6.801/6.866: Machine Vision, Lecture 5

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**.

1 Lecture 5: TTC and FOR Montivision Demos, Vanishing Point, Use of VPs in Camera Calibration

In this lecture, we'll continue going over **vanishing points** in machine vision, as well as introduce how we can use brightness estimates to obtain estimates of a surface's orientation. We'll introduce this idea with **Lambertian surfaces**, but we can discuss how this can generalize to many other types of surfaces as well.

1.1 Robust Line Estimation and Sampling

We'll start by covering some of the topics discussed during lecture.

1.1.1 Line Intersection Least-Squares

Helpful when considering multiple lines at a time - this optimization problem is given by:

$$\min_{x,y} \sum_i (x \cos\theta_i + y \sin\theta_i - \rho)^2 \tag{1}$$

1.1.2 Dealing with Outliers

Many of the approaches we've covered thus far are centered around the idea of making estimates from data. But recall that the data gathered by sensors can be noisy, and can be corrupted from phenomena such as crosstalk. Because of this noise, it is advantageous to distinguish and filter outliers from the inliers in our dataset, especially when we make estimates using Least-Squares.

A good choice of algorithm for dealing with outliers is **RANSAC**, or **Random Sample Consensus** [1]. This algorithm is essentially an iterative and stochastic variation of Least-Squares. By randomly selecting points from an existing dataset to fit lines and evaluate the fit, we can iteratively find line fits that minimize the least-squares error while distinguishing inliers from outliers.

Algorithm 1 RANSAC

1: Select randomly the minimum number of points required to determine the model parameters.
2: Solve for the parameters of the model.
3: Determine how many points from the set of all points fit with a predefined tolerance ϵ.
4: If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold τ, re-estimate the model parameters using all the identified inliers and terminate.
5: Otherwise, repeat steps 1 through 4 (maximum of N times).

Figure 1: Pseudocode for the RANSAC Algorithm.

1.1.3 Reprojection and Rectification

These two problems are highly important in mapmaking, and help for solving the equations:

$$R\mathbf{r} = \mathbf{r}' \tag{2}$$

$$\mathbf{r} = R^T\mathbf{r} \tag{3}$$

Where $\mathbf{r}$ is the vector from the Center of Projection (COP) to the image plane.

1.1.4 Resampling

Resampling is also a valuable application in many facets of computer vision and robotics, especially if we seek to run any kind of **interpolation** or **subsampling** algorithms. Some approaches for this:

- **Nearest Neighbor**: This is a class of methods in which we interpolate based off of the values of neighboring points. This can be done spatially (e.g. by looking at adjacent pixels) as well as other image properties such as brightness and color. A common algorithm used here is **K-Nearest Neighbors (KNN)**, in which interpolation is done based off of the K-nearest points in the desired space.
- **Bilinear Interpolation**: An extension of linear interpolation used for functions in two-dimensional grids/of two variables (e.g. (x, y) or (i, j))) [2], such as the brightness or motion in images.
- **Bicubic Interpolation**: Similar to bilinear interpolation, bicubic interpolation is an extension of cubic interpolation of functions in two-dimensional grids/of two variables (e.g. (x, y) or (i, j))) [3], such as the brightness or motion in images. Bicubic interpolation tends to perform much better than bilinear interpolation, though at the cost of additional computational resources.

1.2 Magnification with TTC

- For the Montivision demo, note the following for the bars:
  - $A \rightarrow$ x-dimension for motion estimation problem
  - $B \rightarrow$ y-dimension for motion estimation problem
  - $C \rightarrow \frac{1}{\text{Time to Contact}}$
- Recall from the previous lectures that the percent change of size in the world is the percent change of size in the image. We can derive this through **perspective projection**. The equation for this is:

$$\frac{s}{f} = \frac{S}{Z} \tag{4}$$

Where s is the size in the image plane and S is the size in the world. Differentiating with respect to time gives us (using the chain rule):

$$\frac{d}{dt}(sZ = fS) \rightarrow Z\frac{ds}{dt} + s\frac{dZ}{dt} = 0 \tag{5}$$

Rearranging terms:

$$\frac{\frac{dz}{dt}}{Z} = -\frac{\frac{ds}{dt}}{S} \tag{6}$$

Recall that the intuition here is that the **rate of change of size is the same in the image and the world**.

1.2.1 Perspective Projection and Vanishing Points

Let's begin by defining a few terms. Some of these will be a review, but these review terms connect to new terms that we'll introduce shortly:

- **vanishing points**: These are the points in the image plane (or extended out from the image plane) that parallel lines in the world converge to.
- **Center of Projection (COP)**: This is where in the camera/sensor (not in the image plane) where incident projected light rays converge. An analog to the COP for human vision systems is your eyes. NOTE: COP and vanishing points are oftentimes different.

- **Principle Point**: The orthogonal projection of the Center of Projection (COP) onto the image plane.
- **f**: Similar to the focal length we've seen in other perspective projection examples, this f is the perpendicular distance from the COP to the image plane.

Recall that a common problem we can solve with the use of vanishing points is **finding the Center of Projection (COP)**. Solving this problem in 3D has 3 degrees of freedom, so consequently we'll try to solve it using three equations.

Intuitively, this problem of finding the Center of Projection can be thought of as finding the intersection of three spheres, each of which have two vanishing points along their diameters. Note that three spheres can intersect in up to two places - in this case we have defined the physically-feasible solution, and therefore the solution of interest, to be the solution above the image plane (the other solution will be a mirror image of this solution and can be found below the image plane).

Application of this problem: This problem comes up frequently in photogrammetry, in that simply having two locations as your vanishing points isn't enough to uniquely identify your location on a sphere.

1.2.2 Lines in 2D and 3D

Next, let's briefly review how we can parameterize lines in 2D and 3D:

- **2D**: (2 Degrees of Freedom)
  - $y = mx + c$
  - $ax + by + c = 0$
  - $\sin\theta x - \cos\theta y + \rho = 0$
  - If $\hat{\mathbf{n}} \triangleq (-\sin\theta, \cos\theta)^T$, then $\hat{\mathbf{n}} \cdot \mathbf{r} = \rho$.
- **3D**: (3 Degrees of Freedom)
  - $\hat{\mathbf{n}} = \rho$
  - $aX + bY + cZ + d = 0$

1.2.3 Application: Multilateration (MLAT)

This problem comes up when we estimate either our position or the position of other objects based off of Time of Arrival [4]. One specific application of this problem is localizing ourselves using distances/Time of Arrival of wifi access points inside buildings/other locations without access to GPS.

Like the other problems we've looked at, this problem can be solved by finding the intersection of 3 spheres. Let's begin with:

$$||\mathbf{r} - \mathbf{r}_i||_2 = p_i \; \forall \; i \in \{1, \cdots, N\} \tag{7}$$

Next, let's square both sides of this equation and rewrite the left-hand side with dot products:

$$||\mathbf{r} - \mathbf{r}_i||_2^2 = (\mathbf{r} - \mathbf{r}_i)^T(\mathbf{r} - \mathbf{r}_i) = p_i^2 \tag{8}$$

$$= \mathbf{r} \cdot \mathbf{r} - 2\mathbf{r} \cdot \mathbf{r}_i + \mathbf{r}_i \cdot \mathbf{r}_i = p_i^2 \tag{9}$$

Recall from Bezout's Theorem that this means that are 8 possible solutions here, since we have three equations of second-order polynomials. To get rid of the 2nd order terms, we simply subtract the equations:

$$\mathbf{r} \cdot \mathbf{r} - 2\mathbf{r} \cdot \mathbf{r}_i + \mathbf{r}_i \cdot \mathbf{r}_i = \rho_i^2 \tag{10}$$

$$- \quad \mathbf{r} \cdot \mathbf{r} - 2\mathbf{r} \cdot \mathbf{r}_j + \mathbf{r}_j \cdot \mathbf{r}_j = \rho_j^2 \quad \forall\, i, j \in \{1, 2, 3\}, i \neq j \tag{11}$$

Subtracting these pairs of equations yields a linear equation in $\mathbb{R}$:

$$2\mathbf{r}(r_j - r_i) = (\rho_j^2 - \rho_i^2) - (R_j^2 - R_i^2) \tag{12}$$

(Where the scalar $R_j^2 \triangleq \mathbf{r}_j \cdot \mathbf{r}_j$.)

Putting these equations together, this is equivalent to finding the intersection of three different spheres:

$$\begin{bmatrix} (\mathbf{r}_2 - \mathbf{r}_1)^T \\ \mathbf{r}_3 - \mathbf{r}_2)^T \\ \mathbf{r}_1 - \mathbf{r}_3)^T \end{bmatrix} \mathbf{r} = \frac{1}{2} \begin{bmatrix} (\rho_2^2 - \rho_1^2) - (R_2^2 - R_1^2) \\ (\rho_3^2 - \rho_2^2) - (R_3^2 - R_2^2) \\ (\rho_1^2 - \rho_3^2) - (R_1^2 - R_3^2) \end{bmatrix} \tag{13}$$

However, even though we've eliminated the second-order terms from these three equations, we still have two solutions. Recall from linear algebra equations don't have a unique solution when there is redundancy or linear dependence between the equations. If we add up the rows on the right-hand side of the previous equation, we get 0, which indicates that the matrix on the left-hand side is singular:

$$A \triangleq \begin{bmatrix} (\mathbf{r}_2 - \mathbf{r}_1)^T \\ \mathbf{r}_3 - \mathbf{r}_2)^T \\ \mathbf{r}_1 - \mathbf{r}_3)^T \end{bmatrix} \in \mathbb{R}^{3\times 3} \tag{14}$$

To solve this linear dependence problem, we again use **Bezout's Theorem** and keep one of the second-order equations:

$$(\mathbf{r} - \mathbf{r}_1) \cdot (\mathbf{r} - \mathbf{r}_2) = 0 \tag{15}$$

$$(\mathbf{r} - \mathbf{r}_2) \cdot (\mathbf{r} - \mathbf{r}_3) = 0 \tag{16}$$

$$(\mathbf{r} - \mathbf{r}_2) \cdot (\mathbf{r} - \mathbf{r}_2) = 0 \rightarrow (\mathbf{r} - \mathbf{r}_2) \perp (\mathbf{r}_3 - \mathbf{r}_1) \tag{17}$$

I.e. the plane passes through $\mathbf{r}_2$ - this intersecting point is the solution and is known as the **orthocenter** or the **principal point**. Now, all we need is to find the quantity f to find the Center of Projection.

Next, note the following relations between the vanishing points in the inverse plane and $\hat{\mathbf{z}}$, which lies perpendicular to the image plane:

$$\mathbf{r}_1 \cdot \hat{\mathbf{z}} = 0 \tag{18}$$

$$\mathbf{r}_2 \cdot \hat{\mathbf{z}} = 0 \tag{19}$$

$$\mathbf{r}_3 \cdot \hat{\mathbf{z}} = 0 \tag{20}$$

What else is this useful for? Here are some other applications:

- Camera calibration (this was the example above)
- Detecting image cropping - e.g. if you have been cropped out of an image
- Photogrammetry - e.g. by verifying if the explorer who claimed he was the first to make it "all the way" to the North Pole actually did (fun fact: he didn't).

Next, now that we've determined the principal point, what can we say about f (the "focal length")?

For this, let us consider the 3D simplex, which is triangular surface in $\mathbb{R}^3$ given by the unit vectors:

$$e_1 \triangleq \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \tag{21}$$

$$e_2 \triangleq \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T \tag{22}$$

$$e_3 \triangleq \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{23}$$

Using this 3D simplex, let us suppose that each side of the triangles formed by this simplex take length $v = \sqrt{2}$, which is consistent with the l_2 norm of the triangles spanned by the unit simplex ($\sqrt{1^2 + 1^2} = \sqrt{2}$).

Next, we solve for f by finding the value of a such that the dot product between the unit vector perpendicular to the simplex and a vector of all a is equal to 1:

$$\begin{bmatrix} a & a & a \end{bmatrix}^T \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix}^T = 1 \tag{24}$$

$$\frac{3a}{\sqrt{3}} = 1 \tag{25}$$

$$a = \frac{\sqrt{3}}{3} = \frac{1}{\sqrt{3}} \tag{26}$$

This value of $a = \frac{1}{\sqrt{3}} = f$. Then we can relate the lengths of the sides v (which correspond to the magnitudes of the vectors between the principal point and the vanishing points ($||\mathbf{r} - \mathbf{r}_i||_2$)) and f:

$$v = \sqrt{2}, f = \frac{1}{\sqrt{3}} \implies f = \frac{v}{\sqrt{6}} \tag{27}$$

With this, we've now computed both the principal point and the "focal length" f for camera calibration.

1.2.4 Application: Understand Orientation of Camera w.r.t. World

For this problem, the goal is to transform from the world frame of reference to the camera frame of reference, i.e. find a frame of reference such that the unit vectors $\hat{\mathbf{x}}^c, \hat{\mathbf{y}}^c, \hat{\mathbf{z}}^c$ have the following orthogonality properties:

$$\hat{\mathbf{x}}^c \cdot \hat{\mathbf{y}}^c \approx 0 \tag{28}$$

$$\hat{\mathbf{y}}^c \cdot \hat{\mathbf{z}}^c \approx 0 \tag{29}$$

$$\hat{\mathbf{x}}^c \cdot \hat{\mathbf{z}}^c \approx 0 \tag{30}$$

(Note that the c superscript refers to the camera coordinate system.) If the location of the Center of Projection (COP) is given above the image plane as the vector $\mathbf{p}^c$ and the vanishing points are given as vectors $\mathbf{r}_1^c$, $\mathbf{r}_2^c$, and $\mathbf{r}_3^c$ (note that these vanishing points must be in the same frame of reference in order for this computation to carry out), then we can derive expressions for the unit vectors through the following relations:

$$(\mathbf{p}^c - \mathbf{r}_1) /\!\!/ \hat{\mathbf{x}}^c \implies \hat{\mathbf{x}}^c = \frac{\mathbf{p}^c - \mathbf{r}_1^c}{||\mathbf{p}^c - \mathbf{r}_1^c||_2} \tag{31}$$

$$(\mathbf{p}^c - \mathbf{r}_2) /\!\!/ \hat{\mathbf{y}}^c \implies \hat{\mathbf{y}}^c = \frac{\mathbf{p}^c - \mathbf{r}_2^c}{||\mathbf{p}^c - \mathbf{r}_2^c||_2} \tag{32}$$

$$(\mathbf{p}^c - \mathbf{r}_3) /\!\!/ \hat{\mathbf{z}}^c \implies \hat{\mathbf{z}}^c = \frac{\mathbf{p}^c - \mathbf{r}_3^c}{||\mathbf{p}^c - \mathbf{r}_3^c||_2} \tag{33}$$

Then, after deriving the relative transformation between the world frame (typically denoted w in robotics) and the camera frame (typically denoted c in robotics), we can express the principal point/Center of Projection in the camera coordinate frame:

$$\mathbf{r} = \alpha\hat{\mathbf{x}}^c + \beta\hat{\mathbf{y}}^c + \gamma\hat{\mathbf{z}}^c \tag{34}$$

Where (α, β, γ) are the coordinates in the object coordinate system (since $\hat{\mathbf{x}}^c$, $\hat{\mathbf{y}}^c$, and $\hat{\mathbf{z}}^c$ comprise the orthogonal basis of this coordinate system). Then we can express the relative coordinates of objects in this coordinate system:

$$\mathbf{r}' = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^T \tag{35}$$

To transform between frames, we can do so via the following equation:

$$\mathbf{r} = \begin{bmatrix} -(\hat{\mathbf{x}}^c)^T \\ -(\hat{\mathbf{y}}^c)^T \\ -(\hat{\mathbf{z}}^c)^T \end{bmatrix} \mathbf{r}' \tag{36}$$

Note that the matrix defined by: $R \triangleq \begin{bmatrix} -(\hat{\mathbf{x}}^c)^T \\ -(\hat{\mathbf{y}}^c)^T \\ -(\hat{\mathbf{z}}^c)^T \end{bmatrix}$ is orthonormal, since these vector entries are orthogonal to one another. This matrix R is a rotation matrix, which means it is skew-symmetric, invertible, and its transpose is equal to its inverse:

$R^{-1} = R^T$. Therefore, if we wanted to solve the reverse problem that we did above (finding **object coordinates** from **camera coordinates**), we can do so by simply taking the transpose of the matrix:

$$\mathbf{r}' = \begin{bmatrix} -(\hat{\mathbf{x}}^c)^T \\ -(\hat{\mathbf{y}}^c)^T \\ -(\hat{\mathbf{z}}^c)^T \end{bmatrix}^{-1} \mathbf{r} \tag{37}$$

$$= \begin{bmatrix} -(\hat{\mathbf{x}}^c)^T \\ -(\hat{\mathbf{y}}^c)^T \\ -(\hat{\mathbf{z}}^c)^T \end{bmatrix}^{T} \mathbf{r} \tag{38}$$

1.3 Brightness

We'll now switch to the topic of brightness, and how we can use it for surface estimation. Recall that for a Lambertian surface (which we'll assume we use here for now), the power received by photoreceptors (such as a human eye or an array of photodiodes) depends both on the power emitted by the source, but also the angle between the light source and the object. However, the brightness of the object does not depend on the viewing angle between the object and the viewer.

This is relevant for **foreshortening** (the visual effect or optical illusion that causes an object or distance to appear shorter than it actually is because it is angled toward the viewer [6]): the perceived area of an object is the true area times the cosine of the angle between the light source and the object/viewer. **Definition**: Lambertian Object: An object that appears equally bright from all viewing directions and reflects all incident light, absorbing none [5].

Let's look at a simple case: a **Lambertian surface**. If we have the brightness observed and we have this modeled as:

$$E = \hat{\mathbf{n}} \cdot \mathbf{s}, \tag{39}$$

Where $\mathbf{s}$ is the vector between the light source and the object, then can we use this information to recover the surface orientation given by $\hat{\mathbf{n}}$. This unit vector surface orientation has degrees of freedom, since it is a vector in the plane.

It is hard to estimate this when just getting a single measurement for brightness. But what if we test different lighting conditions?:

$$E_1 = \hat{\mathbf{n}} \cdot \mathbf{s}_1 \tag{40}$$

$$E_2 = \hat{\mathbf{n}} \cdot \mathbf{s}_2 \tag{41}$$

$$\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = ||\hat{\mathbf{n}}||_2 = 1 \tag{42}$$

This is equivalent, intuitively, to finding the intersection between two cones for which we have different angles, which forms a planar curve. We then intersect this planar curve with the unit sphere corresponding to the constraint $||\hat{\mathbf{n}}||_2 = 1$. By Bezout's Theorem, this will produce two solutions.

One problem with this approach, however, is that these equations are not **linear**. We can use the presence of **reflectance** to help us solve this problem. Let us define the following:

Definition: Albedo: This is the ratio of power out of an object divided by power into an object:

$$\text{“albedo”} \triangleq \rho \triangleq \frac{\text{power in}}{\text{power out}} \in [0, 1] \tag{43}$$

Though the definition varies across different domains, in this class, we define albedo to be for a specific orientation, i.e. a specific s_i.

Fun fact: Although an albedo greater than 1 technically violates the 2nd Law of Thermodynamics, **superluminous** surfaces such as those sprayed with flourescent paint can exhibit $\rho > 1$.

Using this albedo term, we can now solve our problem of having nonlinearity in our equations. Note that below we use three different measurements this time, rather than just two:

$$E_1 = \rho\hat{\mathbf{n}} \cdot \mathbf{s}_1 \tag{44}$$

$$E_2 = \rho\hat{\mathbf{n}} \cdot \mathbf{s}_2 \tag{45}$$

$$E_3 = \rho\hat{\mathbf{n}} \cdot \mathbf{s}_3 \tag{46}$$

This creates a system of 3 unknowns and 3 Degrees of Freedom. We also add the following constraints:

$$\mathbf{n} = \rho\hat{\mathbf{n}} \tag{47}$$

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{||\mathbf{n}||_2} \tag{48}$$

Combining these equations and constraints, we can rewrite the above in matrix/vector form:

$$\begin{bmatrix} -\hat{\mathbf{s}}_1^T \\ -\hat{\mathbf{s}}_2^T \\ -\hat{\mathbf{s}}_3^T \end{bmatrix} \mathbf{n} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} \implies \mathbf{n} = \mathbf{S}^{-1} \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} \quad (\text{Where } S \triangleq \begin{bmatrix} -\hat{\mathbf{s}}_1^T \\ -\hat{\mathbf{s}}_2^T \\ -\hat{\mathbf{s}}_3^T \end{bmatrix}.) \tag{49}$$

A quick note on the solution above: like other linear algebra-based approaches we've investigated so far, the matrix **S** above isn't necessarily invertible. This matrix is not invertible when light sources are in a **coplanar** orientation relative to the object. If this is the case, then the matrix **S** becomes singular/linearly dependent, and therefore non-invertible.

1.3.1 What if We Can't use Multiple Orientations?

In the case where the orientation of the camera, object, and light source cannot be changed (i.e. everything is fixed), or if it is expensive to generate a surface estimation from a new orientation, then another option is to use different **color sensors**. Most cameras have RGB ("Red-Green-Blue") sensors, and thus we can use the same approach as above, where each color corresponds to a different orientation.

This RGB approach has a few issues:

- RGB has "crosstalk" between color channels (it can be hard to excite/activate a single channel for color without exciting the others as well).
- The object may not be uniformly-colored (which, practically, is quite often the case).

However, despite the drawbacks, this approach enables us to recover the **surface orientation** of an object from a single RGB monocular camera.

A final note: we originally assumed this object was Lambertian, and because of this used the relation that an object's perceived area is its true area scaled by the cosine of the angle between the light source and the object, Does this apply for real surfaces? No, because many are not ideal Lambertian surfaces. However, in practice, we can just use a **lookup table** that can be calibrated using real images.

1.4 References

1. RANSAC Algorithm: http://www.cse.yorku.ca/ kosta/CompVis_Notes/ransac.pdf
2. Bilinear Interpolation: https://en.wikipedia.org/wiki/Bilinear_interpolation
3. Bicubic Interpolation: https://en.wikipedia.org/wiki/Bicubic_interpolation
4. Multilateration: https://www.icao.int/Meetings/AMC/MA/2007/surv_semi/Day02_SENSIS_Turner.pdf
5. Lambertian Surface: Robot Vision, Page 212
6. Foreshortening: https://en.wikipedia.org/wiki/Perspective