitude, depending on the scoring function) to the gradient direction (magnitude) of the runtime image under the current configuration, and score using one of the scoring functions below.
3. Running this for all/all sampled pose configurations from the pose space produces a multidimensional scoring surface. We can find matches by looking for peak values in this surface.

## 2.2 PatMAx

- This framework builds off of the previous PatQuick patent.
- This framework, unlike PatQuick, does not perform quantization of the pose space, which is one key factor in enabling sub-pixel accuracy.
- PatMAx assumes we already have an approximate initial estimate of the pose.
- PatMAx relies on an iterative process for optimizing energy, and each **attraction step** improves the fit of the configuration.
- Another motivation for the name of this patent is based off of electrostatic components, namely dipoles, from Maxwell. As it turns out, however, this analogy works better with mechanical springs than with electrostatic dipoles.
- PatMAx performs an **iterative attraction process** to obtain an estimate of the pose.
- An iterative approach (e.g. gradient descent, Gauss-Newton, Levenberg-Marquadt) is taken because we likely will not have a closed-form solution in the real world. Rather than solving for a closed-form solution, we will run this iterative optimization procedure until we reach convergence.

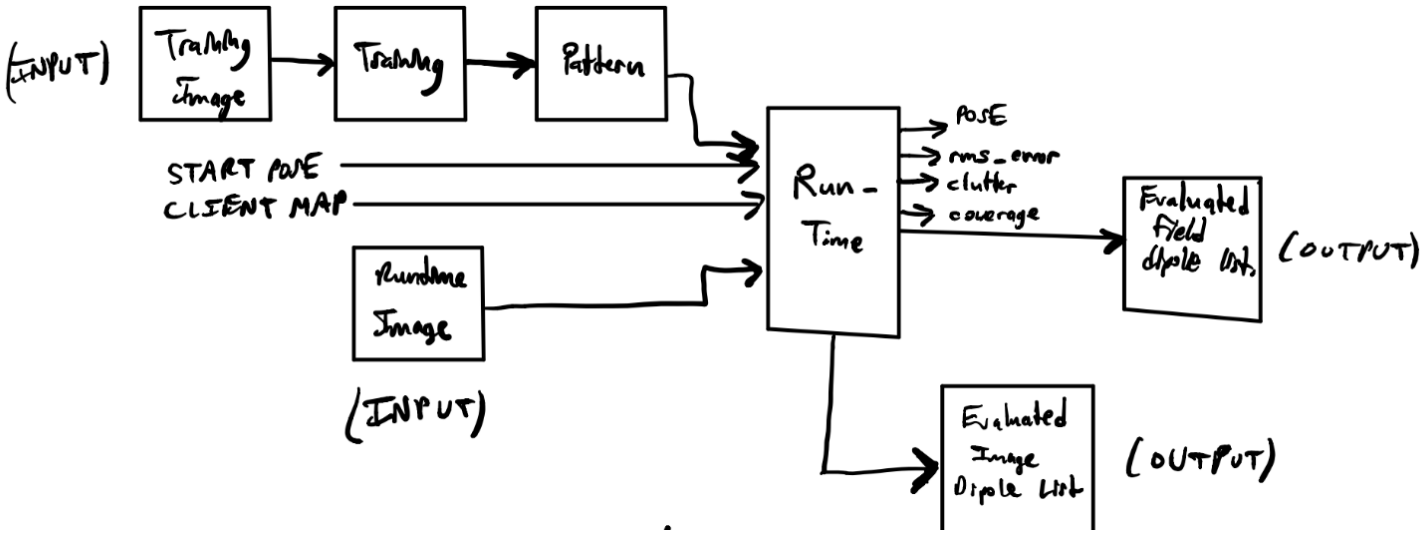


Figure 2: High-level diagram of the PatMAx system.

## 2.3 Fast Convolutions

The patent we explore is **"Efficient Flexible Digital Filtering, US 6.457,032**.

The goal of this system is to efficiently compute filters for multiscale. For this, we assume the form of an $N^{th}$-order piecewise polynomial, i.e. a $N^{th}$-order spline.

### 2.3.1 System Overview

The block diagram of this system can be found below:

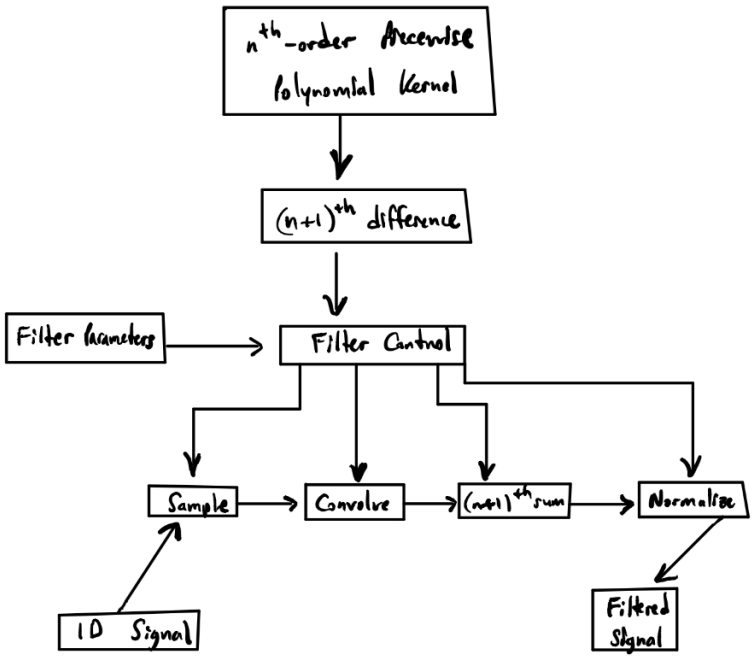


Figure 3: Block diagram of this sparse/fast convolution framework for digital filtering. Note that this can be viewed as a compression problem, in which differencing compresses the signal, and summing decompresses the signal.

A few notes on this system:

- Why is it of interest, if we have $N^{th}$-order splines as our functions, to take $N^{th}$-order differences? The reason for this is that the differences create **sparsity**, which is critical for fast and efficient convolution. Sparsity is ensured because:

$$\frac{d^{N+1}}{dx^{N+1}} f(x) = 0 \; \forall \; x \text{ if } f(x) = \sum_{i=0}^{N} a_i x^i, a_i \in \mathbb{R} \;\; \forall \; i \in \{1, \cdots, N\}$$

(I.e, if $f(x)$ is a order-N polynomial, then the order-(N+1) difference will be 0 for all x.

This sparse structure makes convolutions much easier and more efficient to compute by reducing the size/cardinality of the support.

- Why do we apply an order-(N+1) summing operator? We apply this because we need to "invert" the effects of the order-(N+1) difference:

$$\begin{aligned}
&\textbf{First Order}: \; \mathcal{DS} = I \\
&\textbf{Second Order}: \; \mathcal{DDSS} = \mathcal{DSDS} = (\mathcal{DS})(\mathcal{DS}) = II = I \\
&\vdots \\
&\textbf{Order K}: \; (\mathcal{D})^K(\mathcal{S})^K = (\mathcal{DS})^K = I^K = I
\end{aligned}$$

## 2.4 Hough Transforms

**Motivation**: Edge and line detection for industrial machine vision; we are looking for lines in images, but our gradient-based methods may not necessarily work, e.g. due to non-contiguous lines that have "bubbles" or other discontinuities. These discontinuities can show up especially for smaller resolution levels.

**Idea**: The main idea of the **Hough Transform** is to intelligently map from image/surface space to parameter space for that surface.

**Some notes on Hough Transforms**:

- Often used as a subroutine in many other machine vision algorithms.
- Actually generalize beyond edge and line detection, and extend more generally into any domain in which we map a parameterized surface in image space into parameter space in order to estimate parameters.

**Example: Hough Transforms with Lines** A line/edge in image space can be expressed (in two-dimensions for now, just for building intuition, but this framework is amenable for broader generalizations into higher-dimensional lines/planes): $y = mx + c$. Note that because $y = mx + c, m = \frac{y-c}{x}$ and $c = y - mx$. Therefore, this necessitates that:

- A line in image space maps to a singular point in Hough parameter space.
- A singular point in line space corresponds to a line in Hough parameter space.

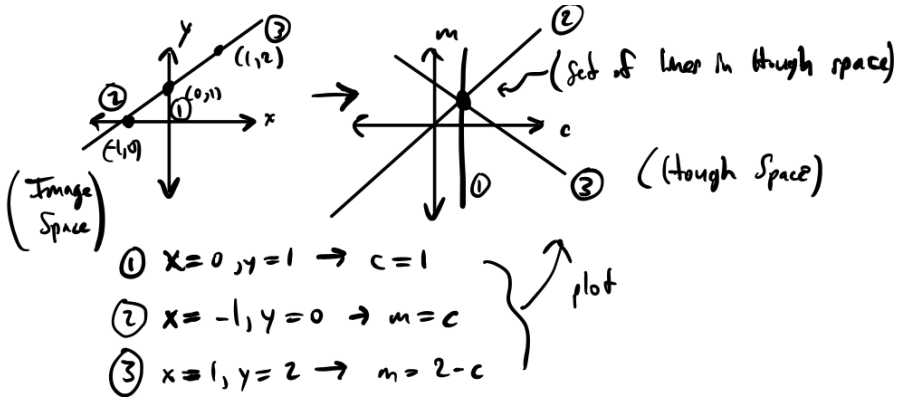


Figure 4: Example of finding parameters in Hough Space via the Hough Transform.

# 3 Photogrammetry

Given the length of the mathematical derivations in these sections, we invite you to revisit notes in lectures 17-21 for a more formal treatment of these topics. In this review, we hope to provide you with strong intuition about different classes of photogrammetric problems and their solutions.

Four important problems in photogrammetry that we covered in this course:

- **Absolute Orientation** $3D \longleftrightarrow 3D$, Range Data
- **Relative Orientation** $2D \longleftrightarrow 2D$, Binocular Stereo
- **Exterior Orientation** $2D \longleftrightarrow 3D$, Passive Navigation
- **Intrinsic Orientation** $3D \longleftrightarrow 2D$, Camera Calibration

Below we discuss each of these problems at a high level. We will be discussing these problems in greater depth later in this and following lectures.

## 3.1 Absolute Orientation

We will start with covering **absolute orientation**. This problem asks about the relationship between two or more objects (cameras, points, other sensors) in 3D. Some examples of this problem include:

1. Given two 3D sensors, such as lidar (light detection and ranging) sensors, our goal is to find the **transformation**, or **pose**, between these two sensors.
2. Given one 3D sensor, such as a lidar sensor, and two objects (note that this could be two distinct objects at a single point in time, or a single object at two distinct points in time), our goal is to find the **transformation**, or **pose**, between the two objects.

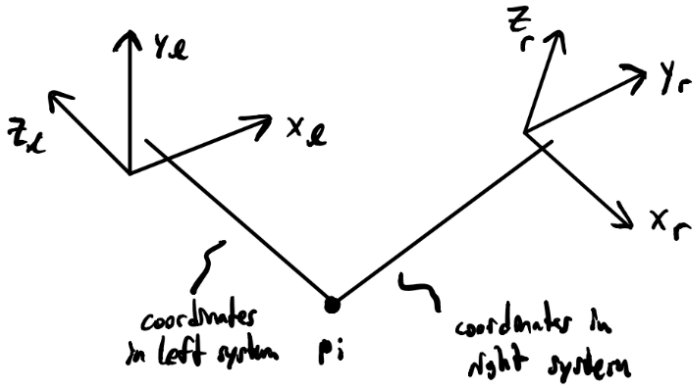


Figure 5: General case of absolute orientation: Given the coordinate systems $(\mathbf{x}_l, \mathbf{y}_l, \mathbf{z}_l) \in \mathbb{R}^{3\times3}$ and $(\mathbf{x}_r, \mathbf{y}_r, \mathbf{z}_r) \in \mathbb{R}^{3\times3}$, our goal is to find the transformation, or pose, between them using points measured in each frame of reference $\mathbf{p}_i$.

### 3.2 Relative Orientation

This problem asks how we can find the 2D $\longleftrightarrow$ 2D relationship between two objects, such as cameras, points, or other sensors. This type of problem comes up frequently in machine vision, for instance, **binocular stereo**. Two high-level applications include:

1. Given two cameras/images that these cameras take, our goal is to extract 3D information by finding the relationship between two 2D images.
2. Given two cameras, our goal is to find the (relative) **transformation**, or **pose**, between the two cameras.

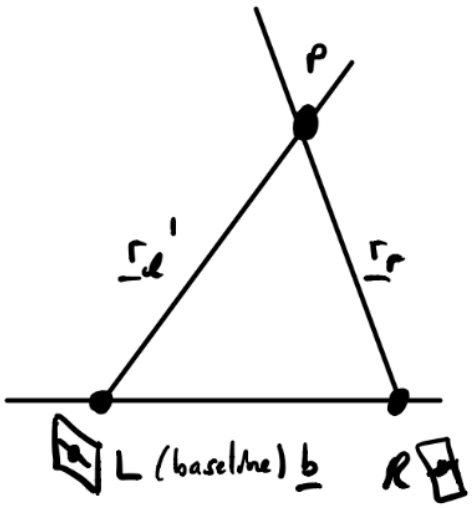


Figure 6: Binocular stereo system set up. For this problem, recall that one of our objectives is to measure the translation, or baseline, between the two cameras.

### 3.3 Exterior Orientation

This photogrammetry problem aims from going 2D $\longrightarrow$ 3D. One common example in robotics (and other field related to machine vision) is **localization**, in which a robotic agent must find their location/orientation on a map given 2D information from a camera (as well as, possibly, 3D laser scan measurements).

More generally, with **localization**, our goal is to find where we are and how we are oriented in space given a 2D image and a 3D model of the world.

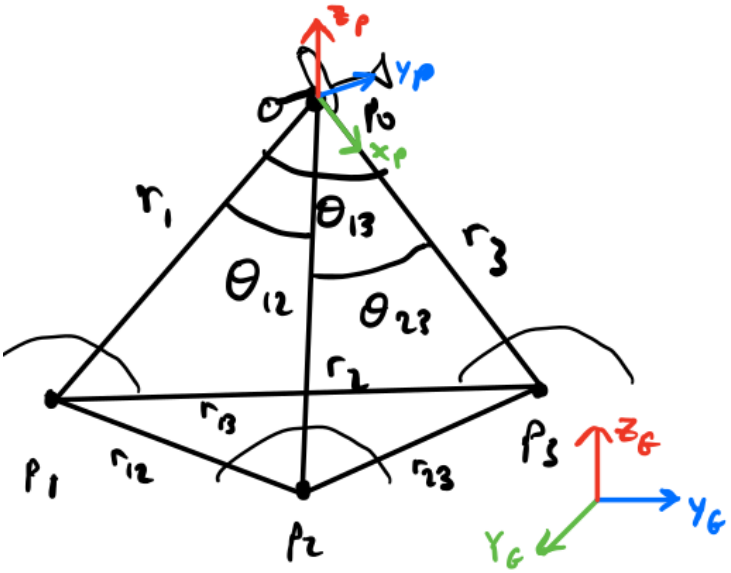


Figure 7: Exterior orientation example: Determining position and orientation from a plane using a camera and landmark observations on the ground.

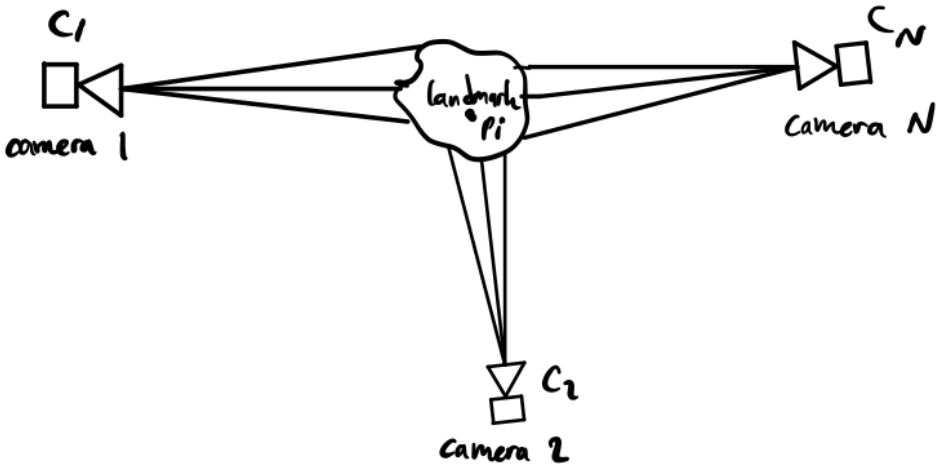


Figure 8: Bundle Adjustment (BA) is another problem class that relies on exterior orientation: we seek to find the orientation of cameras using image location of landmarks. In the general case, we can have any number of $K$ landmark points ("interesting" points in the image) and $N$ cameras that observe the landmarks.

### 3.4 Interior Orientation

This photogrammetry problem aims from going 3D $\longrightarrow$ 2D. The most common application of this problem is **camera calibration**. Camera calibration is crucial for high-precision imaging, as well as solving machine and computer vision problems such as Bundle Adjustment [1]. Finding a principal point is another example of the interior orientation problem.

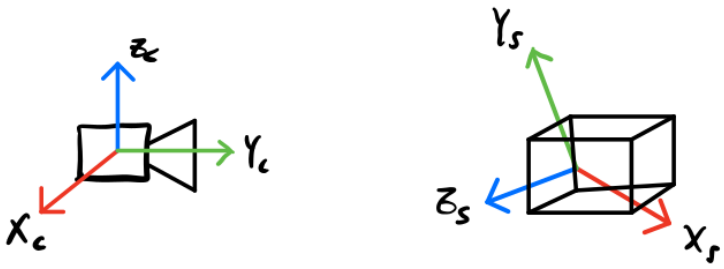


Figure 9: Interior orientation seeks to find the transformation between a camera and a calibration object - a task often known as camera calibration. This can be used, for instance, with Tsai's calibration method (note that this method also relies on exterior orientation).

# 4 Rotation

There are a myriad of representations for rotations - some of these representations include:

1. Axis and angle
2. Euler Angles
3. Orthonormal Matrices
4. Exponential cross product
5. Stereography plus bilinear complex map
6. Pauli Spin Matrices
7. Euler Parameters
8. Unit Quaternions

We would also like our representations to have the following properties:

- The ability to rotate vectors - or coordinate systems
- The ability to compose rotations
- Intuitive, non-redundant representation - e.g. rotation matrices have 9 entries but only 3 degrees of freedom
- Computational efficiency
- Interpolate orientations
- Averages over range of rotations
- Derivative with respect to rotations - e.g. for optimization and least squares
- Sampling of rotations - uniform and random (e.g. if we do not have access to closed-form solutions)
- Notion of a space of rotations

Let us delve into some of these in a little more depth.

## 4.1 Axis and Angle

This representation is composed of a vector $\hat{\boldsymbol{\omega}}$ and an angle $\theta$, along with the **Gibb's vector** that combines these given by $\hat{\boldsymbol{\omega}} \tan\left(\frac{\theta}{2}\right)$, which has magnitude $\tan\left(\frac{\theta}{2}\right)$, providing the system with an additional degree of freedom that is not afforded by unit vectors. Therefore, we have our full 3 rotational DOF.

## 4.2 Orthonormal Matrices

We have studied these previously, but these are the matrices that have the following properties:

1. $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = I, \mathbf{R}^T = \mathbf{R}^{-1}$ (skew-symmetric)
2. $\det|R| = +1$
3. $\mathbf{R} \in \mathbf{SO}(3)$ (see notes on groups above) - being a member of this Special Orthogonal group is contingent on satisfying the properties above.

## 4.3 Quaternions

In this section, we will discuss another way to represent rotations: quaternions.

### 4.3.1 Hamilton and Division Algebras

Hamilton's insight with quaternions is that these quaternions require a "4th dimension for the sole purpose of calculating triples". Hamilton noted the following insights in order to formalize his quaternion. Therefore, the complex components $i, j, k$ defined have the following properties:

1. $i^2 = j^2 = k^2 = ijk = -1$
2. From which follows:
   (a) $ij = k$
   (b) $jk = i$
   (c) $ki = j$
   (d) $ji = -k$
   (e) $kj = -i$
   (f) $ik = -j$

   **Note**: As you can see from these properties, multiplication of the components of these quaternions is not commutative.

We largely use the 4-vector quaternion, denoted $\overset{o}{q} = (q, \mathbf{q}) \in \mathbb{R}^4$.

## 4.4 Properties of 4-Vector Quaternions

These properties will be useful for representing vectors and operators such as rotation later:

1. **Not commutative**: $\overset{o}{p}\overset{o}{q} \neq \overset{o}{q}\overset{o}{p}$
2. **Associative**: $(\overset{o}{p}\overset{o}{q})\overset{o}{r} = \overset{o}{p}(\overset{o}{q}\overset{o}{r})$
3. **Conjugate**: $(p, \mathbf{p})^* = (p, -\mathbf{p}) \implies (\overset{o}{p}\overset{o}{q}) = \overset{o}{q}^*\overset{o}{p}$
4. **Dot Product**: $(p, \mathbf{p}) \cdot (q, \mathbf{q}) = pq + \mathbf{p} + \mathbf{q}$
5. **Norm**: $||\overset{o}{q}||_2^2 = \overset{o}{q} \cdot \overset{o}{q}$
6. **Conjugate Multiplication**: $\overset{o}{q}\overset{o}{q}^*$:

$$\begin{aligned}\overset{o}{q}\overset{o}{q}^* &= (q, \mathbf{q})(q, -\mathbf{q}) \\ &= (q^2 + \mathbf{q} \cdot \mathbf{q}, 0) \\ &= (\overset{o}{q} \cdot \overset{o}{q})\overset{o}{e}\end{aligned}$$

   Where $\overset{o}{e} \triangleq (1, 0)$, i.e. it is a quaternion with no vector component. Conversely, then, we have: $\overset{o}{q}^*\overset{o}{q} = (\overset{o}{q}\overset{o}{q})\overset{o}{e}$.

7. **Multiplicative Inverse**: $\overset{o}{q}^{-1} = \frac{\overset{o}{q}^*}{(\overset{o}{q} \cdot \overset{o}{q})}$ (Except for $\overset{o}{q} = (0, \mathbf{0})$, which is problematic with other representations anyway.)

We can also look at properties with dot products:

1. $(\overset{o}{p}\overset{o}{q}) \cdot (\overset{o}{p}\overset{o}{q}) = (\overset{o}{p} \cdot \overset{o}{p})(\overset{o}{q} \cdot \overset{o}{q})$
2. $(\overset{o}{p}\overset{o}{q}) \cdot (\overset{o}{p}\overset{o}{r}) = (\overset{o}{p} \cdot \overset{o}{p})(\overset{o}{q} \cdot \overset{o}{r})$
3. $(\overset{o}{p}\overset{o}{q}) \cdot \overset{o}{r} = \overset{o}{p} \cdot (\overset{o}{r}\overset{o}{q}^*)$

We can also represent these quaternions as vectors. Note that these quaternions all have zero scalar component.

1. $\overset{o}{r} = (0, \mathbf{r})$
2. $\overset{o}{r}^* = (0, -\mathbf{r})$
3. $\overset{o}{r} \cdot \overset{o}{s} = \mathbf{r} \cdot \mathbf{s}$
4. $\overset{o}{r}\overset{o}{s} = (-\mathbf{r} \cdot \mathbf{s}, \mathbf{r} \times \mathbf{s})$
5. $(\overset{o}{r}\overset{o}{s}) \cdot \overset{o}{t} = \overset{o}{r} \cdot (\overset{o}{s}\overset{o}{t}) = [\mathbf{r}\ \mathbf{s}\ \mathbf{t}]$ (Triple Products)
6. $\overset{o}{r}\overset{o}{r} = -(\mathbf{r} \cdot \mathbf{r})\overset{o}{e}$

Another **note**: For representing rotations, we will use unit quaternions. We can represent scalars and vectors with:

- **Representing scalars**: $(s, \mathbf{0})$
- **Representing vectors**: $(0, \mathbf{v})$

## 4.5 Quaternion Rotation Operator

To represent a rotation operator using quaternions, we need a quaternion operation that maps from vectors to vectors. More specifically, we need an operation that maps from 4D, the operation in which quaternions reside, to 3D in order to ensure that we are in the correct for rotation in 3-space. Therefore, our rotation operator is given:

$$\begin{aligned}\overset{o}{r}' = R(\overset{o}{r}) &= \overset{o}{q}\overset{o}{r}\overset{o}{q}^* \\ &= (Q\overset{o}{r})\overset{o}{q}^* \\ &= (\bar{Q}^T Q)\overset{o}{r}\end{aligned}$$

Where the matrix $\bar{Q}^T Q$ is given by:

$$\bar{Q}^T Q = \begin{bmatrix} \overset{o}{q} \cdot \overset{o}{q} & 0 & 0 & 0 \\ 0 & q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 0 & 2(q_y q_x + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 0 & 2(q_z q_x - q_0 q_y) & 2(q_z q_y + q_0 q_z) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}$$

A few notes about this matrix:

- Since the scalar component of $\overset{o}{q}$ is zero, the first row and matrix of this column are sparse, as we can see above.
- If $\overset{o}{q}$ is a unit quaternion, the lower right $3 \times 3$ matrix of $\bar{Q}^T Q$ will be orthonormal (it is an orthonormal rotation matrix).

Let us look at more properties of this mapping $\overset{o}{r}' = \overset{o}{q}\overset{o}{r}\overset{o}{q}^*$:

1. **Scalar Component**: $r' = r(\overset{o}{q} \cdot \overset{o}{q})$
2. **Vector Component**: $\mathbf{r}' = (q^2 - \mathbf{q} \cdot \mathbf{q})\mathbf{r} + 2(\mathbf{q} \cdot \mathbf{r})\mathbf{q} + 2q(\mathbf{q} \times \mathbf{r})$
3. **Operator Preserves Dot Products**: $\overset{o}{r}' \cdot \overset{o}{s}' = \overset{o}{r} \cdot \overset{o}{s} \implies \mathbf{r}' \cdot \mathbf{s}' = \mathbf{r} \cdot \mathbf{s}$
4. **Operator Preserves Triple Products**: $(\overset{o}{r}' \cdot \overset{o}{s}') \cdot \overset{o}{t}' = (\overset{o}{r} \cdot \overset{o}{s}) \cdot \overset{o}{t} \implies (\mathbf{r}' \cdot \mathbf{s}')\mathbf{t}' = (\mathbf{r} \cdot \mathbf{s}) \cdot \mathbf{t} \implies [\mathbf{r}'\ \mathbf{s}'\ \mathbf{t}'] = [\mathbf{r}\ \mathbf{s}\ \mathbf{t}]$
5. **Composition (of rotations!)**: Recall before that we could not easily compose rotations with our other rotation representations. Because of associativity, however, we can compose rotations simply through quaternion multiplication:

   $$\overset{o}{p}(\overset{o}{q}\overset{o}{r}\overset{o}{q}^*)\overset{o}{p}^* = (\overset{o}{p}\overset{o}{q})\overset{o}{r}(\overset{o}{q}^*\overset{o}{p}^*) = (\overset{o}{p}\overset{o}{q})\overset{o}{r}(\overset{o}{p}\overset{o}{q})^*$$

   I.e. if we denote the product of quaternions $\overset{o}{z} \overset{\Delta}{=} \overset{o}{p}\overset{o}{q}$, then we can write this rotation operator as a single rotation:

   $$\overset{o}{p}(\overset{o}{q}\overset{o}{r}\overset{o}{q}^*)\overset{o}{p}^* = (\overset{o}{p}\overset{o}{q})\overset{o}{r}(\overset{o}{q}^*\overset{o}{p}^*) = (\overset{o}{p}\overset{o}{q})\overset{o}{r}(\overset{o}{p}\overset{o}{q})^* = \overset{o}{z}\overset{o}{r}\overset{o}{z}^*$$

   This ability to compose rotations is quite advantageous relative to many of the other representations of rotations we have seen before (orthonormal rotation matrices can achieve this as well).

# 5 3D Recognition

## 5.1 Extended Gaussian Image

The idea of the extended Gaussian Image: what do points on an obect and points on a sphere have in common? They have the same surface normals.

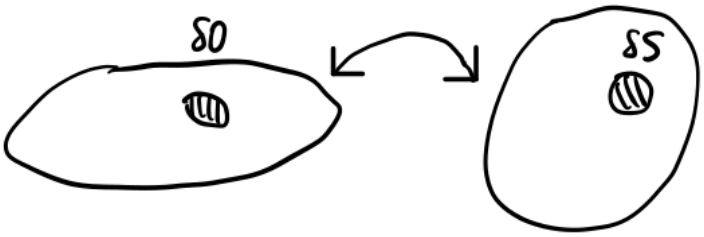


Figure 10: Mapping from object space the Gaussian sphere using correspondences between surface normal vectors. Recall that this method can be used for tasks such as image recognition and image alignment.

- **Curvature**: $\kappa = \frac{\delta S}{\delta O} = \lim_{\delta \to 0} \frac{\delta S}{\delta O} = \frac{dS}{dO}$
- **Density**: $G(\eta) = \frac{\delta O}{\delta S} = \lim_{\delta \to 0} \frac{\delta O}{\delta S} = \frac{dO}{dS}$

## 5.2 EGI with Solids of Revolution

Are there geometric shapes that lend themselves well for an "intermediate representation" with EGI (not too simple, nor too complex)? It turns out there are, and these are the **solids of revolution**. These include:

- Cylinders
- Spheres
- Cones
- Hyperboloids of one and two sheets

How do we compute the EGI of solids of revolution? We can use **generators** that produce these objects to help.

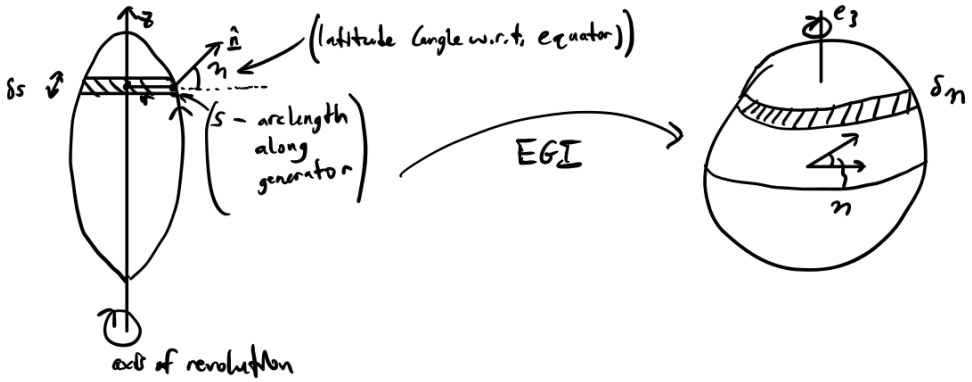


Figure 11: EGI representation of a generalized solid of revolution. Note that bands in the object domain correspond to bands in the sphere domain.

As we can see from the figure, the bands "map" into each other! These solids of revolution are symmetric in both the object and transform space. Let's look at constructing infinitisimal areas so we can then compute Gaussian curvature $\kappa$ and density $G$:

- **Area of object band**: $\delta O = 2\pi r \delta s$
- **Area of sphere band**: $\delta S = 2\pi \cos(\eta) \delta n$

Then we can compute the curvature as:

$$\kappa = \frac{\delta S}{\delta O} = \frac{2\pi \cos(\eta)\delta\eta}{2\pi r \delta s} = \frac{\cos(\eta)\delta\eta}{r\delta s}$$
$$G = \frac{1}{\kappa} = \frac{\delta O}{\delta S} = \sec(\eta)\frac{\delta s}{\delta \eta}$$

Then in the limit of $\delta \to 0$, our curvature and density become:

$\kappa = \lim_{\delta \to O} \frac{\delta S}{\delta O} = \frac{\cos\eta}{r}\frac{d\eta}{ds}$ (Where $\frac{d\eta}{ds}$ is the rate of change of surface normal direction along the arc, i.e. curvature)
$G = \lim_{\delta \to 0} \frac{\delta O}{\delta S} = r\sec\eta\frac{ds}{d\eta}$ (Where $\frac{ds}{d\eta}$ is the rate of change of the arc length w.r.t. angle)
$\kappa = \frac{\cos\eta}{r}\kappa$

Recall that we covered this for the following

### 5.3 Sampling From Spheres Using Regular and Semi-Regular Polyhedra

More efficient to sample rotations from shapes that form a "tighter fit" around the sphere - for instance: **polyhedra**! Some polyhedra we can use:

- **Tetrahedra** (4 faces)
- **Hexahedra** (6 faces)
- **Octahedra** (8 faces)
- **Dodecahedra** (12 faces)
- **Icosahedra** (20 faces)

These polyhedra are also known as the **regular solids**.

As we did for the cube, we can do the same for polyhedra: to sample from the sphere, we can sample from the polyhedra, and then **project** onto the point on the sphere that intersects the line from the origin to the sampled point on the polyhedra. From this, we get **great circles** from the edges of these polyhedra on the sphere when we project.

**Fun fact**: Soccer balls have 32 faces! More related to geometry: soccer balls are part of a group of **semi-regular** solids, specifically an **icosadodecahedron**.

### 5.4 Desired Properties of Dividing Up the Sphere/Tessellations

To build an optimal representation, we need to understand what our desired optimal properties of our tessellations will be:

1. **Equal areas of cells/facets**
2. **Equal shapes of cells/facets**
3. **"Rounded" shapes of cells/facets**
4. **Regular pattern**
5. **Allows for easy "binning"**
6. **Alignment of rotation**

**Platonic** and **Archimedean** solids are the representations we will use for these direction histograms.

## 6 References

1. Groups, https://en.wikipedia.org/wiki/Group_(mathematics)