6.801/6.866: Machine Vision, Lecture 20

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.

1 Lecture 20: Space of Rotations, Regular Tessellations, Critical Surfaces in Motion Vision and Binocular Stereo

In this lecture, we will transition from solving problems of **absolute rotation** (which, if you recall, finds the transformation between two 3D coordinate systems) into **relative orientation**, which finds the transformation between two 2D coordinate systems. We will start by covering the problem of **binocular stereo**, and in the process talk about tessellations from solids, critical surfaces.

1.1 Tessellations of Regular Solids

As a brief review from the last lecture, recall that we saw we can encode rotations as 4D points on the unit sphere, where the coordinates of these 4D points correspond to our coefficients for the unit quaternion.

What are tessellations? "A filling or tessellations of a flat surface is the covering of a plane using one or more geometric shapes (polygons)." [1]. **Tessellations** of the surface of the sphere can be based on **platonic solids**, with 4, 6, 8, 12, and 20 faces. Each of the tessellations from the platonic solids results in equal area projections on the sphere, but the division is somewhat coarse.

For **greater granularity**, we can look at the 14 **Archimedean solids**. This allows for having multiple polygons in each polyhedra (e.g. squares and triangles), resulting in unequal area in the tessellations on the unit sphere.

Related, we are also interested in the **rotation groups** (recall **groups** are mathematical sets that obey certain algebras, e.g. the **Special Orthogonal** group) of these regular polyhedra:

- 12 elements in rotation group for **tetrahedron**.
- 24 elements in rotation group for **hexahedron**.
- 60 elements in rotation group for **dodecahedron**.
- The **octahedron** is the dual of the **cube** and therefore occupies the same rotation group as it.
- The **icosahedron** is the dual of the **dodecahedron** and therefore occupies the same rotation group as it.

A few other notes on tessellations:

- One frequently-used method for creating tessellations is to divide each face into many **triangular** or **hexagonal** areas.
- Histograms can be created in tessellations in planes by taking square sub-divisions of the region.
- **Hexagonal tessellations** are a dual of **triangular tessellations**.

1.2 Critical Surfaces

What are Critical Surfaces? Critical surfaces are geometric surfaces - specifically, hyperboloids of one sheet, that lead to ambiguity in the solution space for relative orientation problems.

Why are Critical Surfaces important? Critical surfaces can negatively impact the performance of relative orientation systems, and understanding their geometry can enable us to avoid using strategies that rely on these types of surfaces in order to find the 2D transformation, for instance, between two cameras.

We will discuss more about these at the end of today's lecture. For now, let's introduce **quadrics** - geometric shapes/surfaces defined by second-order equations in a 3D Cartesian coordinate system:

1. **Ellipsoid**: $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$

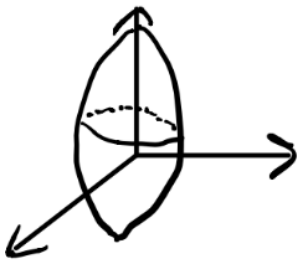

Figure 1: 3D geometric depiction of an ellipsoid, one member of the quadric family.

2. **Sphere**: $x^2 + y^2 + z^2 = 1$ (or more generally, $= r \in \mathbb{R}$)

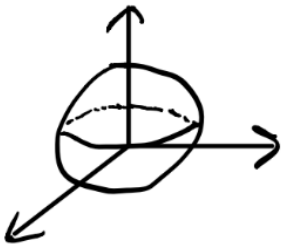

Figure 2: 3D geometric depiction of a sphere, another member of the quadric family and a special case of the ellipse.

3. **Hyperboloid of One Sheet**: $x^2 + y^2 - z^2 = 1$

   This quadric surface is **ruled**: we can embed straight lines in the surface, despite its quadric function structure.

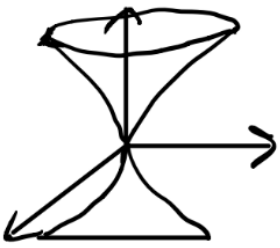

Figure 3: 3D geometric depiction of a hyperboloid of one sheet, another member of the quadric family and a special case of the ellipse.

4. **Hyperboloid of Two Sheets**: $x^2 - y^2 - z^2 = 1$

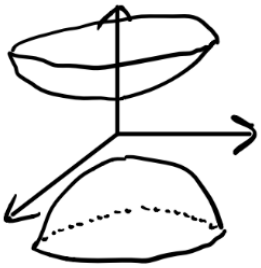

Figure 4: 3D geometric depiction of a hyperboloid of one sheet, another member of the quadric family and a special case of the ellipse.

5. **Cone**: $\frac{z^2}{c^2} = \frac{x^2}{a^2} + \frac{y^2}{b^2}$

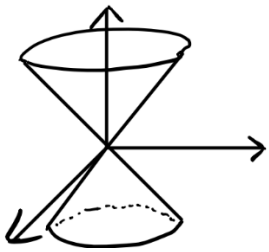

Figure 5: 3D geometric depiction of a cone, another member of the quadric family and a special case of the hyperboloid of one sheet.

6. **Elliptic Paraboloid**: $\frac{z}{c} = \frac{x^2}{a^2} + \frac{y^2}{b^2}$

   Note that this quadric surface has a linear, rather than quadratic dependence, on z.

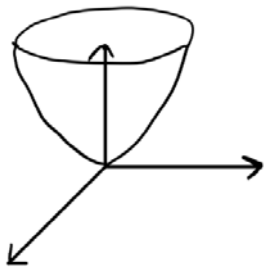

Figure 6: 3D geometric depiction of a elliptic paraboloid, another member of the quadric family and a special case of the hyperboloid of one sheet.

7. **Hyperbolic Paraboloid**: $\frac{z}{c} = \frac{x^2}{a^2} - \frac{y^2}{b^2}$

   Note that this quadric surface also has a linear, rather than quadratic dependence, on z.

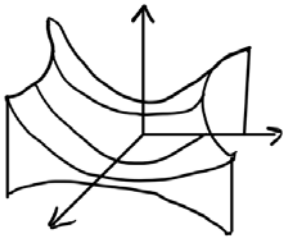

Figure 7: 3D geometric depiction of a hyperbolic paraboloid, another member of the quadric family and a special case of the hyperboloid of one sheet.

Another special case is derived through **planes**. You may be wondering - how can we have a quadratic structure from planes? We can derive a surface with quadratic terms by considering the intersection of two planes. This intersection of planes is computed analytically as a product of two linear equations, resulting in a quadratic equation.

1.3 Relative Orientation and Binocular Stereo

Problem of interest: Computing 3D from 3D using two cameras - a problem known as **binocular stereo**. We found this problem was easy to solve if the geometry of the two cameras, in this case known as the **two-view geometry**, is known and calibrated (usually this results in finding a calibration matrix K). To achieve high-performing binocular stereo systems, we need to find the relative orientation between the two cameras. A few other notes on this:

- Calibration is typically for **binocular stereo** using **baseline calibration**.
- Recall that like absolute orientation, we have **duality** in te problems we solve: we are able to apply the same machine vision framework regardless of "if the camera moved of it the world moved".
- Therefore, the dual problem to **binocular stereo** is **structure from motion**, also known as **Visual Odometry (VO)**. This involves finding transformations over time from camera (i.e. one moving camera at different points in time).

1.3.1 Binocular Stereo

Recall that our goal with binocular stereo is to find the **translation** (known as the **baseline**, given by $\mathbf{b} \in \mathbb{R}^3$. The diagram below illustrates our setup with two cameras, points in the world, and the translation between the two cameras given as this aforementioned baseline **b**.

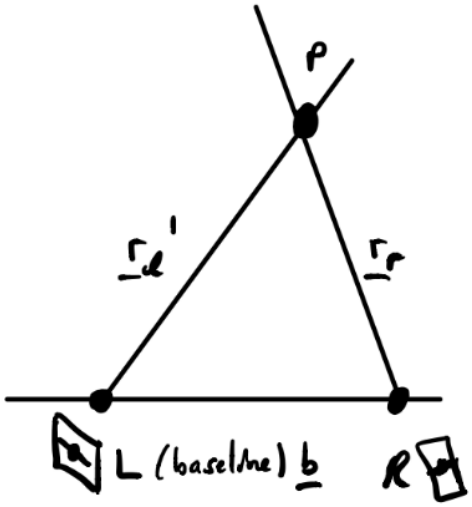

Figure 8: Binocular stereo system set up. For this problem, recall that one of our objectives is to measure the translation, or baseline, between the two cameras.

When we search for correspondences, we need only search along a line, which gives us a measure of **binocular disparity** that we can then use to measure distance between the cameras.

The lines defined by these correspondences are called **epipolar lines**. Places that pass through **epipolar lines** are called **epipolar planes**, as given below:

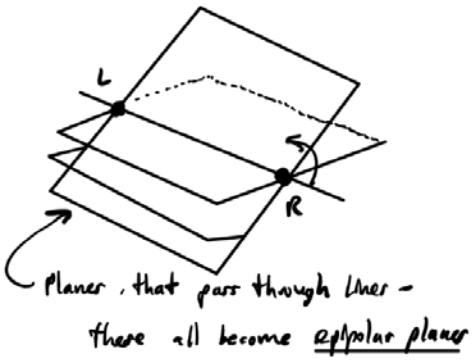

Figure 9: Epipolar planes are planes that pass through epipolar lines.

Next, in the image, we intersect the image plane with our set of epipolar planes, and we look at the intersections of these image planes (which become lines). The figure below illustrates this process for the left and right cameras in our binocular stereo system.

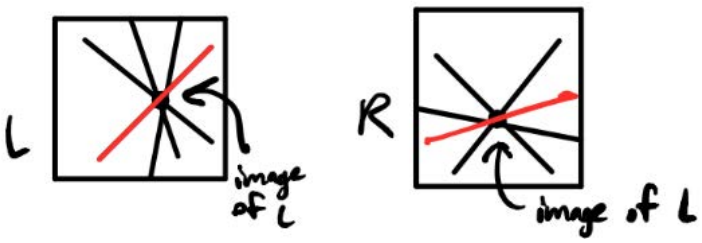

Figure 10: After finding the epipolar planes, we intersect these planes with the image plane, which gives us a set of lines in both the left and righthand cameras/coordinate systems of our binocular stereo system.

A few notes on this process:

- For system convergence, we want as much overlap as possible between the two sets of lines above, depicted in Figure 10.
- The entire baseline **b** projects to a single point in each image.
- As we stated before, there are line correspondences if we already know the geometry, i.e. if we want correspondences between the lines in red above, we need only look along the red lines in each image.
- Recall that our goal is to find the **transformation** between two cameras - i.e. **baseline** (or **translation**, in dual structure from motion problem).
- Recall, as we saw for absolute orientation, that we also solve for **rotation** in addition to the **baseline/translation**. Together, these translation + rotation variables lead us to think that we are solving a problem of 6 degrees of freedom (DOF), but because of **scale factor ambiguity** (we cannot get absolute sizes of objects/scenes we image), we cannot get absolute length of the baseline **b**, and therefore, we treat the baseline as a unit vector. Treating the baseline as a unit vector results in one less DOF and 5 DOF for the system overall (since we now only have 2 DOF for the unit vector baseline).

1.3.2 How Many Correspondences Do We Need?

As we have done so for other systems/problems, we consider the pertinent question: **How many correspondences do we need to solve the binocular stereo/relative orientation problem?**. To answer this, let us consider what happens when we are given different numbers of correspondences:

- With 1 correspondence, we have many degrees of freedom left - i.e. not enough constraints for the number of unknowns remaining. This makes sense considering that each correspondence imposes a maximum of 3 DOF for the system.

- With 2 correspondences, we still have the ability to rotate one of the cameras without changing the correspondence, which implies that only 4 out of the 5 constraints are satisfied. This suggests we need more constraints.

It turns out we need **5 correspondences** needed to solve the binocular stereo/relative orientation problem - each correspondence gives you **1 constraint**. Why? Each image has a **disparity**, with two components (this occurs in practice). There are two types of disparity, each corresponding to each of the 2D dimensions. Note that **disparity** computes pixel discrepancies between correspondences between images.

- **Horizontal disparity** corresponds to depth.
- **Vertical disparity** corresponds to differences in orientation, and actually takes the constraints into account. In practice, vertical disparity is tuned out first using an iterative sequence of 5 moves.

In practice, we would like to use **more correspondences for accuracy**. With more correspondences, we no longer try to find exact matches, but instead minimize a sum of squared errors using least squares methods. We minimize sum of squares of error in **image position**, not in the 3D world.

However, one issue with this method is **nonlinearity**:

- This problem setup results in 7 second-order equations, which, by Bezout's Theorem, gives $2^7 = 128$ different solutions to this problem.
- Are there really 128 solutions? There are methods that have gotten this down to 20. With more correspondences and more background knowledge of your system, it becomes easier to determine what the true solution is.

1.3.3 Determining Baseline and Rotation From Correspondences

To determine baseline and rotation for this binocular stereo problem, we can begin with the figure below:

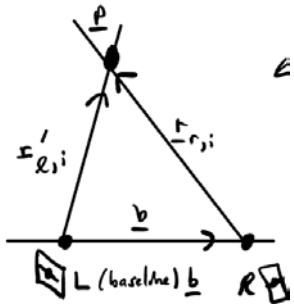

Figure 11: Epipolar plane for the binocular stereo system.

The vectors $\mathbf{r}'_{l,i}$ (the left system measurement after the rotation transformation has been applied), $\mathbf{r}_{r,i}$, and $\mathbf{b}$ are all coplanar in a perfect system. Therefore, if we consider the triple product of these 3 vectors, the volume of the parallelipiped, which can be constructed from the **triple product**, should have **zero volume** because these vectors are coplanar (note that this is the ideal case). This is known as the **coplanarity condtion**:

$$V = [\mathbf{r}'_{\mathrm{l,i}}\ \mathbf{r}_{\mathrm{r,i}}\ \mathbf{b}] = 0$$

A potential solution to find the optimal baseline (and rotation): we can use least squares to minimize the volume of the parallelipiped corresponding to the triple product of these three (hopefully near coplanar) vectors. This is a feasible approach, but also have a **high noise gain/variance**.

This leads us to an important question: What, specifically, are we trying to minimize? Recall that we are **matching correspondences in the image, not in the 3D world.** Therefore, the volume of the parallelipiped is **proportional to the error**, but does not match it exactly. When we have measurement noise, the rays from our vectors in the left and righthand systems no longer intersect, as depicted in the diagram below:

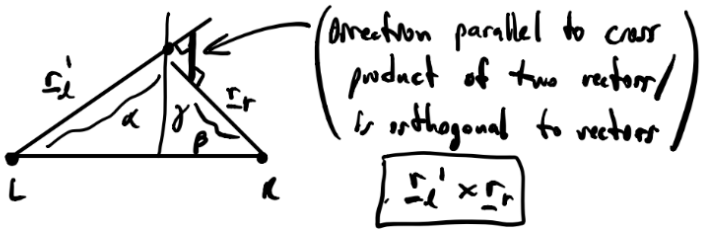

Figure 12: With measurement noise, our rays do not line up exactly. We will use this idea to formulate our optimization problem to solve for optimal baseline and rotation.

We can write that the error is perpendicular to the cross product between the rotated left and the right rays:

$$\mathbf{r}'_{l,i} \times \mathbf{r}_r$$

And therefore, we can write the equation for the "loop" (going from the rotated left coordinate system to the right coordinate system) as:

$$\alpha\mathbf{r}'_l + \gamma(\mathbf{r}'_l \times \mathbf{r}_r) = \mathbf{b} + \beta\mathbf{r}_r$$

Where the error we seek to minimize $(\mathbf{r}'_l \times \mathbf{r}_r)$ is multiplied by a parameter γ.

To solve for our parameters α, β, and γ, we can transform this vector equation into 3 scalar equations by taking dot products. We want to take dot products that many many terms drop to zero, i.e. where orthogonality applies. Let's look at these 3 dot products:

1. $\cdot(\mathbf{r}'_l \times \mathbf{r}_r)$: This yields the following:

$$\begin{aligned}
&\textbf{Lefthand side}: \quad (\alpha\mathbf{r}'_l + \gamma(\mathbf{r}'_l \times \mathbf{r}_r)) \cdot (\mathbf{r}'_l \times \mathbf{r}_r) = \gamma||\mathbf{r}'_l \times \mathbf{r}_r||_2^2 \\
&\textbf{Righthand side}: \quad (\mathbf{b} + \beta\mathbf{r}_r) \cdot (\mathbf{r}'_l \times \mathbf{r}_r) = \mathbf{b} \cdot \mathbf{r}'_l \times \mathbf{r}_r + 0 = [\mathbf{b}\ \mathbf{r}'_l\ \mathbf{r}_r] \\
&\textbf{Combining}: \quad \gamma||\mathbf{r}'_l \times \mathbf{r}_r||_2^2 = [\mathbf{b}\ \mathbf{r}'_l\ \mathbf{r}_r]
\end{aligned}$$

   Intuitively, this says that the error we see is proportional to teh triple product (the volume of the parallelipiped). Taking our equation with this dot product allows us to isolate γ.

2. $\cdot(\mathbf{r}'_l \times (\mathbf{r}'_l \times \mathbf{r}_r))$: This yields the following:

$$\begin{aligned}
&\textbf{Lefthand side}: \quad (\alpha\mathbf{r}'_l + \gamma(\mathbf{r}'_l \times \mathbf{r}_r)) \cdot ((\mathbf{r}'_l \times (\mathbf{r}'_l \times \mathbf{r}_r)) = \beta||\mathbf{r}'_l \times \mathbf{r}_r||_2^2 \\
&\textbf{Righthand side}: \quad (\mathbf{b} + \beta\mathbf{r}_r) \cdot ((\mathbf{r}'_l \times (\mathbf{r}'_l \times \mathbf{r}_r)) = (\mathbf{b} \times \mathbf{r}'_l) \cdot (\mathbf{r}'_l \times \mathbf{r}_r) \\
&\textbf{Combining}: \quad \beta||\mathbf{r}'_l \times \mathbf{r}_r||_2^2 = (\mathbf{b} \times \mathbf{r}'_l) \cdot (\mathbf{r}'_l \times \mathbf{r}_r)
\end{aligned}$$

   Taking this dot product with our equation allows us to find β. We can repeat an analogous process to solve for α.

3. $\cdot(\mathbf{r}_r \times (\mathbf{r}'_l \times \mathbf{r}_r))$: This yields the following:

$$\begin{aligned}
&\textbf{Lefthand side}: \quad (\alpha\mathbf{r}'_l + \gamma(\mathbf{r}'_l \times \mathbf{r}_r)) \cdot ((\mathbf{r}_r \times (\mathbf{r}'_l \times \mathbf{r}_r)) = \alpha||\mathbf{r}'_l \times \mathbf{r}_r||_2^2 \\
&\textbf{Righthand side}: \quad (\mathbf{b} + \beta\mathbf{r}_r) \cdot ((\mathbf{r}_r \times (\mathbf{r}'_l \times \mathbf{r}_r)) = (\mathbf{b} \times \mathbf{r}_r) \cdot (\mathbf{r}'_l \times \mathbf{r}_r) \\
&\textbf{Combining}: \quad \alpha||\mathbf{r}'_l \times \mathbf{r}_r||_2^2 = (\mathbf{b} \times \mathbf{r}_r) \cdot (\mathbf{r}'_l \times \mathbf{r}_r)
\end{aligned}$$

   Taking this dot product with our equation allows us to find α.

With this, we have now isolated all three of our desired parameters. We can then take these 3 equations to solve for our 3 unknowns α, β, and γ. We want $|\gamma|$ to be as small as possible. We also require α and β to be non-negative, since these indicate the scalar multiple of the direction along the rays in which we (almost) get an intersection between the left and right coordinate systems. Typically, a negative α and/or β results in intersections behind the camera, which is often not physically feasible.

It turns out that one of the ways discard some of the 20 solutions produced by this problem is to throw out solutions that result in negative α and/or β.

Next, we can consider the distance this vector corresponding to the offset represents. This distance is given by:

$$d = \gamma||\mathbf{r}'_l \times \mathbf{r}_r||_2 = \frac{[\mathbf{b}\ \mathbf{r}'_l\ \mathbf{r}_r]}{||\mathbf{r}'_l \times \mathbf{r}_r||_2}$$

Closed-Form Solution: Because this is a system involving 5 second-order equations, and the best we can do is reduce this to a single 5th-order equation, which we do not (yet) have the closed-form solutions to, we cannot solve for this problem in closed-form. However, we can still solve for it numerically. We can also look at solving this through a weighted least-squares approach below.

1.3.4 Solving Using Weighted Least Squares

Because our distance d can get very large without a huge error in image position, d is not representative of our error of interest. However, it is still proportional. We can account for this difference by using a weighting factor w_i, which represents the conversion factor from 3D error to 2D image error). We will denote the i$^{\text{th}}$ weight as w_i.

Therefore, incorporating this weighting factor, our least squares optimization problem becomes:

$$\min_{\mathbf{b},R(\cdot)} \sum_{i=1}^{n} w_i [\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]^2, \text{ subject to } \mathbf{b}\cdot\mathbf{b} = ||\mathbf{b}||_2^2 = 1$$

How do we solve this? Because w_i will change as our candidate solution changes, we will solve this problem iteratively and in an alternating fashion - we will alternate between updated our conversion weights w_i and solving for a new objective given the recent update of weights. Therefore, we can write this optimization problem as:

$$\begin{aligned}\mathbf{b}^*, R^* &= \arg \min_{\mathbf{b},||\mathbf{b}||_2^2=1,R(\cdot)} \sum_{i=1}^{n} w_i [\mathbf{b}\ \mathbf{r}_l'\ \mathbf{r}_r]^2 \\ &= \arg \min_{\mathbf{b},||\mathbf{b}||_2^2=1,R(\cdot)} \sum_{i=1}^{n} w_i \left((\mathbf{r}_{r,i} \times \mathbf{b}) \cdot \mathbf{r}_{l,i}'\right)^2\end{aligned}$$

Intuition for these weights: Loosely, we can think of the weights as being the conversion factor from 3D to 2D error, and therefore, they can roughly be thought of as $w = \frac{f}{Z}$, where f is the focal length and Z is the 3D depth.

Now that we have expressed a closed-form solution to this problem, we are ready to build off of the last two lectures and apply our unit quaternions to solve this problem. Note that because we express the points from the left coordinate system in a rotated frame of reference (i.e. with the rotation already applied), then we can incorporate the quaternion into this definition to show how we can solve for our optimal set of parameters given measurements from both systems:

$$\overset{o}{r}_l' = \overset{o}{q}\overset{o}{r}_l\overset{o}{q}^*$$

Then we can solve for this as t:

$$\begin{aligned}t &= (\mathbf{r}_r \times \mathbf{b}) \cdot \mathbf{r}_l' \\ &= \overset{o}{r}_r \overset{o}{b} \cdot \overset{o}{q}\overset{o}{r}_l\overset{o}{q}^* \\ &= \overset{o}{r}_r (\overset{o}{b}\overset{o}{q}) \cdot \overset{o}{q}\overset{o}{r}_l \\ &= \overset{o}{r}_d \overset{o}{d} \cdot \overset{o}{q}\overset{o}{r}_l, \text{ where } \overset{o}{d} \triangleq \overset{o}{b}\overset{o}{q}, \text{ which we can think of as a product of baseline and rotation.}\end{aligned}$$

Recall that our goal here is to find the **baseline** $\overset{o}{b}$ - this can be found by solving for our quantity $\overset{o}{d}$ and multiplying it on the righthand side by $\overset{o}{q}^*$:

$$\begin{aligned}&\overset{o}{d}\overset{o}{q}^* = \overset{o}{b}\overset{o}{q}\overset{o}{q}^* = \overset{o}{b}\overset{o}{e} = \overset{o}{b} \\ &\text{(Recall that } \overset{o}{e} = (1,0) = \overset{o}{q}\overset{o}{q}^*\text{)}\end{aligned}$$

At first glance, it appears we have 8 unknowns, with 5 constraints. But we can add additional constraints to the system to make the number of constraints equal the number of DOF:

1. **Unit quaternions**: $||\overset{o}{q}||_2 = \overset{o}{q} \cdot \overset{o}{q} = 1$
2. **Unit baseline**: $||\overset{o}{b}||_2 = \overset{o}{b} \cdot \overset{o}{b} = 1$
3. **Orthogonality of $\overset{o}{q}$ and $\overset{o}{d}$**: $\overset{o}{q} \cdot \overset{o}{d} = 0$

Therefore, with these constraints, we are able to reach a system with 8 constraints. Note that we have more constraints to enforce than with the absolute orientation problem, making the relative orientation problem more difficult to solve.

1.3.5 Symmetries of Relative Orientation Approaches

We can interchange the left and right coordinate system rays. We can do this for this problem because we have **line intersections** rather than **line rays**. These symmetries can be useful for our numerical approaches. The equation below further demonstrates this symmetry by showing we can interchange the order of how $\overset{o}{d}$ and $\overset{o}{q}$ interact with our measurements to produce the same result.

$$\begin{aligned} t &= \overset{o}{r}_r \overset{o}{d} \cdot \overset{oo}{q} r_l \\ &= \overset{o}{r}_r \overset{o}{q} \cdot \overset{o}{d} r_l \end{aligned}$$

Given these symmetries, we can come up with some sample solutions:

1. $\{\overset{o}{q}, \overset{o}{d}\}$
2. $\{-\overset{o}{q}, \overset{o}{d}\}$ $(\times 2)$
3. $\{\overset{o}{q}, -\overset{o}{d}\}$ $(\times 2)$
4. $\{\overset{o}{d}, \overset{o}{q}\}$ $(\times 2)$

How do we avoid having to choose from all these solutions? One approach: Assume/fix one of the two unknowns $\overset{o}{d}$ or $\overset{o}{q}$. This results in a linear objective and a linear problem to solve, which, as we know, can be solved with **least squares** approaches, giving us a closed-form solution:

- **Option 1**: Assume $\overset{o}{q}$ is known and fix it $\longrightarrow$ solve for $\overset{o}{d}$
- **Option 2**: Assume $\overset{o}{d}$ is known and fix it $\longrightarrow$ solve for $\overset{o}{q}$

Note that both of the approaches above can be solved with least squares!

While this is one approach, a better approach is to use **nonlinear optimization**, such as **Levenberg-Marquadt** (often called LM in nonlinear optimization packages). An optional brief intro to Levenberg-Marquadt is presented at the end of this lecture summary.

Levenberg-Marquadt (LM) optimization requires a non-redundant parameterization, which can be achieved with either **Gibbs vectors** or **quaternions** (the latter of which we can circumvent the redundancies of Hamilton's quaternions by treating the redundancies of this representation as extra constraints. Recall that Gibbs vectors, which are given as $(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\hat{\boldsymbol{\omega}})$, have a singularity at $\theta = \pi$.

1.3.6 When Does This Fail? Critical Surfaces

Like all the other machine vision approaches we have studied so far, we would like to understand when and why this system fails:

- Recall that we need at least 5 correspondences to solve for the **baseline** and **rotation**. With less correspondences, we don't have enough constraints to find solutions.
- **Gefärliche Flaächen**, also known as **"dangerous or critical surfaces"**: There exist surfaces that make the relative orientation problem difficult to solve due to the additional ambiguity that these surfaces exhibit. One example: **Plane flying over a valley with landmarks:**

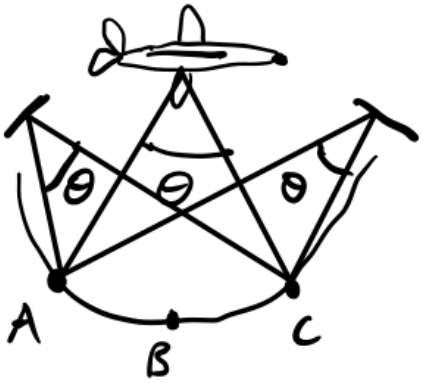

Figure 13: A plane flying over a valley is an instance of a critical surface. This is because we can only observe angles, and not points. Therefore, locations of two landmarks are indistinguishable.

To account for this, pilots typically plan flight paths over ridges rather than valleys for this exact reason:

Figure 14: When planes fly over a ridge rather than a valley, the angles between two landmarks do change, allowing for less ambiguity for solving relative orientation problems.

How do we generalize this case to 3D? We will see that this type of surface in 3D is a **hyperboloid of one sheet**. We need to ensure sections of surfaces you are looking at do not closely resemble sections of hyperboloids of one sheet.

There are also other types of **critical surfaces** that are far more common that we need to be mindful of when considering relative orientation problems - for instance, the intersection of two planes. In 2D, this intersection of two planes can be formed by the product of their two equations:

$$(ax + by + c)(dx + ey + f) = adx^2 + aexy + afx + bdxy + bey^2 + bfy + cdx + cey + cf = 0$$

We can see that this equation is indeed second order with respect to its spatial coordinates, and therefore belongs to the family of quadric surfaces and critical surfaces.

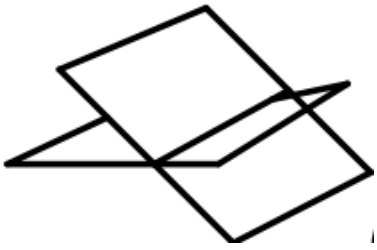

Figure 15: The intersection of two planes is another type of critical surface we need to be mindful of. It takes a second-order form because we multiply the two equations of the planes together to obtain the intersection.

1.3.7 (Optional) Levenberg-Marquadt and Nonlinear Optimization

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:

- θ 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 λI, 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 α is the step size, which dictates how quickly the estimates of our approaches update.

1.4 References

1. Tesselation, https://en.wikipedia.org/wiki/Tessellation

6.801 / 6.866 Machine Vision
Fall 2020