CS180 Final Cheat Sheet

simonxie2004.github.io

Lec 2: Capturing Light

1. Psychophysics of Color:
   Mean=Hue; Var=Saturation; Area=Brightness
2. Image Processing Sequence
   1. Auto Exposure -> White Balance ->
   2. -> Contrast -> Gamma
3. White Balancing Algorithms
   1. Grey World: force avg.color of scene to grey
   2. White World: force brightest object to white
4. Quad Bayer Filters
   1. Why more green pixels? Because human are most sensitive to green light.
5. Color Spaces: RGB, CMY(K), HSV, L*a*b* (Perceptually uniform color space)
6. Image similarity:
   1. SSD, i.e. L2 Distance
   2. NCC, invariant to local avg and contrast

$$\mathrm{NCC}(I,T) = \frac{\sum_{x,y}\left(I'(x,y)\cdot T'(x,y)\right)}{\sqrt{\sum_{x,y} I'(x,y)^2 \sum_{x,y} T'(x,y)^2}}$$

Lec 3-4: Pixels and Images

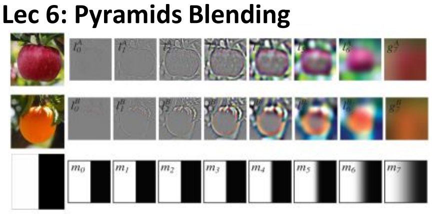
1. Lambertian Reflectance Model:
   1. $(1-\rho)$ absorbed, ρ reflected (either diffusely or specularly)
   2. Diffuse Reflectance: $I(x) = \rho(x)\cdot \mathbf{S}\cdot \mathbf{N}(x)$
2. Image Acquisition Pipeline:

3. Image Processing: Gamma Correction
   1. Power-law transformations: $s = c\cdot r^{\gamma}$
   2. Contrast Stretching: S curve (input gray level to output)
4. Histograms Matching and Color Transfer
5. Image Filter
   1. Cross Correlation $C(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} I(x,y)\cdot K(x+u,y+v)$
   2. Convolution: $C(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} I(x,y)\cdot K(x-u,y-v)$
6. Example: Gaussian Filter
   1. Rule of Thumb: Half-Width = 3σ
7. Image Sub-sampling: Must first filter the image, then subsample (Anti-Aliasing)
8. Image Derivatives: To avoid the effects of noise, first smooth, then derivative (i.e. look for peaks in $\frac{d}{dx}(f*g)$)
   1. This leads to LoG or DoG filters

Lec 5: Furrier Transform

1. Math:
   1. Furrier Transform: $F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}\,dt$
   2. Inv Transform: $f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega t}\,d\omega$
2. Low pass, High pass, Band pass filters
   1. Details = High-freq components;
   2. Sharpening Details: $f+\alpha(f-f*g) = (1+\alpha)f - af*g = f*\left((1+\alpha)e - ag\right)$
   3. Remark that $(1+\alpha)e - \alpha g$ is approximately Laplacian of Gaussian.

Lec 6: Pyramids Blending



$$l_k = l_k^A * m_k + l_i^B * (1-m_k)$$

1. Gaussian Pyramids and Laplacian Pyramids (Remember to add lowest freq!)

--- Some other image algorithms ---

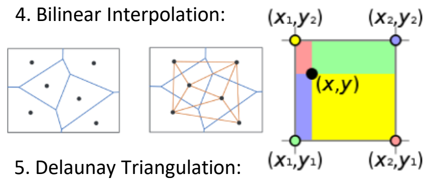
2. Denoising: Median Filter
3. Lossless Compression (PNG): Huffman Coding
4. Lossy Compression (JPEG): Block-based Discrete Cosine Transform (DCT)
   Compute DCT Coefficients; Coarsely Quantize; Encode (e.g. with Huffman Coding)

Lec 7-9: Affine Transformations

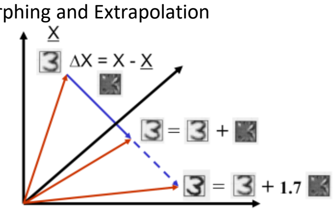
1. Scaling, Shearing and Translation: $S = \begin{bmatrix} a & sh_x & t_x \\ sh_y & b & t_y \\ 0 & 0 & 1\end{bmatrix}$
2. Rotation: $R = \begin{bmatrix}\cos(\theta) & -\sin(\theta) & 0\\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1\end{bmatrix}$
3. Calculating Affine Matrix:

$$\begin{pmatrix} a & b & tx \\ c & d & ty \\ 0 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1\end{pmatrix} = \begin{pmatrix} x_1' & x_2' & x_3' \\ y_1' & y_2' & y_3' \\ 1 & 1 & 1\end{pmatrix}$$

4. Bilinear Interpolation:

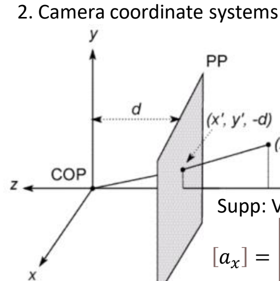


5. Delaunay Triangulation:
   Dual graph of Voroni Diagram
6. Morphing and Extrapolation



Lec 10: Cameras

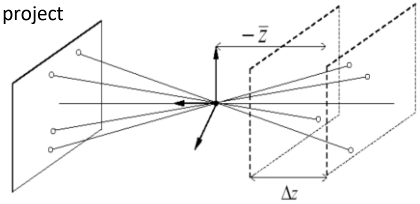
1. Pinhole camera model
   1. Defines Center of Projection and Image Plane
   2. Defines Effective Focal Length as d
2. Camera coordinate systems



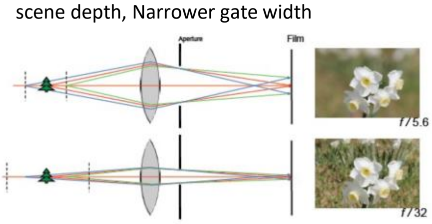
Supp: Vector Cross Product

$$[a_x] = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0\end{bmatrix}$$

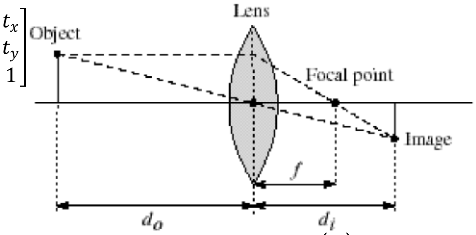
3. Projection:
   1. Perspective Projection: $(x,y,z)\to\left(-d\frac{x}{z}, -d\frac{y}{z}, -d\right)\to\left(-d\frac{x}{z}, -d\frac{y}{z}\right)$
   2. Orthographic Projection: $(x,y,z)\to(x,y)$
      special case when d(COP, PP) = inf
   3. Weak Perspective/Orthographic:
      $\Delta z << -\bar{z}, (x,y,z)\to(-mx,-my), m = -\frac{f}{\bar{z}}$
      Special case when scene depth is small relative to avg. distance from camera
      Equivalent to scale first then orthographic project



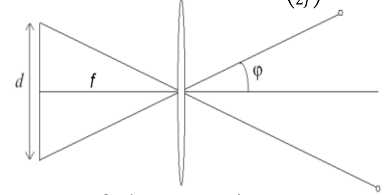
   4. Spherical Projection: $(\theta,\phi)\to(\theta,\phi,d)$
4. Camera Parameters
   1. Aperture: Bigger aperture = Shallower scene depth, Narrower gate width



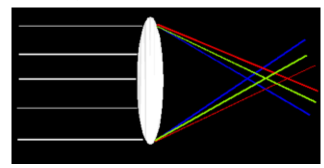
   2. Thin Lenses: $\frac{1}{d_o}+\frac{1}{d_i} = \frac{1}{f}$



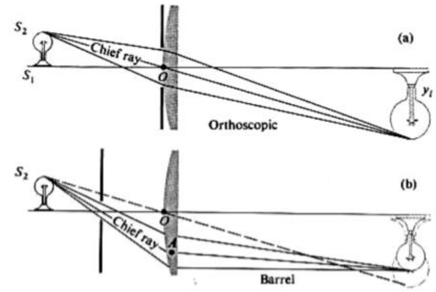
3. FOV (Field of View): $\phi = \tan^{-1}\left(\frac{d}{2f}\right)$



4. Exposure & Shutter Speed
   Example: F5.6+1/30Sec = F11+1/8Sec
5. Lens Flaws
   1. Chromatic Aberration: Due to wavelength-dependent refractive index, modifies ray-bending and focal length



   2. Radial Distortion: Normal, Barrel, Pin-cushion



Lec 11: Perspective Transforms

1. Formula: H is a 3x3 homography matrix, rank=8

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_Nx_N' & -y_Nx_N' \\ 0 & 0 & 0 & x_N & y_N & 1 & -x_Ny_N' & -y_Ny_N'\end{pmatrix}\cdot\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h\end{pmatrix} = \begin{pmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ \vdots \\ x_N' \\ y_N'\end{pmatrix}$$

Solution: Least Squares, $x = \left(A^TA\right)^{-1}A^Tb$

Lec 12-14: Feature Extraction

1. Change in appearance of window W for the shift $[u,v]$ is:

$$E(u,v) = \sum_{(x,y)\in W}[I(x+u,y+v) - I(x,y)]^2$$

2. Second moment matrix M: an approximate of local change on images.

$$M = \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2\end{bmatrix}$$

Corner strength:

$R = \det(M) - k*\mathrm{tr}(M)^2$ or $\det(M)/\mathrm{tr}(M)$

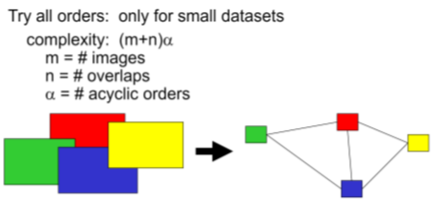
Remark: for flat areas, both λ_1,λ_2 are small; for edges, one of the λ is big; for corners, both are big.

3. Scale Invariant Detection: choose the scale of best corner independently!
4. Feature Selection: ANMS (only those points that are a maximum in a neighborhood of r pixels are retained)

$r_i = \min_j |x_i - x_j|, \text{s.t.} f(x_i) < c_{\text{robust}} f(x_j), X_j \in I$

5. Feature Descriptor (Multi-scale Oriented Patches): 8x8 oriented patch, described by (x, y, scale, orientation)
   Maybe normalized by $I' = (I-\mu)/\sigma$

6. Matching Feature:
Step 1: Lowe's Trick, match(1-NN) - match(2-NN)
Step 2: RANSAC (random choose 4 points; calc homography; calc outliers; finally select best homography)
7. Further Techniques: Order images to reduce inconsistencies



Do the loop: match images - order images - match images - ...
8. Optical Flow Algorithm
$I(x+u, y+v) - H(x,y) \approx [I(x,y) - H(x,y)] + I_x u + I_y v = I_t + \nabla I \cdot [u,v] = 0$
The component of the flow in the gradient direction is determined; The component of the flow parallel to an edge is unknown.
To have more constraint, consider a bigger window size!

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

Solve by least square: (Lukas & Kanade, 1981)

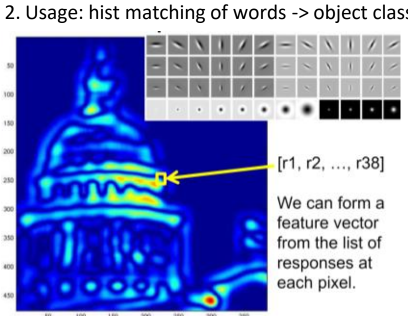
$$(A^T A)\mathbf{d} = A^T \mathbf{b}$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

This is solvable when: no aperture problem.
How to make it even better? Do it multi-hierachical!

Lec 15-16: Texture Models

1. Human vision patterns
   1. Pre-attentive vision: parallel (~100-200ms)
   2. Attentive vision: serial search (~50ms)
2. Order statistics of Textures:
   1. First order: mean, var, std, ...
   2. Second order: co-occurence matrix, contrast, ...
3. Introduction: Cells in Retina
   1. Receptive field of a retinal ganglion cell can be modeled as a LoG filter. (Corner Detectors)
   2. Cortical Receptive Fields -> (Line/Edge Detectors)
   3. They are connected just like a CNN network.
4. From Cells to Image Filters: [Filter Banks]
   1. Detects Statistical unit of texture (texton) in real images: from object to bag of "words"
   2. Usage: hist matching of words -> object classify



5. Image-2-Image Translation
Eg: Calc Depths, Normals, Pixelwise-Segmentation...
Answer: Encoder+Decoder, Convolutions and Pooling
How about missing details when up-sampling? Copy a high-resolution Version! (U-Net)
How about loss function? L2 don't work for task: image colorization
Use per-pixel multinomial classification!
$L(\hat{\mathbf{Z}}, \mathbf{Z}) = -\frac{1}{HW}\sum_{h,w}\sum_q \mathbf{Z}_{h,w,q} \log(\hat{\mathbf{Z}}_{h,w,q})$ where q is probability of label q; This is a cross-entropy.

6. Universal loss for Img2Img Tasks: GAN
D's task: $\text{argmax}_D \mathbb{E}_{x,y}[\log D(G(x)) + \log(1 - D(y))]$
G's task: Tries to synthesize images that fool the best D: $\text{argmin}_G \max_D \mathbb{E}_{x,y}[\log D(G(x)) + \log(1 - D(y))]$
Example Img2Img Tasks: Day->Night, Thermal->RGB, ...
7. Revision of an Early Vision Texture Model:
Multi-scale filter decomposition (Convolve both images with filter bank) + Match per channel histograms (from noise to data); Collapse pyramid
8. Make it better?
Match joint histograms of pairs of filter responses at adjacent spatial locations, orientations, scales, ...
9. Make it more modern: Use CNN.
Now, we use Gram Matrices on CNN Features as texture features.
Define CNN output of some layer as:

$$F_{N\times C} = [f_1, f_2, \dots, f_N]^T$$

We have:

$$G = FF^T = \begin{bmatrix} \langle f_1, f_1\rangle & \cdots & \langle f_1, f_N\rangle \\ \vdots & & \vdots \\ \langle f_N, f_1\rangle & \cdots & \langle f_N, f_N\rangle \end{bmatrix}$$

This describes the correlation of image feature f_i and f_j, which are both length C (channel) vectors.

Remark: The CNN used here is just a pre-trained texture-recognition CNN network, where VGG-16 or VGG-19 nets can be used.

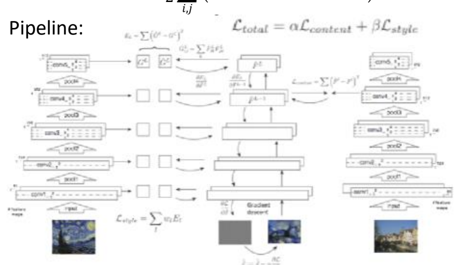
Basically, select any CNN network that is trained to map from image to label (e.g. "dog") will recognize features totally fine. They are already trained on ImageNet dataset.

10. Similar task: artistic style transfer
Loss Function Design:

$$L_{\text{style}} = \sum_l \frac{1}{C_l^2 H_l^2 W_l^2} \| G_l(\hat{I}) - G_l(I_{\text{style}}) \|_F^2$$

$$L_{\text{content}} = \frac{1}{2}\sum_{i,j}\left(F_{i,j}^{\text{generated}} - F_{i,j}^{\text{content}}\right)^2$$

Pipeline: $\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$



Lec 17-18: Diffusion Models

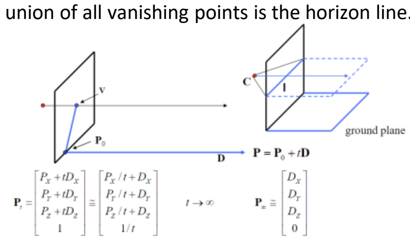
1. Sampling Methods: DDPM v.s. DDIM
2. Make Sampling Faster: Train a "distilled" network
3. Edit desired area: Genearte a Mask that a word correspond to.
Common Image Generating Models:
Parti: self-regressive model; generates block by block
Imagen: Diffusion; Dalle-2: Parti + Imagen

Lec 19: Sequence Models

1. Shannon, 1948: N-gram model; Compute prob. dist. of each letter given N-1 previous letters (Markov Chain)
2. Video Textures, Sig2000:
Compute L2 distance $D_{i,j}$ for between all frames
Transition costs: $C_{i\to j} = D_{i+1,j}$; Probability Calculated as: $P_{i\to j} \propto \exp(-C_{i\to j}/\sigma^2)$
3. Image Analogies Algorithm: Process an image by example (A:A' :: B:B') (Siggraph 2001)
Compare area of pixels (e.g. 10*10) from imgA and B. Find the best match, then copy some smaller area of pixels (e.g. 3*3) from imgA' to imgB'
Remark: New method use VQGAN and MAE.
4. Word Embedding (word2Vec, GloVe)...
Attention + Prediction: Word sequence tasks possible explanation: different layers (attention+prediction) works for different functions (syntax, semantics, ...)

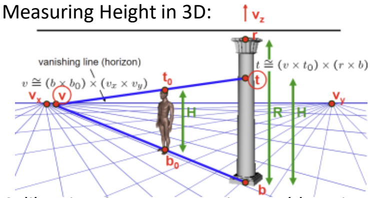
Lec 20-21: 3D Vision Geometry

1. A vanishing line in the image correspond to a plane of rays through origin.
An image may have more than one vanishing point.
The union of all vanishing points is the horizon line.

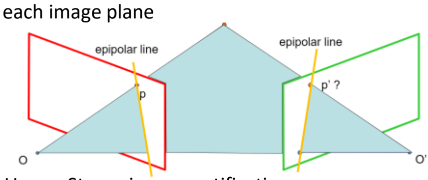


$$P_t = \begin{bmatrix} P_X + tD_X \\ P_Y + tD_Y \\ P_Z + tD_Z \\ 1 \end{bmatrix} \cong \begin{bmatrix} P_X/t + D_X \\ P_Y/t + D_Y \\ P_Z/t + D_Z \\ 1/t \end{bmatrix} \quad t\to\infty \quad P_\infty = \begin{bmatrix} D_X \\ D_Y \\ D_Z \\ 0 \end{bmatrix}$$

2. Measuring Height in 3D:



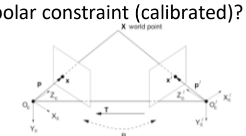
3. Calibrating Camera: Learning Problem, just like solving homography.
4. Epipolar geometry
Baseline: the line connecting the two camera centers
Epipole: point of intersection of baseline with the image plane
Epipolar plane: the plane that contains the two camera centers and a 3D point in the world
Epipolar line: intersection of the epipolar plane with each image plane



Usage: Stereo image rectification
Reproject image planes onto a common plane parallel to the line between optical centers
Then pixel motion is horizontal after transformation
Two homographies (3x3 transforms), one for each input image reprojection
5. How to describe epipolar constraint (calibrated)?

$$\mathbf{X}'\cdot(\mathbf{T}_\times \mathbf{R}\mathbf{X}) = 0$$

Let $\mathbf{E} = \mathbf{T}_\times\mathbf{R}$



This holds for the rays **p** and **p'** that are parallel to the camera-centered position vectors **X** and **X'**, so we have: $\mathbf{p}'^T\mathbf{E}\mathbf{p} = 0$

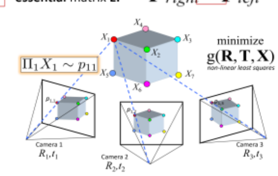
E is called the **essential matrix**, which relates corresponding image points [Longuet-Higgins 1981]

Properties of $E = T_x R$:
Ex' is the epipolar line associated with x' ($l = Ex'$)
$E^T x$ is the epipolar line associated with x ($l' = E^T x$)
$Ee' = 0$ and $E^T e = 0$; E is singular (rank two)
E has five degrees of freedom (3 for rotation R, 2 for translation t since it is up to a scale)
6. How to describe epipolar constraint (uncalibrated?)
Use image points to estimate F (fundamental mat)!
(Remark: F has rank 2, must drop out least singular)

$\mathbf{p}_{(right)}^T\mathbf{E}\mathbf{p}_{(left)} = 0$ From before, the essential matrix **E**. $\quad \overline{\mathbf{p}}_{right}^T \mathbf{F} \overline{\mathbf{p}}_{left} = 0$

Lec 22: SFM

1. Bundle Adjustment Problem:



- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m}\sum_{j=1}^{n} \underbrace{w_{ij}}_{\substack{\text{indicator variable:} \\ \text{is point } i \text{ visible in image } j\text{ ?}}} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)}_{\substack{\text{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\text{observed} \\ \text{image location}}} \right\|^2$$