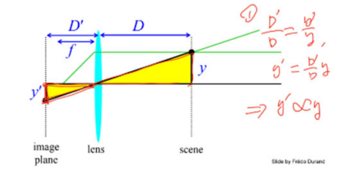
Author: Zhengqi Dong, created at 03/16/2022

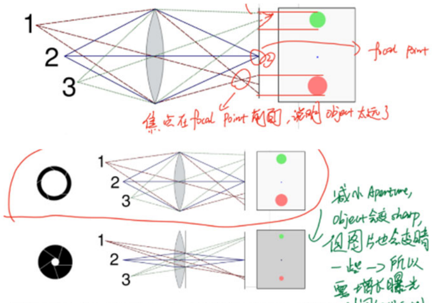
lec2:camera

focal length: how far the plane needs to be, so we can get a sharp image?



- if D' > f or < f, the image will get blurry
- y'/y = D'/D = (D'-f)/f
- y' ∝ y, 焦距和 lens 到 scene 长度成正比
- 1/D' = 1/f – 1/D → D' = fD / (D-f)
- What happen if D is very large? → 1/D becomes 0, then D' is close to f

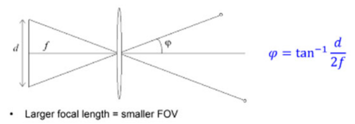
Depth of field(DoF): the distance between nearest and farthest object in a scene that appear acceptably sharp in an image. ==> f-number 越大(e.g., f/22), aperture 越小， 及 DOF 越大, 适合拍自然风光； f-number 越小(e.g., f/1.4), aperature 越大， 及 DOF 越小，适合拍近景或人物。



Changing the *aperture* size affects depth of field

- For a fixed focal length(f) and dist of image plane(D'), if object is too far away(D) → focal points will converge before image plane; on the contrary, if too close → object haven't converged yet, and image will get blurry
- **Effect of aperture**: smaller aperture(e.g., f/22) + longer exposure (with tripod) = Sharp image

Field of view(FOV): how much/wide of scene we can capture?

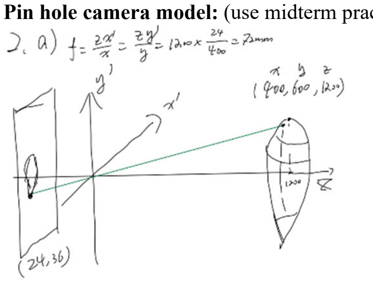


- What determine the FOV? → d(sensor size), f(focal length) ==> $\phi = arctan(d/2f)$
- 视野(ϕ) 和 focal length(f) 是成反比的 →比如说，用长焦可以看得更远，也就是他聚焦的能力变强了，但视野(FOV)就变小了。

Dolly zoom: Continuously adjusting the focal length while the camera moves away from (or towards) the subject → that's how to get the "Vertigo shot".

Types of Lens aberrations(光差) or flaws related to lens: Vignetting, Radical Distortion--caused by imperfect lenses, Spherical aberration, Chromatic and monochromatic aberration--different refractive indices can cause color fringing, Astigmatism(散光)

Pin hole camera model: (use midterm practice as example)



- We want to know where a point (X, Y, Z) in the world gets imaged or landed in our camera?
- How to compute the focus length for a given scene point(400, 600, 1200) and image point (24, 36)? → f = zx'/ x or zy'/ y
- How did we get $f = \frac{zx'}{x}$? → we are deriving this from x'/x = D'/D, where we assume D = z, and D'=f → That's when the distance from lens to camera equals to focal length.

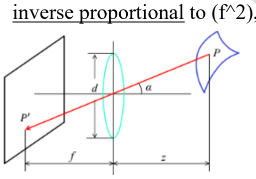
Color filter arrays: Why demosaicing 去马赛 (filling missing pixel value) bias on green? → human is more sensitive to green!

Misc. Digital camera artifacts: Sensor noise, in-camera processing(e.g., over-sharpening can produce halos 光晕), compression algorithm (e.g., JPEG artifacts), Blooming (related to CCD charge overflowing), Color artifacts (e.g., purple fringing from micro lenses).

lec3_light and shading:

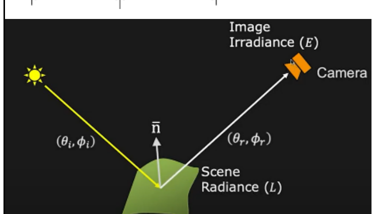
Image irradiance(E) and scene radiance(L):

Image irradiance (E) is proportional to scene radiance(L), area of lens ($\pi d^2/4$), and inverse proportional to (f^2), and view of field(α)



$$E = \left[\frac{\pi}{4}\left(\frac{d}{f}\right)^2 \cos^4\alpha\right] L$$

- Assume: P is a point a surface; P' is the point at image plane.
- **Scene radiance(L):** 发光度, lights emitted from surface P to P'. It measured how much energy leaved surface or being reflected (Watts per sq. meter per steradian) → Note: P can be the original light source, or can be an item that is reflecting light.



- **Image irradiance(E):** 光照度, lights fall on image plane P'. The image irradiance of a point in the image plane is defined to be the power per unit area of radiant energy falling on the image plane. Radiance is outgoing energy; irradiance is incoming energy.

Reflectance properties:

- **Specular reflection:** light is reflected about surface normal
- **Diffuse reflection:** light scatter equally in all direction.
- **Other possible effect:** transparency, refraction, subsurface scattering, Fluorescence, phosphorescence.
- Why does specular reflection make it more difficult to recognize objects in a scene? → Because light is reflected about surface normal, so the viewer can only see the reflected ray along the path of the reflected ray.

Bidirectional reflectance distribution function (BRDF):

- Idea: How bright a surface appears when viewed from one direction when light falls on it from another.
- There are two directions are important to us: 1) the incident direction, where the lights arrived (E), 2) the emitted direction, in which light is being reflected(L) → Definition: the ratio between two.

$E(\theta_i, \phi_i)$: Irradiance due to source in direction (θ_i, ϕ_i)

$L(\theta_r, \phi_r)$: Radiance of surface in direction (θ_r, ϕ_r)

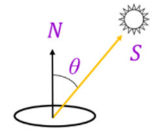
BRDF: $f(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{L(\theta_r, \phi_r)}{E(\theta_i, \phi_i)}$

Unit: 1/sr [Nicodemus 1977]

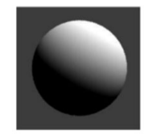
- **isotropic surface:** A point in the surface, if the change of rotation of the surface doesn't affect the brightness of points, then that surface is isotropic → like diffuse reflection → In this case light scatters equally in all direction, so BRDF is constant.

Diffuse reflection: Lambert's law

Definition: A Lambertian surface has a constant BRDF, which means that a Lambertian surface scatters the entire incident light uniformly in all directions, such that **same amount of energy** is seem from any direction. Since its BRDF is independent of outgoing directions, Lambertian surface is also called **ideal matte or diffusion surface**.



$$I = \rho\,(S \cdot N) = \rho \|S\| \cos\theta$$



I: the reflection intensity, how much light being produced → **Known**
ρ: albedo, the amount of light being reflected, range bet [0, 1], where 1 means 100% of ray being reflected, and 0 means no reflection. → **Unknown**
S: direction of light source → **Unknown** in general, bec we can't tell if the light is being reflected by object, or object itself is the original light source?
N: surface normal, or unit vector → Use to detect whether it's 90deg or not → **Unknow**

Q1: Can we recover the surface normal from single image? → Only if the surface normal is makes a 0 deg angle with the direction of light source →However, in general case, we would need at least three images.

Photometric stereo

Problem statement: Aka shape from shading. Assuming the object is a Lambertian object, and we have more than one photo of an object from multiple light source S_j, and given the image pixel value, I(x, y), can we reconstruct the object shape (N) and albedo $\rho(x,y)$. → $I_j(x,y) = (\rho(x,y)N(x,y)) \cdot (kS_j) = g(x,y) \cdot V_j$

Assumption: 1) A Lambertian object; 2) A local shading model; 3)A set of known light sources;4)a set of pictures of object captured by same camera;5)orthographic projection

lec4: color

Grayscale pixel intensity: These pixel values denote the intensity of the pixels. For a grayscale or b&w image, we have pixel values ranging from 0 to 255. The smaller numbers closer to zero represent the darker shade while the larger numbers closer to 255 represent the lighter or the white color.

Digital Image File Formats

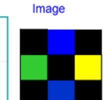
Example of PPM image:

Image file

P3
3 3 255

0	0	0	255	0	0	0	0	0
0	255	0	0	0	0	255	255	0
0	0	0	0	0	255	0	0	0

Image



- The header P3 indicates it is a color image, with size 3 x 3, and the maximum value is 255
- The data is ordered from top to bottom, and left to right. The ordering is RGB.

HSV Color Space: think of HSV as a transformation of an RGB colorspace.

Hue calculation:

- near G: $H = \frac{\pi}{3}\frac{B-R}{V-min(R,G,B)} + \frac{2\pi}{3}$
- near B: $H = \frac{\pi}{3}\frac{R-G}{V-min(R,G,B)} + \frac{4\pi}{3}$
- near R: $H = \frac{\pi}{3}\frac{G-B}{V-min(R,G,B)} + 2\pi$

Value calculation:

$V = (R+B+G)/3$ or $V = max(R,G,B)$

Saturation calculation:

$S = (V - min(R,G,B))/V$

- When would we use HSV over RGB? HSV is more robust towards external lighting changes. This means that in cases of minor changes in external lighting (such as pale shadows, etc.) Hue values vary relatively lesser than RGB values. Besides, it's fast to convert. → But, in general it's not as good as RGB colorspace

Problem of Object Detection: Assuming we have a grayscale image, how do we find bounding box around black object in grayscale background?

Algo1-- Black color & Projections:

- Idea: 1) Count number of black pixels in each row and column; 2) analyze these histograms or projections of black pixels onto x- and y-axes
- Advantage: 1) tell us more about the shape of object, and how prevalent a particular value a pixel is. 2) By further analyzing the pattern of histogram, we can identify where the object is.

Algo2--Flood fill (alternative of algo1):

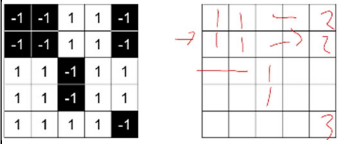
-1	-1	1	1	-1
-1	-1	1	1	-1
1	1	-1	1	1
1	1	-1	1	1
1	1	1	1	41

All labeled 1; 2; 3

1. Convert to binary image, use -1 or 0 to represent black pixels and 1 to represent white pixels
2. scan row by row until first black pixel is reached and label it object 2 (Object label starts from 2 to differentiate from white pixels)
3. Find all neighbors of the current pixel that are black and assign the object label of the current object in a recursive, DFS manner
4. When there are no more black neighbors, continue scanning the image until next black pixel is reached and label it with the next object label and go to step 3.

Author: Zhengqi Dong, created at 03/16/2022

Algo3--Sequential Labeling (a fixed for algo2)**:**



- Why use sequential labeling over flood dill? → Flood Fill uses DFS approach, which would scan the image twice in the worst case; sequential labeling aims to solve this issue, so that we only need to scan the image once(in the average case).
- **Advantage:** Only scan image once. Fast, and good result.

Lec5: Template Matching & Segmentation

What's **template matching**: use an image as template, scan through each frame and try to find the object that matched in the template → Note: This approach is not scale invariant → What to do if we have some imbalance between template and object in the image? → Apply image pyramid techniques → What if we have some lighting changes condition? → Consider using NCC or change color space to one more robust to lighting changes.

Boundary issue of filtering: "Full"→ bigger image, "Same"→ same, "valid"→ smaller

Property of filters:

- Linearity: filter $(I, f_1 + f_2) =$ filter $(I, f_1) +$ filter (I, f_2), adding filter is same as applying two filters separately,
- Shift invariance: order doesn't matter, $\text{filter}(shift(I), f) = \text{shift}(filter(I, f))$

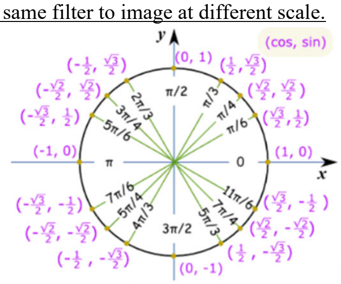
NCC vs SSD (applied in ps1, image stitching)

- Why NCC is better than SSD when comparing similarities? --> it can be greatly affected by the magnitudes of vector (probably caused by high illumination light condition) --> A better and robust idea, is to use NCC, where $\text{Corr} = \frac{1}{n}\sum_{i=1}^{n} \frac{((s_i - mean(s))(m_i - mean(m)))}{\sigma_s \sigma_m}$, where s is all the pixels in subimage, and m is all pixels in template/feature map.
- When consider using Image Pyramid?(applied in lab2) when we have quite imbalance between the size of filter and the target object we want to match → The idea of image pyramid is to generalize this to many different scales --> Since we don't know the true scale, we are looking at, so we are going to apply same filter to image at different scale.

Segmentation vis thresholding:

Method1--Absolute thresholding: compute histogram of an image → select a good threshold to separate bright and dark pixels → assigned foreground and background to two regions. → not work if object is illuminated evenly.

Method2—Percentile(heuristic approach): Assume we can see the image and can guess the percentage of object occupied in an image



Algo 1: K-Mean

- **Idea:** partition an image into K clusters.

1. Pick K cluster centers, either randomly or based on some heuristic method, for example K-means++
2. Assign each pixel in the image to the cluster that minimizes the distance between the pixel and the cluster center
3. Re-compute the cluster centers by averaging all of the pixels in the cluster
4. Repeat steps 2 and 3 until convergence is attained (i.e. no pixels change clusters)

- **Pros:** Simple, fast, and easy to implement
- **Cons:** Need to choose K, sensitive to outliers, and initialization parameters.

Algo 2: Mean shift

- **Idea:** For each pixel, we assign it as a new mean and will run "mean shift", which will move it from its original position to a peak/mode, and all pixels has same mode will be assigned as the same cluster.
- **Attraction Basin:** the region for which all trajectories lead to the same mode/peak
- **Cluster:** all data points in the attraction basin of a mode.
- **Pros:** We don't need to choose K (# of cluster): Being able to find arbitrary number of clusters with a given window size W; 2) Robust to outlier; 3) A good general-purpose segmentation methods.
- **Cons:** 1) Simple but "hill climbing" process is computationally expensive (we need to re-compute the mean at each location, and for each pixel within the window size); 2) not suitable for high-dimensional features)

Watershed algorithm (example of Superpixel algorithms)

1. Choose local minima as region seeds
2. Add neighbors to priority queue, sorted by value
3. Take top priority pixel from queue
   - 1. If all labeled neighbors have same label, assign that label to pixel
   - 2. Add all non-marked neighbors to queue
4. Repeat step 3 until the priority queue is empty (All the non-labeled pixels correspond to the boundary or watershed lines)

- How to reduce the number of regions? → Apply Gaussian or median filter.
- **Pros**: Fast (< 1 sec for 512 x 512 images); preserves boundaries
- **Cons:** 1) Only as good as the soft boundaries (which may be slow to compute); 2) Not easy to get variety of regions for multiple segmentations

Hausdorff distance (applied in ps2)

- **Idea:** it computes the largest degree of mismatch between two sets by measuring the distance of point of A that is farthest from any point of B, and vice versa.

Formula: $H(A, B) = \max(h(A, B), h(B, A))$, where $h(A, B) = \max_{a \in A} \min_{b \in B} ||a - b||$

Lec7 & 8: Edges & keypoints & Alignment

Partial derivative of images: for discrete data, we can approximate using finite differences: $\frac{\partial f(x,y)}{\partial x} = \frac{f(x+1,y) - f(x,y)}{1}$ → If conv with filter [f(x, y) f(x+1, y)].T @ [-1, 1]

Image gradient:

- Gradient orientation, $\theta = \tan^{-1}\frac{\partial f/\partial y}{\partial f/\partial x}$, Gradient magnitude: $|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

Canny Edge Detection

- Step1: Remove/Suppress noise by applying gaussian filter/kernel
- Step 2: Taking derivative, and find the magnitude and orientation of gradient
- Step 3: Apply Non-maximum suppression(Thinning): The edge is too thick, we want to suppress it, so the edge can be clearer, have a "Thinning" contour
- Step 4: Apply Hysteresis Thresholding: And link of connect edge pixels

Q1.Why we need Hysteresis thresholding 滞后阈值法? → Because some pixels along edge might not survive the thresholding, so we need to define two thresholds: We use the high threshold to start an edge curve and use the low-level threshold to continue it.

Q2: Why is orientation and gradient important for edge detection?→ we use gradient to find the edge (the slope is steepest along the gradient), and orientation tells us the direction of greatest intensity change in the neighboring pixel, which useful for NMS.

Q3: What is the problem with thick trail? → a thick edge not only has high value at the real edge location, but also in a small neighborhood around it!

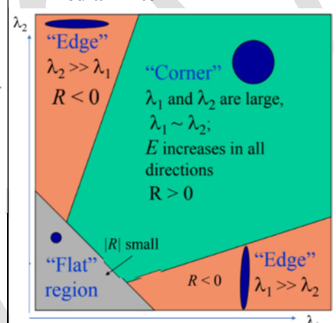
Q4: How's the sigma (Gaussian kernel size) effect out result? → The choice of σ depends on desired feature → large σ find large scale edge, and small σ for fine feature.

Q5: why decomposing a 2-D gaussian filter into two 1-D filters? → to reduce computational cost, for an MxM image with NxN filter, we have $O(N^2M^2)$; By decomposing/factoring it into two 1-D filters, we haves $O(2NM^2)$

Q6: Discuss at least two methods to reduce artifacts at the boundary of an image → Change zero padding to wrap around, or copy edge, or reflection.

Q7: What kind of filter would you want to use to reduce the noise in the image and why? → The image with many "spooky" noises or high-frequency noise are outliers, and median filter is more robus to outliers.

- **Box/average filter** → removed too many details, and can cause high-frequency artifacts
- **Gaussian filter** → help to remove the high-frequency noise, and good at smoothing additive, zero-mean noise (like Normal Distr) --> so it failed for shot noise, or outlier
- **Median filter** → Robust to the outliers



Harris Corner detector

Step1: Compute partial derivatives Ix and Iy at each pixel.

Step2: Compute second moment matrix in a Gaussian window around each pixel: $M = [\sum_{x,y} w(x,y) I_x^2 \ \sum_{x,y} w(x,y) I_x I_y; \sum_{x,y} w(x,y) I_x I_y \ \sum_{x,y} w(x,y) I_y^2]$

Step3: Compute corner response function: $R = det(M) - \alpha \mathrm{Tr}(M)^2$

Step4: Determine threshold R

Step5: Take only points of local maxima of R (NMS)

Q8: How do you know a gradient edge is a corner? → When both the diagonal entries of second moment matrix (λ_1 *and* λ_2) should be large.

Q9: What are good characters of keypoint? → compactness(# ketpoints < # pixels), saliency(each is distinctive), locality(occupy small area), repeatability(same keypoint can be found in different image despite geometric or photometric transformation).

Q10: What transformations is the Harris Corner detector invariant to and why? → Harris Corner detector is NOT invariant to **intensity scaling** $(I \to \alpha I)$, and **image scaling**. Harris Corner detector is invariant to **intensity shifts**$(I \to I + b)$, **image translation** (change of position), and **Image rotation** (bec eigenvalues stay the same).

Voting Techniques:

- **Idea:** let each point vote for all models that are compatible or consistent with, and the model with the highest number of votes is the best fit to the image.
- **Why it works?** → Noise and clutter features will cast votes too, but typically their votes should be consistent with the majority of "good" features.

RANSAC(Random Sample Consensus): (applied in ps3, panorama stitching)

- **Idea:** A robust framework for model fitting in the presence of outlier.
- 机器人火星任务探险 **Algo Summary**: Use SIFT algo to find potential keypoints and features descriptors → compute spatial distance between each pair of descriptor, which will give us the putative matches between two images → run **RANSAC algorithm** to filter out outliers and preserve only the good matches → we would go through a loop. In each iteration, we would choose 4 pairs of keypoint matches randomly and hypothesize a homography matrix. Then, we test how good the homography matrix is in trying to align the two images. This can be done by trying to project keypoints from one image through the homogra phy matrix and calculating the distance between the projected points and the actual keypoints in the other image. → We count the number of inliers for each model (point within inlier threshold), and we keep the model with the most inliers, which is the best model.
- **Pros:** simple, and applicable to many real-world problem
- **Cons:** lots of hyperparameter (e.g., threshold t for inliers, initial number of points s, consensus set size d) and doesn't work well for low inlier ratios.
- **Definition of orthogonal matrix:** A square matrix A is orthogonal is $A^T = A^{-1}, or$ $AA^T = A^T A = I$, where I is identity matrix
- What is a **'singular' matrix**? All of the following conditions are equivalent. We say a square (n × n) matrix is singular if any one of these conditions (and hence all of them) is satisfied: 1) det(A) = 0; 2) some columns or columns are not linearly independent; 3) the matrix is not invertible; 4) the matrix is not full rank (rank < n).