

University of Arizona

CS 477/577: Introduction to Computer Vision

Instructor: Kobus Barnard

On-line midterm

October 24, 2024

Credit: 10%

Out of: CS 477 40, CS 577 60

Those enrolled in CS 477 should stop after doing question #10

Estimated time if you approach it like you would an in-class exam (*): 2 hours

Allowed Time: 24 hours

Due Friday, October 25, 11:59PM (Tucson time).

(*) If you write an exam with a time limit, then you give up at sensible points in time. Also, you would likely be more prepared. Finally, you would be focused and stressed out, instead of relaxed and enjoying yourself.

NAME: JOSES OMOJOLA

NetID or CS Login ID: jomojo1

I am enrolled in CS: 477 577 (circle or box or underline one)

Certification: *I certify that the answers I have written on this exam are the result of my own work. I have not consulted with other people, or used their work, or conspired to show answers dishonestly, thereby facilitating others to hand in work that is not theirs.*

I have limited my use of materials to:

- A) Up to 3 textbooks
- B) materials supplied by the instructor on D2L (notes, example write-ups, etc.),
- C) dictionaries including cross language dictionaries (on-line is OK),
- D) personal assignments,
- E) and personal notes or whiteboard photos created by myself.

Signed: 

General instructions:

You need to hand in a PDF. You can use the word document version of this file as a skeleton if that makes it easier. You could also overlay solutions on the PDF file if you know how to do that. Finally, you could writeup your solution using word or Latex and submit the PDF version. If you must use an image of hand-written solutions, **it must be very clear and legible.**

Where applicable please show your work. Part marks will be awarded for solutions going in the right direction. In some cases, intermediate results are required for full marks. The number of points allotted to each question is shown in parenthesis at the end of the questions.

When providing numeric answers, forms that can be reduced to a number with a calculator are sufficient, and often preferred. For example, I prefer $50\sqrt{2}$ to its numerical equivalent.

Good luck and be careful!

Possibly useful facts




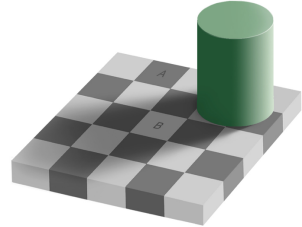
Cross product:



$$(A_x, A_y, A_z) \times (B_x, B_y, B_z) = (A_y B_z - A_z B_y, A_z B_x - A_x B_z, A_x B_y - A_y B_x)$$

Pseudo-inverse:

$$U^\dagger = (U^T U)^{-1} U^T$$

1. For each of the eight images in the table select one of the phrases from the list below that best corresponds to an issue illustrated in class using that image. It is intended that you use each phrase exactly once (the mapping is 1-to-1). Indicate your choice by writing the letter corresponding to the phrase in the box below each image. **(1 point per image, 8 points total).**

			
F	B	A	E

			
C	D	H	G

A	Your brain assumes parallelograms are due to perspective
B	Your brain assumes texture patterns in the world are relatively uniform
C	A classic example of a Lambertian shading failure
D	Your brain assumes that shading is due to geometry of objects in the world
E	Your brain tries to see what is in the world, discounting the effect of illumination
F	Your brain knows objects appear smaller as they get further away
G	Lambertian surfaces have the same appearance from different viewing directions
H	Your brain assumes that light comes from above

2. Briefly explain how scattering can explain Lambertian reflection by dielectric surfaces. (3 points).

When light is reflected off a dielectric surface like a plastic, internal pigments near the surface interact with scattered light particles, and the interaction with different light wavelengths can alter the light ray resulting in a color change along the reflected ray. This helps to explain Lambertian reflection because Lambertian surfaces are surfaces that have lots of scattering. The exiting light rays forgets its original direction and is reflected equally in all directions. Surfaces that behave this way are examples of Lambertian reflection.

Light interaction with non-dielectric surfaces like metals are notably different because the reflected light behaves like a mirror, going in one direction, and the reflections do not change light spectra. Hence, dielectric surfaces are important for explaining Lambertian reflection.

3. Consider a yellow pool ball. In other words, when illuminated by a white light it is seen as yellow (high R and G, low B). As you might recall, “white” light is relative to the camera, and is simply a light that yields equal R, G, and B, for a uniform reflector. You can assume the pool ball is a diffuse (Lambertian) reflector, which has a specular non-Lambertian component for light reflected in mirror-like fashion, which leads to a specular highlight.

To reduce the variance of the answers, assume that for our camera a blue light has RGB (50, 50, 250) and that a yellow light has RGB (200, 200, 50). Note that none of the components are zero. Having any components being zero is rare and could lead to degenerate answers.

- a) Consider a blue light (e.g., a blue LED) illuminating the yellow pool ball. What color (in common, informal language), do you expect the specular highlight to be. Justify your answer a little bit (**2 points**).

The specular highlight on the yellow pool ball illuminated by a blue LED would appear blue (50, 50, 250). In mirror reflections, the reflected color is the same as the incident light. The blue light will interact with blue pigments in the ball, absorbing some of it to make the rest of the ball dark but the specular highlight would reflect the same color as the blue LED source.

- b) Suppose now the pool ball is illuminated with the yellow light. When that yellow light illuminates a uniform reflector, the RGB is the same as the RGB for the body of the pool ball when illuminated by our white light (i.e., (200, 200, 50)). Do you expect the specular highlight to be the same color (chromaticity) as the rest of the pool ball? Justify your answer a bit, perhaps by making up some reasonable numbers for reflectance and light color (**2 points**).

The specular highlight of the yellow pool ball behaves like a uniform reflector and will have the same color as the yellow light (200, 200, 50), however, the rest of the ball will not reflect light uniformly and will have a slightly darker yellow color. Assuming the body of the ball has a reflectance of 0.8, the rest of the ball will have a color

$$(200, 200, 50) \times 0.8 = (160, 160, 40)$$

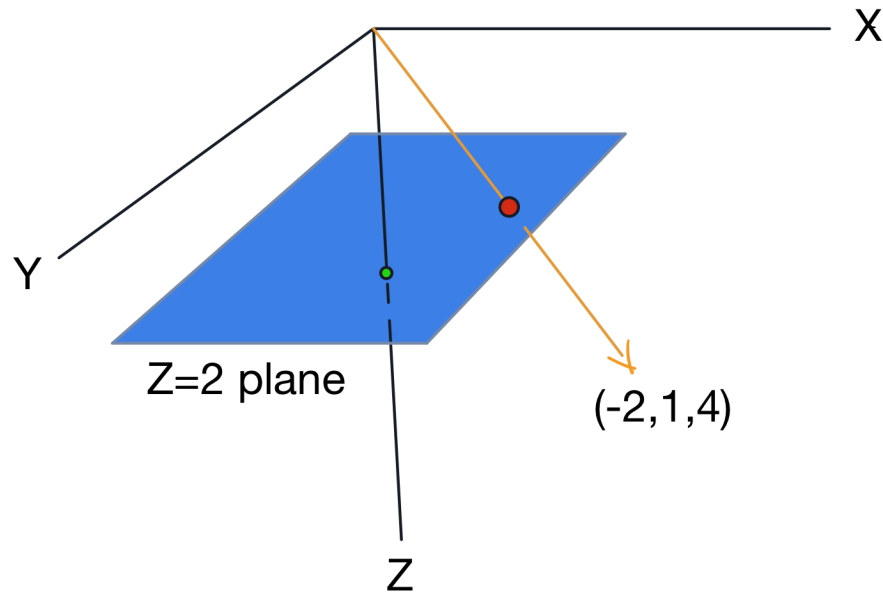
The specular highlight on the other hand has a reflectance of 1 (uniform) with a color

$$(200, 200, 50) \times 1 = (200, 200, 50)$$

4. Suppose you have the surface normals indexed by geographic coordinates (e.g., latitude and longitude) for mountainous surface of an ocean island. Suppose that the surface is relatively smooth, and there are no overhangs. Can you estimate the elevation at every point? If so, describe **briefly** how you would do it. If not, explain what ambiguities or other problems need to be resolved **(3 points)**.

Since we have surface normals and the surface is smooth (minimal noise), we can estimate the slope of the surface at each point. By integrating the normals along a path (e.g. columns first (latitude), then rows (longitude)), we can get relative elevations across the mountainous surface. To convert the relative elevation to absolute values, we can assume the normals over the ocean surface are at sea level, hence zero, and use one of those points as the beginning of our path. If the normals don't extend to the sea surface, we can only recover an estimate of relative elevations, and not absolute values.

5. Consider a standard camera model (based on a pin-hole camera) that has the focus (pin-hole) at the origin, and (virtual) projection plane $z=2$. Consider the point $(-2,1,4)$.
- a) Draw this imaging situation. Include the X, Y, and Z axes, some representation of the imaging plane, the point, and some representation of its projection onto the imaging plane (**2 points**).



The projection onto the imaging plane is shown with the red dot.

- b) Compute the coordinates of the projection of the point onto the plane. Full marks requires showing a bit of work (not just the answer) (**1 point**).

Given a point (x,y,z) , the projected point at the $z=2$ plane will have coordinates

$$(x', y', z') = \left(\frac{2}{z} \cdot x, \frac{2}{z} \cdot y, 2 \right)$$

When $(x,y,z) = (-2,1,4)$

$$x' = \frac{2}{4} \cdot (-2) = -1$$

$$y' = \frac{2}{4} \cdot (1) = 0.5$$

$$z' = 2$$

The projected point coordinates are $(-1,0.5,2)$.

6. Let M be a camera matrix of dimensions 3×4 (as in HW4 and HW5).
- a) Let $P=(X,Y,Z)$ be a point in space with respect to the calibration coordinate system. Determine the coordinates in the image for P introducing any needed notion. This should be in algebraic terms (it is not possible to give a number) **(2 points)**.

We can represent the 3D point $P = (X, Y, Z)$ in homogeneous coordinates by appending 1 as the fourth component.

$$P_{\text{homog}} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

The camera matrix M is of the form

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix}$$

The image point in homogeneous coordinates $p(u, v, w)$ is given as $p = M \cdot P_{\text{homog}}$. Substituting in the matrices, we get:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

The coordinates in the image plane (x, y) are obtained by converting the homogeneous coordinates (u, v, w) into inhomogeneous coordinates, and we can expand the matrix multiplication to get

$$x = \frac{u}{w} = \frac{m_{11}X + m_{12}Y + m_{13}Z + m_{14}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

$$y = \frac{v}{w} = \frac{m_{21}X + m_{22}Y + m_{23}Z + m_{24}}{m_{31}X + m_{32}Y + m_{33}Z + m_{34}}$$

This gives us an algebraic notation for the point $P = (X, Y, Z)$

- b) Given M , is it possible to draw the standard XYZ axes (rays anchored at the origin in the X, Y, and Z directions) for the world coordinate system? Here, “draw” means to color the appropriate pixels in the image plane. If you argue “no” explain what other information you need. If you think “yes”, how would you do it? **(2 points)**

Yes, it is possible to draw the standard XYZ axes. For each real-world axes, we can create a matrix of linearly increasing points e.g.

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ \vdots & \vdots & \vdots \\ n & 0 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ \vdots & \vdots & \vdots \\ n & 0 & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ \vdots & \vdots & \vdots \\ n & 0 & 0 \end{bmatrix}$$

Using M , convert each row of the axis's matrices from world to image coordinates. Plot the image coordinates and connect a line through them to form a ray from the origin.

7. Suppose that a black and white camera sensor has sensitivities that are represented by the vector:
(1, 2, 3, 4, 5, 4, 3, 2, 1)

Provide three distinctly different light signals, also represented by a 9-dimensional row vector that all have the grey value (camera response) of 12 (**2 points**).

The camera response (grey value) (**G**) is given by the dot product of the sensitivity vector (**s**) and the light signal (**l**)

$$G = s \cdot l = \sum_{i=1}^9 s_i \cdot l_i$$

We need 3 light signals with 9 values each, where the sum of their cross product with the sensitivity vector is 12 e.g., for light 1.

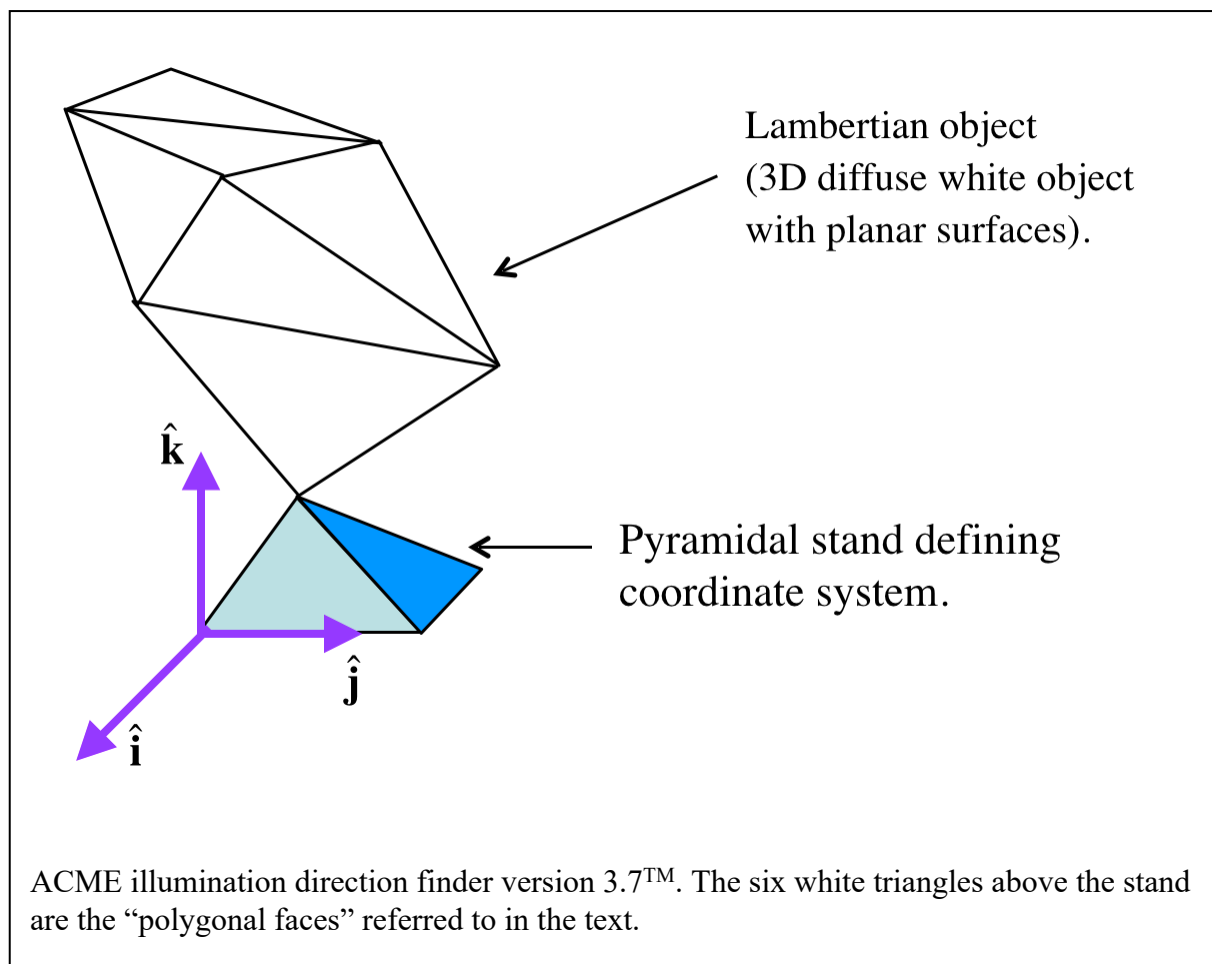
$$\begin{aligned} l_1 &= (1, 0, 1, 0, 1, 0, 1, 0, 1) \\ G &= 1 \cdot 1 + 2 \cdot 0 + 3 \cdot 1 + 4 \cdot 0 + 5 \cdot 1 + 4 \cdot 0 + 3 \cdot 1 + 2 \cdot 0 + 1 \cdot 1 \\ &= 1 + 3 + 5 + 3 + 1 = 12 \end{aligned}$$

Light 2. $l_2 = (1, 5, 0, 0, 0, 0, 0, 0, 1)$

Light 3. $l_3 = (0, 0, 2, 1, 0, 0, 0, 1, 0)$

The camera will have a gray value of 12 for the 3 light signals.

8. The illumination direction finder (illustrated below) is a **pure white** (100% reflectance) polyhedral Lambertian object which is designed so that a program can easily identify each of the polygonal faces (all of them happen to be triangles in the figure), and then look up their surface normal in a coordinate system defined by the illumination finder. The illumination direction finder is placed within an area of interest, and you take a picture of it with a black and white camera. To estimate the illuminant direction, the software first extracts the average shade of gray for each image region corresponding to a polygon on the surface of the object. It then associates this gray value with a unit vector, \hat{n} , for the surface (known from an internal lookup table).



- a) Under what conditions (if any) can you use this setup to determine the **direction** of a single unknown point source in an open area? In particular, does it matter how many polygonal faces are illuminated (i.e., they are not in shadow)? Note that the illumination brightness (relative to the camera) is **not known (3 points)**.

For the setup to accurately determine the light direction, we need uniform light intensity, at least 2 or 3 (if we need to estimate albedo as well) non-coplanar, illuminated (non-shadowed) polygonal faces with known normal vectors. Less than 2 faces give us an under constrained solution, and 3 faces are required to uniquely constrain the direction of the light source relative to the object.

Having more than three illuminated faces helps reduce noise and improve accuracy by making the solution overdetermined, allowing us to solve for the illuminant L using a least squares approach. This provides a more robust estimate of the light direction through averaging and assumes a Lambertian reflectance model and homogeneous material properties for each polygonal face.

- b) Suppose that you are given the observed value for the average brightness for each of M visible faces. You can assume that each of the values are positive. They are given to you in a vector \mathbf{P} . Suppose that you have the normal vectors as rows of a matrix \mathbf{N} . Further suppose that all error contributions are Gaussian. Solve for the illuminant \mathbf{L} . Note that \mathbf{L} contains both the illumination brightness (relative to the camera settings and object albedo) and the illuminant direction **(5 points)**.

Given a brightness vector \mathbf{P} for each visible face \mathbf{M} , and a normal matrix \mathbf{N} where each row n_i is a unit normal vector. Our illuminant vector \mathbf{L} represents both the direction and intensity of the illumination. The brightness observed for each face is assumed to be proportional to the dot product $n_i \cdot \mathbf{L}$.

Our brightness vector \mathbf{P} can thus be formulated as

$$\mathbf{P} = \mathbf{N}\mathbf{L}$$

Since number of faces/observations $M > 3$, the system is overdetermined, and we can solve it using least squares.

$$\mathbf{L} = (\mathbf{N}^T \mathbf{N})^{-1} \mathbf{N}^T \mathbf{P}$$

This is the pseudoinverse solution for an overdetermined system, which minimizes the sum of squared errors assuming Gaussian noise.

9. Suppose you have an RGB image with only two patches, viewed under the canonical light. Under this camera, the RGB of the canonical light is (250,250,250).
- a) Suppose your friend declares that they have a *really* good algorithm for estimating the color of the illuminant from an image given this camera. When applied to this two patch image under the canonical light, what should your friend's algorithm output? **(1 point)**

Since the image is viewed under the canonical light with an RGB value of (250,250,250), a good illuminant estimation algorithm should output this same value, as it represents the color of the canonical light source. The algorithm should output (250,250,250).

- b) Can you provide RGB values for the two patches where the gray world algorithm will do very well, but the max RGB algorithm will do less well, as measured by **chromaticity**? Full marks require a little bit of explanation **(2 points)**

Patch 1 = (50, 175,100)

Patch 2 = (200, 75, 150)

The gray world algorithm

$$\text{Average RGB} = \left(\frac{50 + 200}{2}, \frac{175 + 75}{2}, \frac{100 + 150}{2} \right) = (125, 125, 125)$$

The gray world estimate is neutral and can be multiplied by 2 to get the original illuminant.

Assuming chromaticity and scaling the results to estimate errors, the retrieved error values will be low.

Max-RGB will give an illuminant estimate of (200,175,150). Even when accounting for chromaticity, the estimated illuminant is not neutral and will introduce color biases into the error estimate.

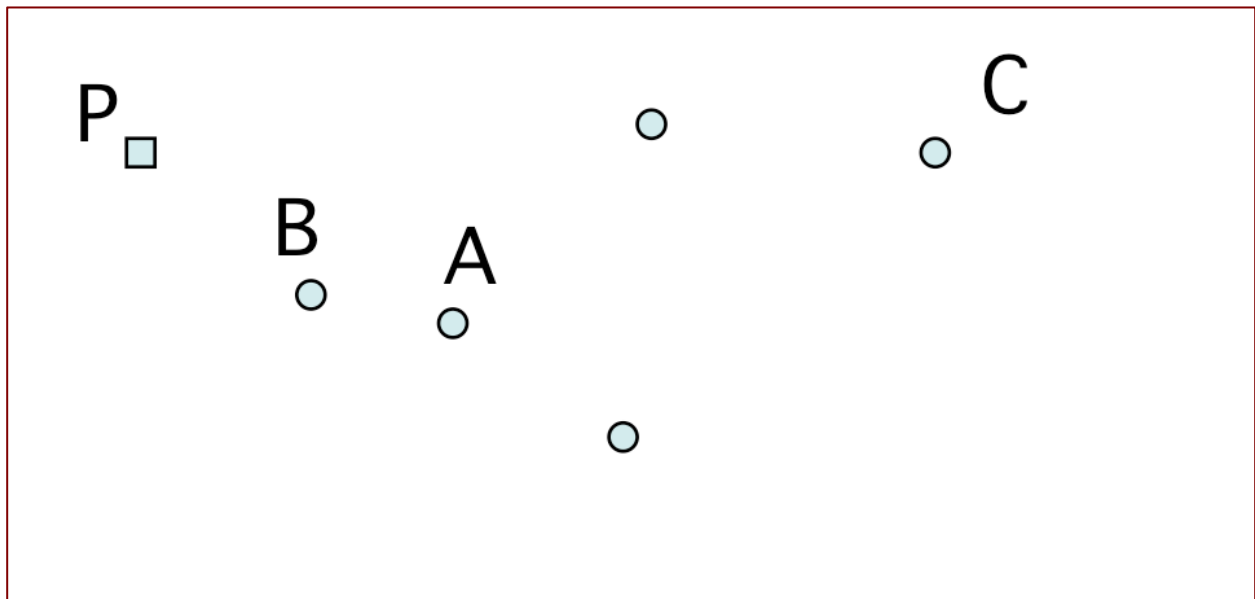
10. The following image shows stars as dots, and a planet in our solar system as a square. This is meant to be a picture taken from earth, perhaps from a telescope, or as seen by yourself. Assume that you have no useful brightness information. However, you can assume that the planet, P, is much, much closer to you than the stars are. Evaluate the two statements below (1 point each, 2 total).

(a) B is closer to A than it is to C [i.e., $d(B,A) < d(B,C)$]

Circle one of : TRUE FALSE CAN'T TELL

(b) P is closer to B than it is to C [i.e., $d(P,B) < d(P,C)$]

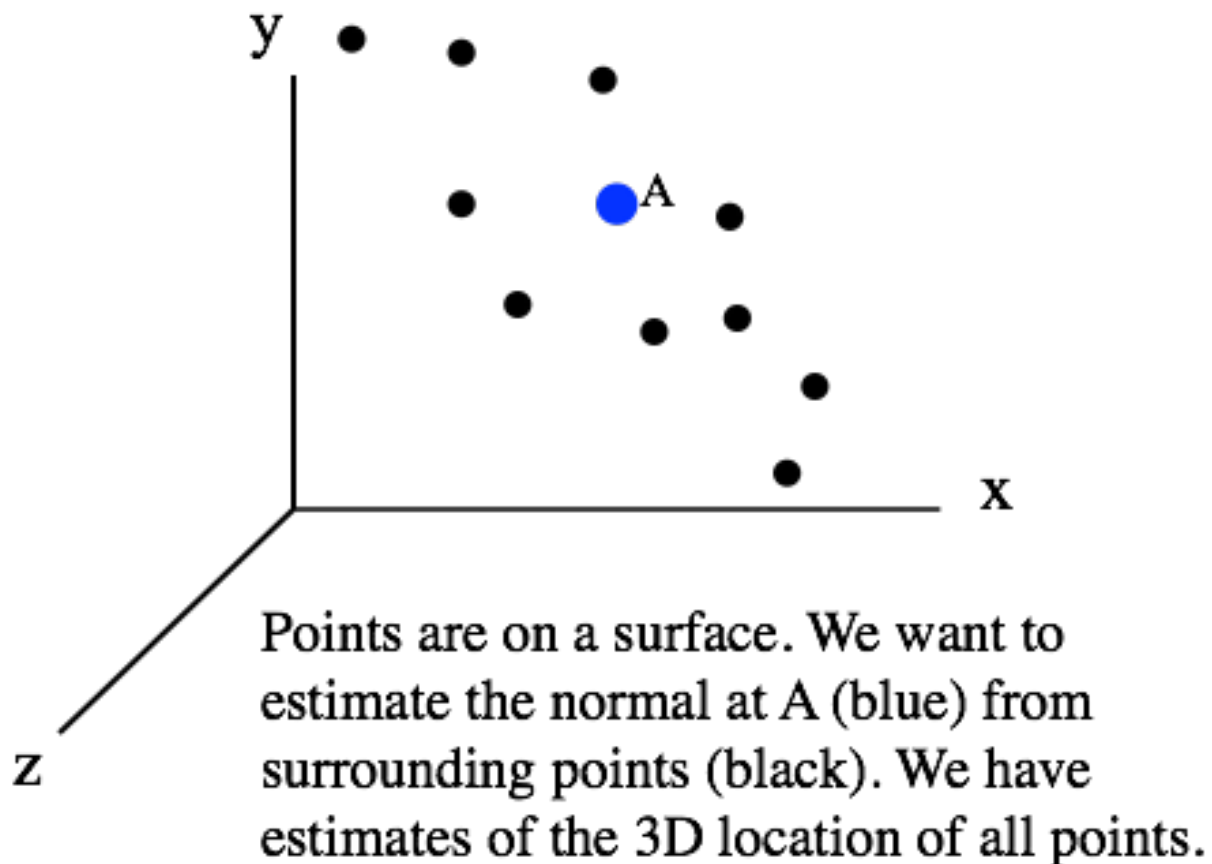
Circle one of: TRUE FALSE CAN'T TELL



[Space for a picture if you choose to provide one. If you sketch one on a scrap piece of paper, but just want to circle the answers, there is no need to provide it. It is not required.]

End of questions shared by CS 477 and CS 577. Only students taking this as CS 577 (graduate credit) should continue beyond this point.

11. The figure below shows some “point cloud data” for a surface, which are 3D points on the surface measured using some sort of scanner that can estimate distance as well as XY position (e.g., Kinect, LIDAR, stereo cameras with fancy software). Consider one of these points, A, and ten more nearby surrounding points, P_i ($i = 1:10$). We assume that all 11 points are on the same patch of a smooth surface. We want to estimate the normal to that surface at point A, whose coordinates are assumed to be measured very accurately, but the surrounding points have noise (see figure).



- a) Set this up as a **homogenous** least squares problem, and indicate how one would be able to solve it with Matlab (or other software package) if we had the accurate coordinates of A and coordinates for the ten P_i measured less accurately. **(6 points)**

Let's define the accurate measured point A and noisy points P as follows

- $A = (x_A, y_A, z_A)$
- $P_i = (x_{P_i}, y_{P_i}, z_{P_i})$ for $i = 1, 2, \dots, 10$

For each surrounding point P_i , we can define the vector from A to P_i as:

$$v_i = P_i - A = \begin{pmatrix} x_{P_i} - x_A \\ y_{P_i} - y_A \\ z_{P_i} - z_A \end{pmatrix}$$

The normal vector $n = (n_x, n_y, n_z)$ must be approximately perpendicular to each vector v_i , which gives us the equations

$$v_i \cdot n \approx 0 \quad \text{for each } i$$

This can be rewritten as:

$$(x_{P_i} - x_A)n_x + (y_{P_i} - y_A)n_y + (z_{P_i} - z_A)n_z = 0$$

And we can stack these equations into a single matrix equation

$$Vn = 0$$

where V is a 10 x 3 matrix constructed from the vectors v_i :

$$V = \begin{pmatrix} x_{P_1} - x_A & y_{P_1} - y_A & z_{P_1} - z_A \\ x_{P_2} - x_A & y_{P_2} - y_A & z_{P_2} - z_A \\ \vdots & \vdots & \vdots \\ x_{P_{10}} - x_A & y_{P_{10}} - y_A & z_{P_{10}} - z_A \end{pmatrix}$$

To ensure that the normal vector has unit length, we can apply a constraint:

$$\|n\|^2 = n_x^2 + n_y^2 + n_z^2 = 1$$

The normal vector that minimizes the squared error while satisfying the unit length constraint, can be estimated from the eigenvalues.

$$M = V^T V$$

The resulting M matrix is 3x3, the direction of the normal vector can be obtained by finding the eigenvector that corresponds to the smallest eigenvalue of M.

The MATLAB implementation would look like

```
% Define coordinates of point A (accurately measured)
```

```
A = [x_A, y_A, z_A];
```

```
% Define coordinates of surrounding points Pi (noisy measurements)
```

```
P = [
    x_P1, y_P1, z_P1;
    x_P2, y_P2, z_P2;
    x_P3, y_P3, z_P3;
    % ...
    x_P10, y_P10, z_P10
];
```

```
V = P - A; % Construct matrix V -row-wise subtraction of A from P
```

```
M = V' * V; % Calculate V^T * V
```

```
[eigenvectors, eigenvalues] = eig(M); % Compute eigval and eigvec
[~, min_index] = min(diag(eigenvalues)); % Index of smallest eigenvalue
n = eigenvectors(:, min_index); % Smallest eigenvector gives the normal
n = n / norm(n); % Normalize the normal vector to ensure unit length
```

- b) Suppose you wanted to weight points so that the further they are away from A, the less you want to penalize error. Specifically, you want to encode that their importance is inversely proportional to their distance from A. How would you change your answer to the above question? **(3 points)**

For each surrounding point P_i calculate the Euclidean distance to point A

$$d_i = \sqrt{(x_{P_i} - x_A)^2 + (y_{P_i} - y_A)^2 + (z_{P_i} - z_A)^2}$$

To avoid division by zero, we can add a small constant value ϵ to the distances. The weights w_i can be defined as the inverse of the distance:

$$w_i = \frac{1}{d_i + \epsilon}$$

This can be used to create a weighted diagonal matrix W , where each diagonal element is the weight w_i :

$$W = \text{diag}(w_1, w_2, \dots, w_{10})$$

We define our V matrix as before, but now we need to minimize the weighted sum of squared errors $\|WVn\|^2$. The resulting weighted equation can be written as:

$$V^T W V n = 0$$

$$M = V^T W V$$

We can find the normal from the eigenvector that corresponds to the smallest eigenvalue of M .

In MATLAB, we only need to incorporate these additional steps into our original results

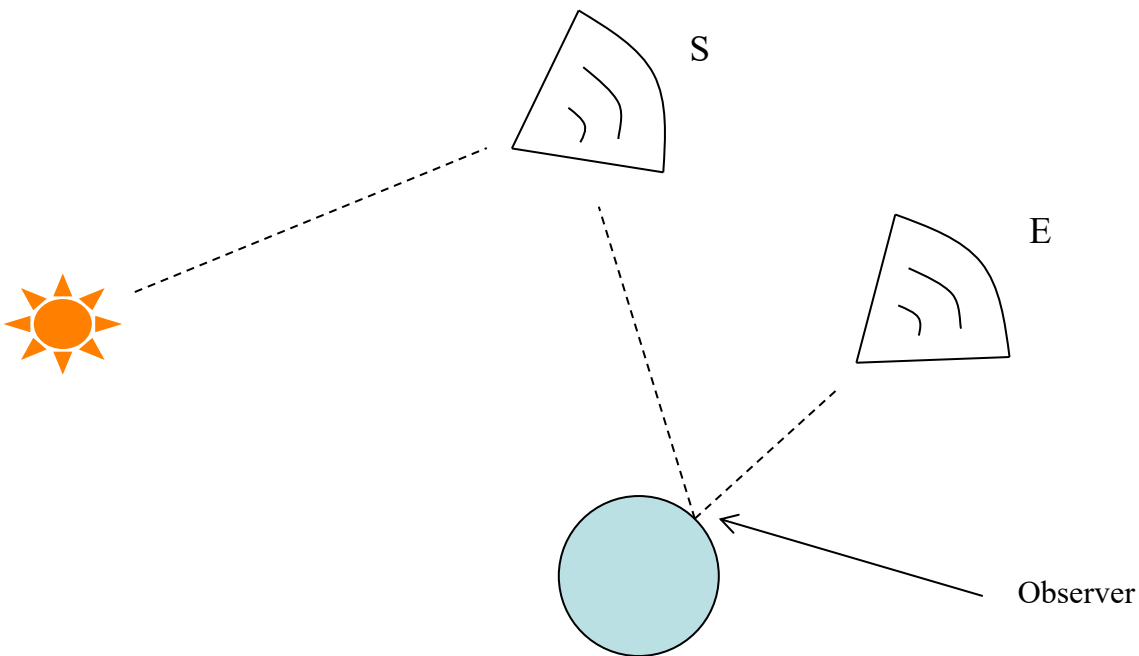
```
distances = sqrt(sum((P - A).^2, 2)); % Euclidean distances
epsilon = 1e-6; % Small constant to avoid division by zero
weights = 1 ./ (distances + epsilon); % Inverse distances
```

```
V = P - A; % Construct matrix V
W = diag(weights); % Construct diagonal weight matrix W
M = V' * W * V; % Calculate W * V^T * V
```

Use the eigenvalue method to solve for n as demonstrated above.

12. Lambertologists(*) claim the moon is a cone! Recall that the fact that the moon appears relatively evenly lit is evidence that the moon is not a Lambertian sphere. But the Lambertologists claim that the mystery that perplexed Lambert in the eighteenth century can be resolved by assuming the moon is a cone. In this problem we will investigate this claim.

Consider the moon in two positions S and E in the figure. In the parts that follow, explain your answers for full credit.



(*) A Lambertologist is a member of the Lambertology society, which believes that all reflection is Lambertian. This society has close ties with the flat Earth society. Understanding this is not necessary to answer the questions.

- a) Suppose that the moon was a Lambertian cone that was always pointed towards the center of the *sun* as in S in the figure. Would it look circular to the observer on earth? **(1 point)**.

No, it wouldn't look circular. When a cone is viewed from the side, it'll resemble a triangle/ellipse.

- b) Again, assuming case S, would it look evenly bright from Earth? **(2 points)**

No, it wouldn't look evenly bright. Varying angles between surface normal and viewing direction would create brightness variations allow segments facing the sun to appear brighter.

- c) Now suppose that the moon was a Lambertian cone that was always pointed to the center of the *earth* via where you were standing (on the earth) as in E in the figure. Would it look circular? **(1 point)**.

Yes, it would look circular. In case E, looking at a Lambertian cone from earth is like a top-down view and at far-away distances, the base of the cone would make it appear circular.

- d) Again, assuming case E, would it look evenly bright from Earth? **(2 points)**

No, it wouldn't. Varying angles between surface normal and light direction would create brightness variations that can make the outer edges appear darker than the center.

13. Your neighbor has expanded their house. You suspect that the plane of a slanted roof is now in a position that violates zoning regulations. You want to test this, but you cannot access the property due to a big, mean dog. You decide that computer vision is your only hope. You rent a plane, and using calibrated cameras and fancy techniques that we have not yet covered, you get estimates for the absolute 3D coordinates for 50 points on the roof. The roof is obviously meant to be planar, and can be assumed to be a plane.

Because of the nature of the above view, the X and Y are accurate, but the Z has lots of error. So, you can assume that all the error is in Z. Full marks require a solution that takes advantage of this assumption.

Your task is then to determine the plane in 3D space containing the surface of the roof. Show how this can be done **(5 points)**.

The general form for a plane in 3D space is:

$$ax + by + cz + d = 0$$

Where (a, b, c) defines the normal vector to the plane, and d is the offset from the origin.

Since we know that error is mainly in the z-coordinate, we can rearrange the plane equation to express z as a function of x and y .

$$z = -\frac{a}{c}x - \frac{b}{c}y - \frac{d}{c}$$

The equation can be simplified to replace the coefficients with

$$z = p_1x + p_2y + p_3$$

To minimize the error in the z-direction, we need to find values for p_1, p_2 , and p_3 that best fit the 3D points in a least-squares sense.

Given fifty measured 3D coordinates, we can represent each point as (x_i, y_i, z_i) for each point $i = 1, 2, \dots, 50$. The z-coordinate for each point can be expressed as:

$$z_i \approx p_1x_i + p_2y_i + p_3$$

Following which, we can set up a system of linear equations in matrix form:

$$Z = Xp$$

where Z is a column vector of the noisy z-measurements, X is a $[50 \times 3]$ matrix of $[x_i, y_i, 1]$, and p is a $[3 \times 1]$ vector of the unknowns $[p_1; p_2; p_3]$. p can be solved using the least squares method as:

$$p = (X^T X)^{-1} X^T Z$$

Once we get a solution for p , we can convert back to the original plane parameters

$$a = -p_1, \quad b = -p_2, \quad c = 1, \quad d = -p_3$$

Thus, minimizing the errors from the z-coordinates.

