CSci 4270 and 6270
Computational Vision, Spring 2025
Lecture 4: Lines and Estimation
January 16, 2025

Overview

- Much of computer vision involves models and model fitting
- Examples include lines, plane, camera models, scene models and motion models.
- Need to understand:
  - Geometry and algebra of a model
  - Properties of the data the model is used to explain
  - Fitting of model to the data
- We'll start with the simplest case: lines.

Outline

Topics for today:

- Line representations
- Line distances
- Estimating the parameters of a line from a sets of (image) data point locations.

This material is closely tied to our Lecture 3 discussion of linear algebra.

Line Representations

Many different representations, starting with the simple version most of us learned first.

- *Slope-intercept:* Parameters are m and b,

  $$y = mx + b.$$

  This has (at least) two problems for our use when working point locations from an an image. What might they be?

- *Point / normal*: Parameters are normal $\boldsymbol{\eta}$ and point on line $\mathbf{p}_0$. Point $\mathbf{p} = (x, y)^\top$ is on the line iff

$$\boldsymbol{\eta} \cdot (\mathbf{p} - \mathbf{p}_0) = 0.$$

  This appears to have four degrees of freedom, but really there are only two.

  - First example of an "over-parameterized" equation — more parameters than geometric degrees of freedom.

- *Implicit form:* Given parameter vector $\mathbf{a} = (a, b, c)^\top$, $\mathbf{p} = (x, y)^\top$ is on the line if

$$ax + by + c = 0.$$

By imposing the constraint $a^2 + b^2 = 1$, we end up with only two degrees of freedom.

- *Closest point form:* Find the point, (ρ, θ), in polar coordinates of the point on the line that is closest to the origin. This is really a special case of the point/normal form:
  - It is pretty easy to see that the vector $(\cos\theta, \sin\theta)^\top$ is normal to (perpendicular to) the direction of the line. Hence, when the line goes through the origin – and therefore $\rho = 0$ — the value of θ can be chosen to enforce normality.

– We can plug $\boldsymbol{\eta} = (\cos\theta, \sin\theta)^\top$ and $\mathbf{p}_0 = (\rho\cos\theta, \rho\sin\theta)^\top$ into the above point/normal formulation. After some algebraic manipulation we can show that a point $\mathbf{p} = (x, y)^\top$ is on the line if and only if

$$x\cos\theta + y\sin\theta - \rho = 0.$$

– This clarifies the relationship between the point/normal form and the implicit form, since $a = \cos\theta$, $b = \sin\theta$, $c = -\rho$.

- *Parametric form:* Given x_0, x_1, y_0, y_1, we can generate points on the line as

$$x(t) = x_1 t + x_0$$
$$y(t) = y_1 t + y_0$$

  Once again this only has two real degrees of freedom.

- Using vectors $\mathbf{p}_0 = (x_0, y_0)^\top$ and $\hat{\mathbf{d}} = (x_1, y_1)$, we can write this parametrically as

$$\mathbf{p}(t) = \mathbf{p}_0 + t\hat{\mathbf{d}}.$$

  We will generally assume that $\hat{\mathbf{d}}$ is a unit vector (hence the "hat" in the notation).

Line Representation Issues

- Need to represent all possible lines and introduce no bias in the representation
- Some representations are better for generating points, while others are better for determining when a point is on a line and for determining the point-to-line distance.
- Some parameterizations have more parameters than degrees of freedom.
  - When we get to the problem of estimating the parameters of a line from a set of points, we will need to handle this problem carefully.

Calculations Needed on Lines

- Is a point on a line?
  - This can be solved with a simple test for each form except the parametric.
- Generating points on a line (as in graphics)
  - Easy using the parametric form; harder using the others
- We have two more calculations — the most important for our purposes:
  - What is the distance of a point to a line?
  - Given a set of points, what is the best-fitting line?

  Both depend on what we mean by "distance".
- We will define two different distances and then look at several solutions.

Distance Measures

A data point will be written as $\mathbf{p}_i = (x_i, y_i)^\top$,

- For the slope/intercept form, the distance is generally measured as the difference in y:
$$d_i = |y_i - mx_i - b|$$
- This is appropriate for "regression" problems where the x value is given and fixed — the "independent variable" — and the y value is measured — the "dependent variable".
- For most other forms the distance is the distance between $\mathbf{p}_i$ and the closest point on the line in a Euclidean distance sense.
- The easiest way to envision this is geometrically with the closest-point formulation.
- With a little bit of work, combining geometric and algebraic arguments, this leads to the simple solution that
$$d_i = |x_i \cos\theta + y_i \sin\theta - \rho|.$$
- For the parametric form, we need to solve for a different t for each data point to find the closest point. This leads back to the implicit and point/normal formulations, and therefore the parametric formulation is not used directly in line fitting.

Least-Squares Line Fitting

It is important to be clear about our goals:

- Given is a set of N point locations $\{(x_i, y_i)\}, i = 1, \ldots, N$. In other words these locations are the data and are "given" to use (for the purposes of the line fitting problem).
- The *unknowns* are the parameters of the line: m and b; a, b and c; or ρ and θ.
- For a given data set, we are to find the parameters that minimize the sum of the squared distances:

$$\sum_{i=1}^{N} d_i^2.$$

- This sum is our "objective function". There are other potential forms of the objective function.
- Formulating and then solving objective functions (through minimization) is a central theme to the computations of computer vision.

Least-Squares Regression

For simplicity, we'll start with the slope/intercept form because it is relatively simple:

- Objective function to minimize

$$E(m, b) = \sum_i (y_i - x_i m - b)^2$$

- We calculate the partial derivatives

$$\frac{\partial E}{\partial m} \qquad \text{and} \qquad \frac{\partial E}{\partial b}$$

- We then set each to 0 and solve for m and b.
- This leads to the linear equation

$$\begin{pmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & N \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} \sum_i x_i y_i \\ \sum_i y_i \end{pmatrix}$$

  which we then solve for m and b.

Orthogonal Least-Squares Regression

Now we turn to the more useful form, at least for fitting lines to image point locations.

- We'll form the objective function in terms of the polar coordinates of the closest point.

$$E(\rho, \theta) = \sum_i (x_i \cos\theta + y_i \sin\theta - \rho)^2.$$

  We want to minimize this objective function with respect to ρ and θ.

- Start by taking the derivative with respect to ρ:

$$\frac{\partial E}{\partial \rho} = -2 \sum_i (x_i \cos\theta + y_i \sin\theta - \rho)$$

- Setting this equal to 0 and solving shows that

$$\rho = \mu_x \cos\theta + \mu_y \sin\theta$$

  where

$$\mu_x = \sum_i x_i / N \qquad \text{and} \qquad \mu_y = \sum_i y_i / N$$

  form the "center of mass" of the data points.

- Substituting this expression for ρ into our original objective function and making the substitution $u_i = x_i - \mu_x$ and $v_i = y_i - \mu_y$, the new objective function becomes

$$E(\theta) = \sum_i (u_i \cos\theta + v_i \sin\theta)^2.$$

- We will solve this for θ and then substitute the resulting value of θ into the expression for ρ to obtain our final answer. Specificallym taking the derivative with respect to θ, setting the result to 0, applying some well-known trigonometric formulas results in

$$\tan 2\theta = \frac{2 \sum_i u_i v_i}{\sum_i (u_i^2 - v_i^2)}.$$

- Letting $\phi = 2\theta$, we have

$$\phi = \tan^{-1} \frac{2 \sum_i u_i v_i}{\sum_i (u_i^2 - v_i^2)},$$

  which is really two solutions ϕ and $\phi + \pi$.

- Finally, we get two solutions for θ

$$\theta = \phi/2 \qquad \text{and} \qquad \theta = \phi/2 + \pi/2.$$

- These correspond to the normal direction (minimum) and tangent direction (maximum) of the line. To choose between them, we must either apply the second derivative test, or simply substitute the two angles into the objective function in turn.

Orthogonal Least-Squares Regression, Solution #2

A second solution can be obtained using the method of Lagrange multipliers, a method I certainly don't expect you to already know, but one that appears in a number of forms in computer vision problems

- Start using a and b instead of $\cos\theta$ and $\sin\theta$ in the above expression:

$$E(a,b) = \sum_i (u_i a + v_i b)^2.$$

- The immediate problem with this is that $a = b = 0$ produces $E = 0$, but we can't have $a = b = 0$.

- We address this by imposing the constraint that $a^2 + b^2 = 1$, which means that $(a,b)^\top$ is a unit vector.

- This is incorporated into a new objective function using a Lagrange multiplier, λ, to produce

$$F(a,b,\lambda) = \sum_i (u_i a + v_i b)^2 - \lambda(a^2 + b^2 - 1),$$

  our new objective function.

- Computing partial derivatives with respect to a, b and λ yields

$$\begin{aligned} \frac{\partial F}{\partial a} &= 2\textstyle\sum_i (u_i^2 a + u_i v_i b) - 2\lambda a \\ \frac{\partial F}{\partial b} &= 2\textstyle\sum_i (u_i v_i a + v_i^2 b) - 2\lambda b \\ \frac{\partial F}{\partial \lambda} &= a^2 + b^2 - 1 \end{aligned}$$

- Setting all three to 0 and rearranging yields (in matrix form)

$$\begin{pmatrix} \sum_i u_i^2 & \sum_i u_i v_i \\ \sum_i u_i v_i & \sum_i v_i^2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \lambda \begin{pmatrix} a \\ b \end{pmatrix}$$

  and $a^2 + b^2 = 1$

- Stated another way, the line normal vector $(a,b)^\top$ is a unit eigenvector of the 2×2 matrix

$$\mathbf{M} = \begin{pmatrix} \sum_i u_i^2 & \sum_i u_i v_i \\ \sum_i u_i v_i & \sum_i v_i^2 \end{pmatrix}$$

- Just like the our previous solution where we directly estimated θ, there are two solutions, producing vectors orthogonal to each other. The solution we want corresponds to the *smaller* eigenvalue of the "moment matrix" $\mathbf{M}$.

  - The solution corresponding to the larger eigenvalue is the tangent direction, the direction along which the data vary most.

- We can solve for ρ in the same manner as the previous solution. Recalling that $c = -\rho$, we have therefore solved the minimization problem for the implicit form of the line.

- This new solution generalizes much more easily and cleanly to higher dimensions.

  - If we have time, I will sketch this solution in class.

What Happens When Things Go Wrong?

- Potential problems:
  - Some data values do not obey the model
  - Some data are from different physical instances — other than what we are trying to describe with our current model parameter estimation — and perhaps more appropriate to a different model.
- We'll draw pictures to illustrate the effect of these "outliers" on line fitting.

Solution Ideas

- Throw out points with biggest errors:
  1. Fit line
  2. Find error distances
  3. Remove points with highest distances
  4. Repeat
- Find line where points are "tightly packed" around it:
  1. "Guess" a line
  2. Count how many points are within a fixed distance τ of the line — these points are "inliers"
  3. Repeat the process many times
  4. Keep the line with the most inliers
- These two approaches are called "reweighted least squares" and — for reasons that will become clear — "Random Sample Consensus" (RANSAC).
- We'll focus on RANSAC.

RANSAC Objective Function

- Given:
  - A set of N points $\{(x_i, y_i)\}$ for $i = 1, \ldots, N$.
  - An error tolerance τ.
- Objective function:

$$E(\rho, \theta) = \text{card}\{i \mid (x_i \cos\theta + y_i \sin\theta - \rho)^2 < \tau^2\}$$

- We will compare this to the least-squares objective function we used earlier.
- Major question: how do we minimize this new objective function to obtain our estimates of ρ and θ.
- Answer: random sampling. We'll explain the intuition in class. The algorithm follows.

RANSAC for Line Fitting

- The algorithm is straightforward

  1. $kmax = 0$; $\hat{\theta} = \hat{\rho} = 0$
  2. for s iterations do:
     (a) Choose two indices, i and j, randomly
     (b) Generate the unique line, θ, ρ through (x_i, y_i) and (x_j, y_j).
     (c) Compute:

$$k = \text{card}\{i \mid (x_i \cos\theta + y_i \sin\theta - \rho)^2 < \tau^2\}$$

     (d) If $k > kmax$ then assign $kmax = k$, $\hat{\theta} = \theta$, $\hat{\rho} = \rho$.

- Often a final step is applied to calculate the least-squares estimate of the line parameters from the $kmax$ inliers to the line $\hat{\theta}$, $\hat{\rho}$.

- As we will discuss in class, the number of samples (iterations) s required can be estimated based on two quantities:

  – The minimum fraction of points in the data that are "inliers" to the line.
  – The minimum desired probability that at least one sample of points has been taken from the inlier set.

Summary

- Line representations: make sure your choice fits your application.
- Try to use a distance measure that makes sense geometrically.
- The derivation of the estimation equations can be adapted to many problems:
  - Start by formulating an error (objective) function based on a reasonable distance measure
  - Solve the problem by computing derivatives with respect to each of the parameters — perhaps in vector form — setting the result equal to $\mathbf{0}$, and then solving algebraically.
  - Need to impose constraints when there are more parameters than degrees of freedom in the problem.
- Always be alert to the possibility that all values in a data set may not be described by a single model instance and therefore that methods like RANSAC may be needed.