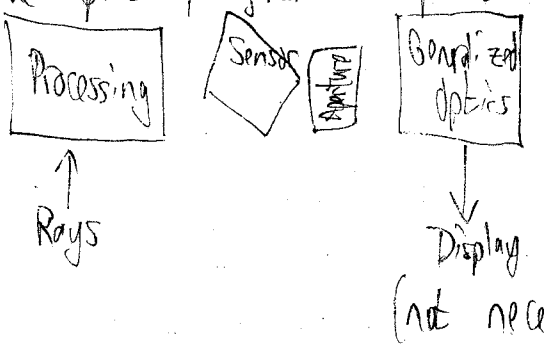
2016-01-14, CS6475, Computational Photography

M01 02

- Question: How computation influences photography, and how light forms an image
- Camera 2.0 \ FrankenCamera - one example
- Combines: Computing
  Digital sensors
  Modern optics
  Actuators
  Smart lights
  ] to generate newer forms of artifacts & photography
  - to 'escape' limitations of traditional film cameras or even 'normal' digital
- Benefits: Unbounded dynamic range!
  - Variable focus & DOF, resolution, lighting, reflectance
- Elements: 3D scene, illumination, optics, sensor (photovoltaic - or chemical!), processing, display, user
  - We may embed computation in all aspects here!
- That whole pipeline: Rays → Pixels
  - Optics also have aperture, "generalized optics" - 'Novel illumination'
  - Novel camera:



#

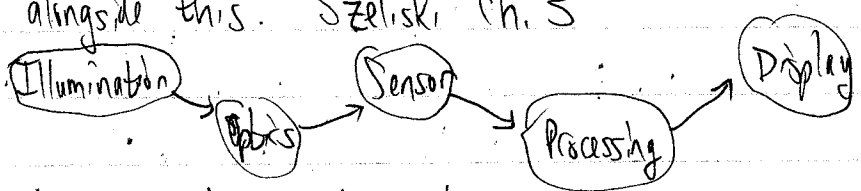
01-03 - Dual Photography
- Remember 'novel illumination' from last unit?
- We may use a projector as a controllable light source & aperture. That is, we may illuminate only with specific rays.
- We can then basically swap camera & light source.
- Helmholtz Reciprocity

01-04 - Panorama stitching
- See: Autostitch, Cloudburst Research
360 Panorama, Occipital

01-05

2016-01-20

02-01 - Reading alongside this: Szeliski Ch. 3
- Recall:



- How do we make an image 'computable'?
- "Pixel", or "picture element": contains light intensity at some location (i,j). $I(i,j)$ = some numerical value & note that 'I' can be discrete or continuous. It's just a function.

2016-01-25

Szeliski Ch.3 - "point operator": Input pixel value → Output pixel value
- Also called 'point processes'
- AHE, adaptive histogram equalization
- CLAHE, contrast (gain) limited version

2016-01-26(b), CS6475

- Linear filter - most common neighborhood operator
  - e.g. convolution;
- Other neighborhood operators: dilation
- How to identify linearly separable filters (one way):
  - Treat 2D kernel as matrix & take SVD, and see if only the first singular value is non-zero. See eg. 3.21, p138
- Image warping - transforms <u>domain</u>, not range, of image.
  - "texture mapping" is closely related
- See p[illegible], table 3.5 - 2D coordinate xforms
- Mipmapping, p167

2016-01-27, CS6475

02-04

- Smoothing
- Edge options: Wrap around
  Copy edge
  Reflect across edge
- Neighborhood size $k \rightarrow 2k+1$
- To normalize: $G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$?
  - For uniform smoothing

- Cross-correlation: In signal processing, "a measure of similarity of two waveforms as a function of a time-lag applied to one of them".
- Also known as a 'sliding dot-product' / 'sliding inner-product'
- $G = h \otimes F$ → $G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]\, F[i+u, j+v]$

→ Replace each pixel with a linear combination of its neighboors

- Note that...

$F[i,j]$:

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

$\otimes$

$h[i,j]$:

a	b	c
d	e	f
g	h	i

$=$

$G[i,j]$:

0	0	0	0	0
0	i	h	g	0
0	f	e	d	0
0	c	b	a	0
0	0	0	0	0

- Kernel is flipped in both axes!

- Convolution: Mathematical op. on functions F & h. Produces third function, 'modified' version, which gives the area of overlap between two functions. It shows the amount one is translated by.

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]\, F[i-u, j-v], \quad G = h * F$$

Note $i-u, j-v$ rather than $i+u, j+v$

↑ of kernel

- This is like a flip in both dimensions, then cross-correlate.
- It is linear & shift-invariant (image shift is irrelevant)
- Commutative & associative: $F * G = G * F$, $(F * G) * H = F * (G * H)$
- $F * E = F$ if E = unit impulse

2016-01-27(b). CS6475

- Convolution is separable. If filter can be separated into horizontal & vertical parts they may be applied to horizontal lines and then vertical lines, respectively.

02-06

- Gradients
  - Filtering to find features, why:
    - To reduce amount of data & preserve useful info by mapping raw pixels to intermediate representation
  - One place to start for this: Edges
  - The aim is to find discontinuities in the scene.
    - Surface normal
    - Depth
    - Surface color
    - Illumination

    - Information theory view: Edges encode change, therefore edges efficiently code an image.
  - Edge is "where there is a rapid change in the image intensity function"
    → Differentiate & find extrema!
  - We may use a kernel to find derivative
  - Gradient = measure of change in image function
  - $\nabla F = \left[\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}\right]$ → $\tan^{-1}\left(\frac{\partial F}{\partial x} / \frac{\partial F}{\partial y}\right)$
    - Points toward direction of most rapid intensity increase
    - Gradient magnitude $\|\nabla F\| = \sqrt{\left(\frac{\partial F}{\partial x}\right)^2 + \left(\frac{\partial F}{\partial y}\right)^2}$ → Edge strength

- In discrete form: $\frac{\partial F}{\partial x} \approx \frac{F(x+1,y)-F(x,y)}{1}$, $\frac{\partial F}{\partial y} \approx \frac{F(x,y+1)-F(x,y)}{1}$

- Note that the edge we want is perpendicular to the gradient!

2016-01-28

02-07

- Edges
  - Note that $\frac{\partial F}{\partial x} = [-1 \quad 1] \cdot [F(x,y) \quad F(x+1,y)]^T$ - dot product
  - That's just cross-correlation with $\boxed{-1 \mid 1}$
  - Or, $\boxed{\begin{matrix}-1\\1\end{matrix}}$ for $\partial F/\partial y$
  - But ideally they should be symmetric about 'image pixel' and these don't really have a 'middle'.
  - One common one:

"Finite Differences" → numerical solution for Diff Eq w/ derivative approximation

$$H_x = \begin{bmatrix} 0 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{bmatrix} \qquad H_y = \begin{bmatrix} 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \end{bmatrix}$$

  - These average 'left' & 'right' derivatives, & 'top' & 'bottom'.
  - Better in some ways, well-defined.
- Other common kernels:

	H_x	H_y	
Prewitt	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	
Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	- Heavier on center part
Roberts	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	- Basically uses diagonals

2016-01-28(b), CS6475

- Noise complicates edge-detection & gradients. Particularly, gradients of noisy images tend to be a bit useless for picking out edges.
- But, we may combine smoothing & gradient in one kernel, thanks to how convolution behaves!
- General pipeline:
  1. Smoothing, suppress noise
  2. Compute gradient
  3. Apply edge 'enhancement'
  4. Edge localization, edges vs noise
  5. Threshold, thin (sometimes)

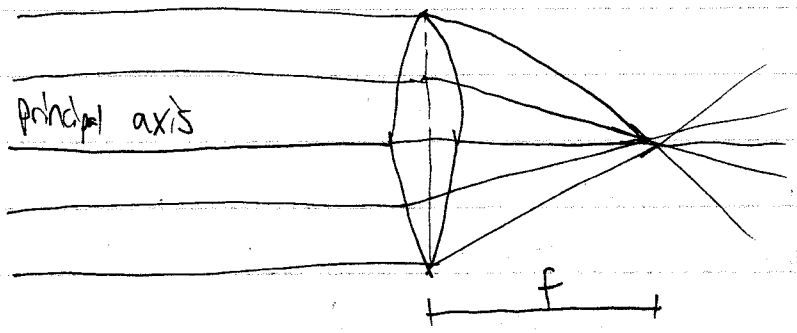
- Canny Edge Detector:
  - Filter image w/ derivative of Gaussian.
  - Find magnitude & orientation of gradient
  - Non-maximum suppression
    - Thin multi-pixel wide 'ridges' down to single-pixel width
  - Linking & thresholding.
    - Two thresholds, low & high. High threshold <u>starts</u> edge, low thresholds <u>continue</u> them.

2016-02-01, M

03-01

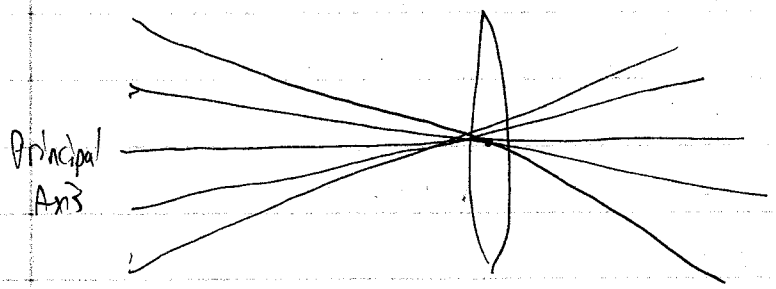
- Cameras
  - We form a scene via a 2D array of pixels
  - Rays are fundamental primitive, & illumination follows a path from source to image - thus we may use geometry on them.

03-01 (cont) -Geometrical optics... my nemesis.



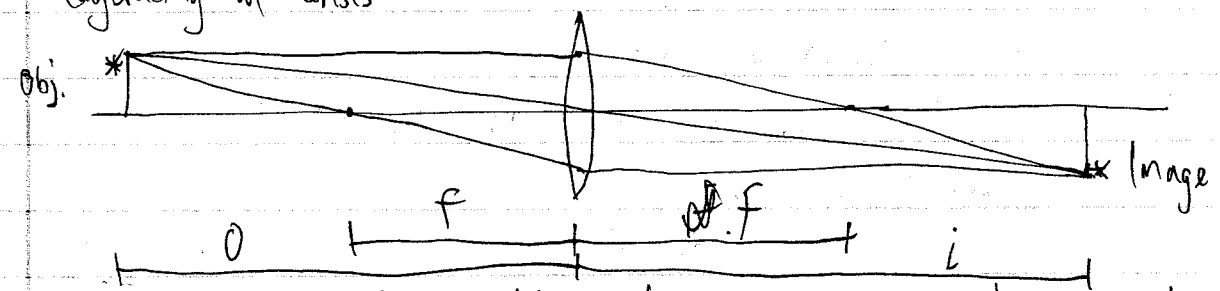
↱ focal length

Parallel rays to principal axis converge at f from lens



-Rays through center of lens don't deviate (acts like pinhole here)

-Raytracing w/ lenses



-Rays from points parallel to lens, forms on plane parallel to lens on other side & upside down

$1/o + 1/i = 1/f$ - lens equation

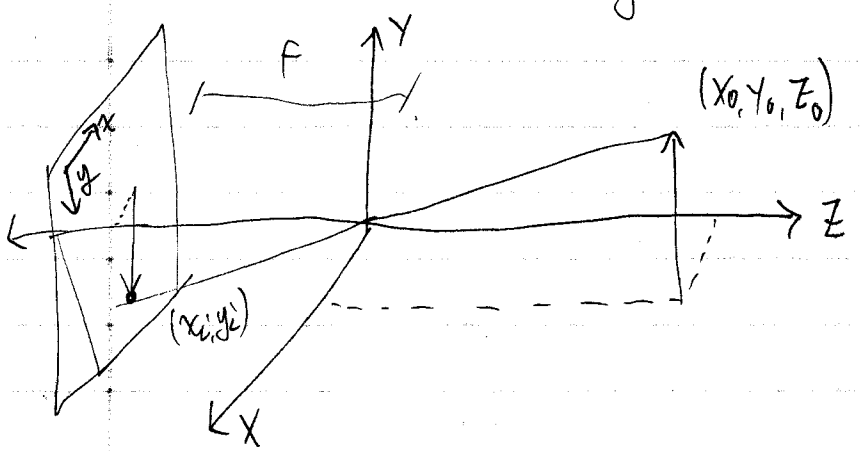
03-02

2016-02-01(b)

- Lenses
  - We are often putting 'image plane' at f away from lns.
  - Larger focal length → larger image formed
- Focusing
  - Moving image plane in/out will make for images that don't have rays converging as well, thus blurrier
  - So, typically focusing consists just in moving image plane (or lenses relative to it).
- FOV:
  - For sensor size h, $\theta = 2 \arctan\left(\frac{h}{2f}\right)$
- Sensor size plays a role too...
  - Full frame: 35 x 24 mm

1:1.26	28.7 x 19.1 mm	
1:1.5	23.7 x 15.5 mm	
1:2	17.3 x 13 mm	(micro 4/3)
	4.54 x 3.52 mm	(iphone 5)

- Changing focal length lets us move back & still capture scene
- Changing viewpoint causes <u>perspective changes</u>
- See Hitchcock's 'vertigo' effect



$x_i/f = X_0/Z_0$

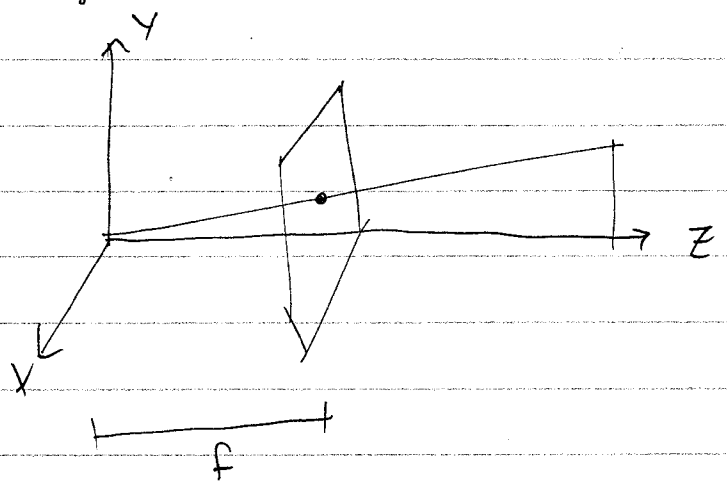
$y_i/f = Y_0/Z_0$

$\rightarrow x_i = \frac{X_0}{Z_0} f$

$y_i = \frac{Y_0}{Z_0} f$

- Typ. 85mm is used for portraits

- But, we can mirror our sensor across origin and image is then not reversed



03-03 - Exposure

- 'Exposure triangle' - shutter speed, aperture, ISO speed
- Exposure = Irradiance x Time → T, time shutter is open

  $H = E \times T$ → E, amount of light falling on a unit area of sensor per second
  - Lens <u>aperture</u> controls this
- In an SLR mirror swings up, then shutter exposes sensor.
- Irradiance is proportional to <u>area</u> of aperture.

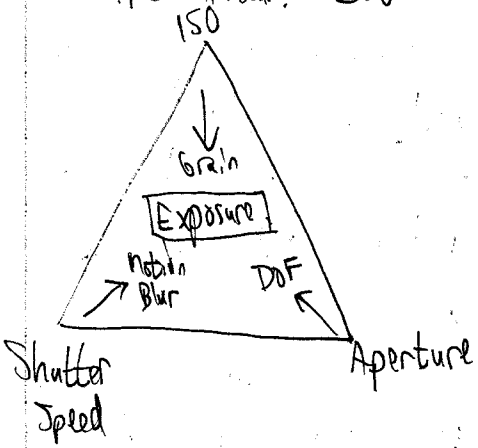
  This area = $\pi \left(\frac{f}{2N}\right)^2$ where N equals aperture number.

  - Usually we see aperture number as f/N so that it gives irradiance regardless of lens focal length.
    - f/2 on 50mm lens → 25 mm aperture
    - f/2 on 200mm lens → 100 mm aperture.
- Doubling N → Reduces A by 2 → Reduces light by 4x
- e.g. f/2.8 to f/5.6 is a factor of 4 <u>decrease</u>
  - Cutting light by factor of 2 → Reduce N by $\sqrt{2}$

2016-02-01c

03-03 (cont)

- Aperture controls depth of field
  - Higher aperture number → smaller aperture → More in focus
- Slower shutter speed → Motion blur
- ISO speed
  - Film: Sensitivity vs. grain
  - Digital: Sensitivity (of sensor itself) vs. noise
  - It's linear. 200 ISO needs half the light of 100 ISO.



- Irradiance, E:
  - Lowering aperture by 1/f doubles H
  - Lowering by 2/f doubles DOF
  - Double T → Double H
    ↘ Double blur

03-04

- Sensor
  - Film vs. Digital: Instructor doesn't consider film & digital cameras different in any essential way
  - The difference is in how light is trapped & preserved, not how it's ~~preserved~~ transported. Film does it chemically, digital does it electronically.
  - Color negative film has multiple layers. Starting from the back (opposite of what light hits first): Film base, adhesive layer, red light layer, green light layer, yellow filter, blue light layer, UV filter, protective layer.
  - In essence, film's a sheet of plastic coated with an emulsion containing light-sensitive silver halide salts (bonded in gelatin)

- Variable crystal sizes determine sensitivity, contrast & resolution
- Digital:

Smith & Boyle, 1969

  - CCD = Charge-Coupled Device, converts electrical charge to digital value.
  - Pixels are represented by capacitors which convert & store incoming photons as electron charges.
  - Layers of CCD: (starting where light hits first)
    - Micro lenses: Capture light & direct toward light-sensitive areas
    - Hot mirror: Lets visible light pass, reflects back invisible light (or, lets in-spectrum light pass, if other types of light like UV or IR should be captured)
    - Color Filter: Since photodiodes are color-blind, color filter matrix separates red, green, and blue, i.e. a Bayer Array (typically)

R	G
G	B

    - Photodiodes: Where light energy converts to electrons
    - Well (Depletion layer): Where electrons are collected
      - Processor normally charges photodiode w/ positive charge
- Raw input from Bayer array must be converted. We have a mosaic of interleaved R, G, B, and need full planes of R, G, B. → 'Demosaicing'
- CMOS - Complementary-Metal-Oxide-Semiconductor - is more common than CCD now (and is a bit cheaper).
  - Photosites in CCD are 'passive' and do no work. They're copied row-by-row and amplified.
  - Photosites in CMOS are amplifiers, and some local processing is possible too.

2016-02-01d

03-04 (cont)

- CMOS sensors have 'rolling shutter' artifacts, particularly on lower end, because of how readout is done.
- Camera RAW contains minimally-processed data from sensor.
  - Captures 'radiometric' characteristics of the scene, rather than manipulating (e.g. whitebalance)
  - Like a negative, has wider dynamic range & preserves most of the information of captured image.

-Device-dependent colorspace

2016-02-07

- Notes on: Feature-Based Image Metamorphosis (Beier & Neely, '92)
  - Also look up: Quantel Mirage
  - Wolberg's book talks about mesh warping with 2D spline mapping.
  - 3.2 gives simple example: Warping from source to destination image by a line segment defined in each image
  - 3.3 extends this to multiple line pairs (and each pixel having a weight that peaks when it is directly on the line, and falls off outside of this)
  - Authors say this is much more expressive than what Wolberg's book describes. (Meshes & splines are very unruly, it sounds like, and lines are much more intuitive for animators).
  - This generalizes to morphing two animations too - lines just need to be keyframed & interpolated

A directed line segment, at that.

Notes on:

- Computational Photography: Epsilon to Coded Photography (Raskar)
  - Coded photography: encoding of photographic signal & post-capture decoding for improved scene analysis.
  - Contrast with 'film-like photography' in which recorded image is 2D projection of scene.
  - Coded photography tries to embed richer representation of scene during encoding.
  - Computational photography reduces to epsilon photography, where multiple images record scene with epsilon variation of camera parameters.

  → Coded computational photography: intermediate images may not be useful to human observer.

- 4 examples: Coded exposure
  Coded aperture optical heterodyning
  Coded illumination
  Coded sensing
- Light Field: describes amount of light traveling in every direction through every point in space. (where radiance measures light amount along ray)
- Plenoptic function: radiance along all rays in a 3D region of space
  - expresses scene from all viewing positions & angles.
  - since it's a function of (x, y, z, θ, ϕ), it's five-dimensional.
- But, if nothing blocks light, radiance along ray is constant. We can measure plenoptic function on the convex hull of an object easily with a digital camera, and it has redundant info even.

  → Radiance on a ray is constant along the ray's length → 1 dimension less

  → 4D function, the 'photic field' or 'light field' or 'Lumigraph'.
  - Formally: "radiance along rays in empty space"

2016-02-07(6), CS6475

Raskar (2009) cont.

- Often uses "two-plane parametrization", though this can't represent everything. ("Light slab" refers to that form.)
- 4D Reflectance Field: BRDF = bidirectional reflectance distribution function, defines ratio of reflected radiance (given incoming & outgoing direction)
- Current digital photography is mostly "electronically-implemented film photography", optimized around old limitations of chemistry & mechanics.
- Comp. photography generalizes:
  - Generalized optics: Optical elements are 4D ray-benders modifying the light field.
  - Generalized sensors: 'Normal' sensors capture only a 2D projection of a 4D light field. CP tries to extend this to 3D or 4D ray representations.
  - Generalized reconstruction:
  - Computational illumination:
- A scene is an '8D ray modulator'
- 'Fluttered shutter': Uses carefully chosen binary sequence to open & close shutter, and encode motion in reversible way (model here uses ferro-electric LCD rather than mechanical)
- Coded aperture
  - Patterned attenuating mask near aperture → full-res digital refocusing
  - Same near sensor: recover 4D light field, no lenslet array needed

'wrap-around views', multiple center-of-projection

spatial optical heterodyning

- Can be done w/ lenslet arrays, or patterned masks
- Multi-flash cameras can locate depth discontinuities
- "gradient sensing camera" - senses differences in neighboring pixels rather than actual intensities.
  - Even for very high-dynamic range scenes, these differences remain small. Thermal & quantization noise are in gradient domain - so they produce low-frequency 'cloudy' noise rather than uncorrelated high-frequency noise.

- Notes on: "A Multiresolution Spline With Application to Image Mosaics" (Burt & Adelson)
  - The problem: How can two surfaces (as in images) be gently distorted so that they can be joined together with a smooth seam?
  - This relates to the spatial frequency content of the images.
  - The 'transition zone' must be comparable in size to the wavelength of the lowest prominent feature in the image.
    - Too small: Obvious edge. Too large: 'double exposure' effect.
  - Bandwidth of images to be splined should be roughly one octave.
  - What if it's not?
    - Then decompose into bandpass component images, spline w/ proper T, & recombine.

Multi-resolution Spline

pyramid

  - This then ties into the Laplacian described in other papers...

2016-02-07c, CS6475

- Notes on: "Image Quilting for Texture Synthesis & Transfer" (Efros & Freeman)
  - Describes: How to ~~[illegible]~~ synthesize textures from existing ones, in a way that looks seamless
  - Also: How to transfer a given texture to a different source image.
- Notes on: "Graphcut Textures: Image & Video Synthesis Using Graph Cuts" (Kwatra, Schödl, Essa, Turk, Bobick)
  - Extends the above to further dimensions and also doesn't require patch size to be chosen in advance
  - MRF, Markov Random Field
  - http://www.cc.gatech.edu/cpl/projects/graphcuttextures/

Yes, as in Irfan Essa and Aaron Bobick.

04-01
04-02
04-03

- More Fourier...
- More blending
- Gaussian pyramids...
- Pyramid blending
  - We may use pyramids to cross-fade at every frequency band
  - We may also blend with some mask also
    - Need Laplacian pyramids for source images and mask
    - Use Gaussians of Mask as weights at each level to form new Laplacians

04-04

- Blends vs. Cuts
  - Cuts are needed more when combining images into mosaics, perhaps where blending would allow through details of both when only one is preferred (e.g. a panoramic, or shots with a moving person). - moving objects cause 'ghosting' with blending.
  - Aim is to find optimal seam.
  - One method: Compute squared error between two images. Look for minimum path through this error image → seams
  - See instructor's paper here...
  - Images can be extended this way (incl. with perspective)
  - Uses 'graph cuts', representing pictures w/ node structures on pixels, & cost function between nodes (related to pixel similarity).
  - Look up Boykov & Jolly (2001) - Max-Flow/min-cut algorithms
    - Dynamic programming is also used
  - Another method: Seam Carving (Grundmann et al. 2010, Avidan & Shamir 2007)
    - Can change aspect ratio? Relates to content-aware fill

See final slide for list of all these papers!

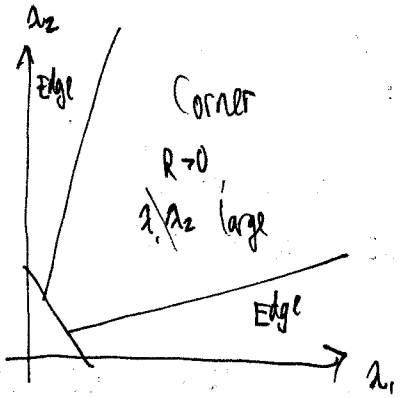
04-05

- Features
  - A lot of this is redundant with CS6476 4A-L1 (see 2015-09-20 notes), 4A-L2
  - We usually look for corners, which are repeatable & distinctive.
  - Edges are ambiguous (one can move at right angle to gradient).

2016-02-10, CS6475

04-05 (cont.)

- For $R = \det(M) - \alpha \, \text{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$
  - α: 0.04 to 0.06



Depends only on M eigenvalues.

- We might use Harris corners in a scale-invariant way with Harris-Laplacian (look this up)
  - → Local max of
    - Harris corner in space
    - Laplacian in scale
- SIFT
  - Find local maxima of...
    - DoG in space & scale
    - DoG is simply pyramid of difference of Gaussians within each octave

04-06

- Feature Detection & Matching
- M is the second moment matrix, but what's that mean?
  - 'Image moment' is a particularly weighted average of pixel intensities, also a function thereof, to give some property at a point.
  - 2nd image moment is derived from gradient
    - Summarizes direction & magnitude at point, and coherency of directions there.

step-by-step:

1. Compute horizontal & vertical derivatives (convolve w/ derivative of Gaussians)
2. Compute outer products of gradients M
3. Convolve with large Gaussian (some smoothing is needed)
4. Compute scalar interest measure R
5. Find local maxima over threshold, detect features!

- Harris is invariant to intensity shift but less so to intensity scale.
- It's not in general scale-invariant (spatial scale that is)
- A 'good' function for detecting scale needs to have one stable, sharp peak, & respond to contrast.
- SIFT
  - refer back to CS6476, 4A-L3\4B-L1, 2015-09-20\21

2016-02-29

05-01

- Image Transformations
  - Rigid, affine, projective transformations
  - Image filtering: Change range of image, $g(x) = T(f(x))$
  - Image warping: Change domain, $g(x) = f(T(x))$
- Parametric global warping: Translation, rotation, scaling, aspect (changing just one scale), perspective, shearing
- Simple 2D transforms: $p' = Mp$, $\begin{bmatrix} x' \\ y' \end{bmatrix} = M \begin{bmatrix} x \\ y \end{bmatrix}$
- Scaling: $\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$, $a = b$ for uniform scaling
- Mirroring over (0,0): $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
- Shear: $\begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix}$

2016-02-29(b), CS6475

05-01 (cont.)

- For rotation: $x' = x\cos\theta - y\sin\theta$, $y' = x\sin\theta + y\cos\theta$ $\rightarrow M = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
- Note here that all of these are just linear combinations
  - Origin always maps to origin! & Lines always map to lines.
  - Parallel lines stay parallel.
- But for translation: $x' = x + t_x$, $y' = y + t_y$.
  - No matrix exists for this if we have $\begin{bmatrix} x' \\ y' \end{bmatrix} = M \begin{bmatrix} x \\ y \end{bmatrix}$
  - We must use homogeneous coordinates for this.

$(x, y, w) \rightarrow (x/w, y/w)$, $(x, y, 0) \rightarrow \infty$

$(0, 0, 0)$ is just not allowed.

$$\rightarrow M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = M \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{for translation.}$$

Since we want $w = 1$, this is 8 DOF, not 9

  - For other transforms we did already, just put the 2x2 matrix at the upper 2x2 of this.
- Helpfully, we can combine transformations by multiplying matrices.
- Affine translations: Linear transformations & translations
  - Lines stay lines, parallel stays parallel, ratios of distance stay the same - but origin may change

6 DOF

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

- Projective transformation: Affine transformations & projective warps
  - Lines stay lines but parallel might not stay parallel, ratios aren't preserved.

8 DOF

- So, how do we figure out the transformations between two images?

↳ Preserves lengths

05-01 (cont.)
- Euclidean transform = translation & rotation, 3 DOF
- Similarity transform = Euclidean & scale, 4 DOF
  ↳ Preserves angles
  cv2.getRotationMatrix, cv2.getAffineTransformation
- See: cv2.warpAffine
  cv2.resize
  cv2.getPerspectiveTransform
- Forward warping: send pixel $f(x,y)$ to corresponding location $(x',y') = T(x,y)$ in 2nd image.
  - But what if pixel lands 'between' two pixels?
    - Distribute colour w/ neighboring pixels
- Inverse warping: Get pixel $g(x',y')$ from its corresponding location, $(x,y) = T^{-1}(x',y')$ in first image
  - If it comes from 'between' two pixels, interpolate from neighbors in source image
- Generally we want inverse warping; it eliminates holes. However, it requires an invertible warp which isn't always possible.
- See Szeliski, ch. 2

05-02
- Transformations: Lines → Lines
- Warping: Points → Points (→ Input, → Output)
- Forward: S & T image, $S(u,v) \rightarrow T(x,y)$.
  $(x,y) = [X(u,v), Y(u,v)]$ - generate new coords in T
- Inverse: $(u,v) = [U(x,y), V(x,y)]$ - generate coords in S that (x,y) came from

2016-08-29c, CS6475

05-02 (cont.)

- Forward warping can have 'holes' or have magnification issues → missing info at some x,y
- Inverse warping can have 'minification' where a pixel in (x,y) maps to < 1 pixel in (u,v)
- In both cases a pixel can map to much more than one pixel, and cause aliasing or other distortion.

See: 'two-pass transforms'

- One may have to subdivide pixels to handle this
- Texture mapping has to solve many of these problems
- Forward warping uses "splatting" often for when (u,v) → (x,y) between two pixels
- For inverse warping, we interpolate between other (u,v) neighbors.
- Mesh-based warping can avoid many of the minification/magnification issues.
  - Use sparse set of corresponding points & interpolate with a displacement field
  - Triangulate set of points on source
  - Use affine model for each triangle
  - Triangulate target with displaced points
  - Use inverse mapping

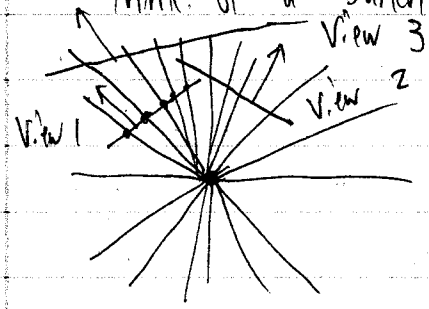
See Szeliski chapter 3 & Beier and Neely (1992)

- Image Morphing - Widely used in movies
  - Book to check out: "On Growth & Form" (Thompson)
  - Often uses tri/quad mesh, or looks for corresponding feature meshes, or corresponding oriented line segments
-

Basic steps:
- Capture Images
- Detection & Matching
- Warping → Aligning Images
- Blending, Cutting, Fading
- Optional Cropping

05-03

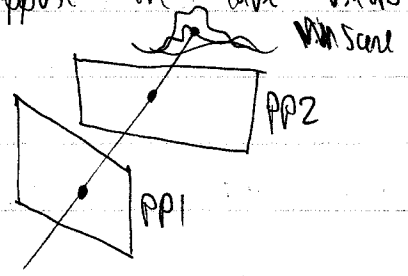
- Panoramas
  - Simple case: Left & right images
    - Generally we need to warp both images
- "Bundle of rays" - Contains all views
  - Think of a bunch of concentric rays of light



If we have views 1 & 2 in "real" images, can we synthesize view 3? (Yes, if center of projection is the same!)

→ Image re-projection

- Suppose we have views PP1 & PP2



- Cast ray through each pixel in PP1
- Draw pixel where that intersects PP2
- This is just a 2D warp
- Also, it's why we have feature detection.
- Note that we dont need 3D information from this!

See CS4476 3D-L2, L1

- This 2D warp is a homography
  - Rectangle maps to arbitrary quadrilateral (straight lines stay straight, parallel lines aren't preserved)
  - For $p' = Hp$, $p' = \begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}$, $p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ - need at least 8 equations
  - Ideally, get > 8 constraints & minimize
- Use this to warp into shared coordinate space.

2016-02-29 d, CS6475

05-03 (cont.)

- Feature detection still may produce bad matches (outliers)
  - RANSAC is good at dealing with these (see CS6476, 4C-L2)
    1. Select 4 feature pairs at random
    2. Compute homography H
    3. Compute inliers where $SSD(p'_{i,n}, H \cdot p_{i,n}) < \epsilon$
    4. Keep largest set of inliers
    5. Re-compute least-squares H estimate from that set

See cv2.ORB, cv2.SIFT and keypoints

- Kolor autopano giga v3 — Dr. Essa uses these
- We projected onto a plane. We can also project onto things like cylinders or spheres, and these have their own distortion properties (eg. cylinders curve 'horizontal' lines, spheres curve all lines)

ICCV 2003

- See paper: Brown and Lowe (2003) - "Recognising Panoramas"
- Microsoft Research, Image Composite Editor (ICE)
- PTGui, Hugin

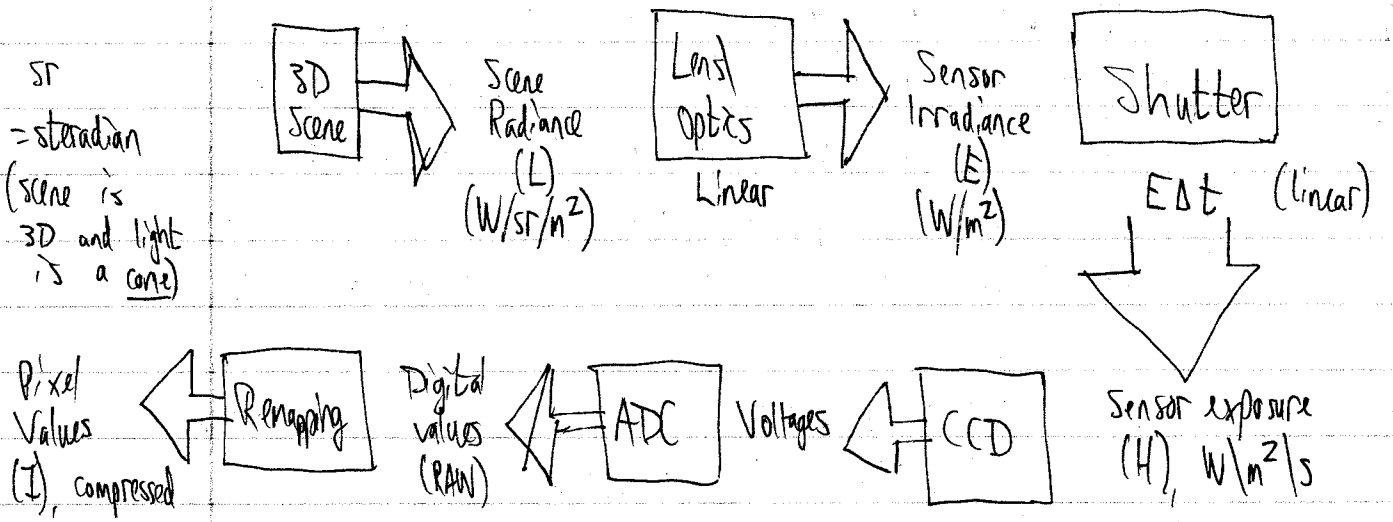
2016-03-01

05-04

- High Dynamic Range
- Luminance: Photometric measure of luminous intensity per unit area of light traveling in a given direction.
  Measured in candela per sq. meter (cd/m^2).
  - 10^{-6} cd/m^2 is starlight (up to $\sim 10^{-3}$)
  - 10^{-1} cd/m^2 is moonlight
  - 10^{2} cd/m^2 is indoor lighting
  - 10^{5} cd/m^2 is sunlight

↳ Unchanging scene

- Human static contrast ratio is ~ 100:1 (~6.5 f-stops)
- " dynamic " " " ~ 10^6:1 (~20 f-stops)
  ↳ Changing lighting conditions
- Goal in HDR is to capture this wider range.
- Need ~ 5-10 $\times 10^6$ values for 'normal' brightnesses but an 8-bit image has only 256 values.
- Image acquisition:



sr = steradian (scene is 3D and light is a cone)

These are all non-linear (top part is linear)

$g: L \to E \to H \to I \iff g^{-1}: I \to H \to E \to L$

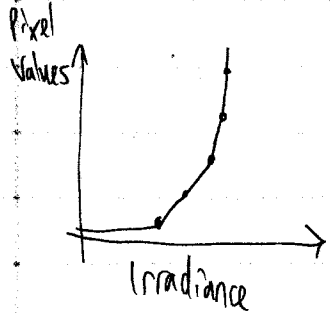
- Can we find this inverse?

- Camera calibration: - Geometric (how pixel coords relate to world)
  - Radiometric/photometric (how pixel values relate to radiance in world)
- We might not have good measurements on this 'world', but we can relate images to other images (as we did in CS6476).

2016-03-01(b), CS6475

05-04 (cont.)

- One way of radiometric calibration: Multiple exposures of a color chart, aiming to reach all areas of curve



- Assumes constant, faraway light source
- Since g is monotonic, an inverse exists

See: Debevec & Malik (1997)

- Pixel values $(I) = g(\text{Exposure})$
  Exposure $(H) = \text{Irradiance}\ (E) * \Delta t$
  $\log H = \log E + \log \Delta t$
- We may plot 'response curves' for each pixel type, assuming unit radiance for each pixel
- For each pixel site i in each image j, compute:
  $$\ln(E_i) + \ln(\Delta t_j) = g(Z_{ij})$$
- Solve over-determined linear system for N pixels & P different exposure images:

$$\underbrace{\sum_{i=1}^{N}\sum_{j=1}^{P}\left[\ln(E_i) + \ln(\Delta t_j) - g(Z_{ij})\right]^2}_{\text{Fitting term}} + \underbrace{\lambda \sum_{z=Z_{min}}^{Z_{max}} g''(z)^2}_{\text{Smoothness term (?)}}$$

- From this we can get a radiance map.

Ward (2001)

- May need a new image format for this, e.g. Radiance Format (8 bits each for R, G, B, and exponent n → factor 2^{n-128})

so do printers & projectors

- But, displays have similar limitations as cameras, so how do we make the HDR image displayable?

See: Banterle, et al. (2011)
Reinhard et al. (2002)
Durand & Dorsey (2002)

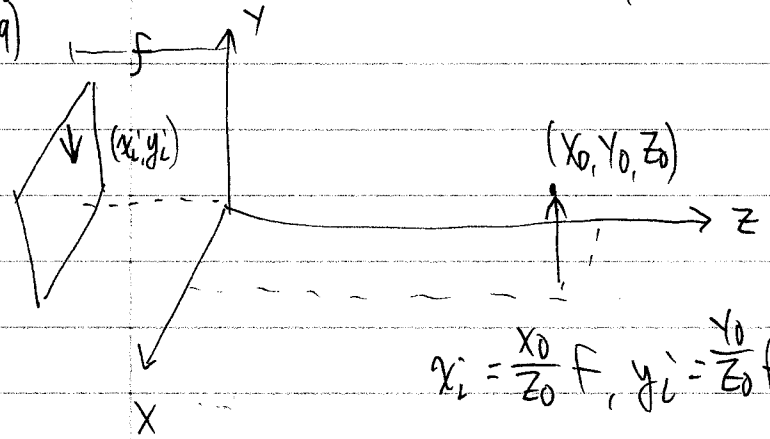
- So: Tone mapping, mapping colors to display on a medium with limited dynamic range
  - Reduces contrast strongly
  - Preserves image details & color appearance
  - A lot of commonly-used tonemapping just looks unnatural - beware.
  - It does lose information (just as a single low-dynamic-range image does). But, we keep the real radiance values too and can further process them.

See: Grossberg & Nayar (2013)

05-05

- Stereo
  - What can we do with stereo images (e.g. from our eyes)?
  - We're interested in capturing 3D scene w/ geometry
  - See CS4476 3B-L1 (2015-09-05)

See: McMillan & Gortler (1999)



$x_i = \frac{X_0}{Z_0} f, \quad y_i = \frac{Y_0}{Z_0} f$ → Any points on same ray map to same point in image

Since $\frac{X_0}{Z_0} = \frac{kX_0}{kZ_0}$ for instance

- While there is depth ambiguity, things like vanishing lines/points give us depth cues
  - And objects of known sizes, object occlusions, shading, structured light, texture, defocus, structure from motion (see Fredrikkson & Olsson, 2012)

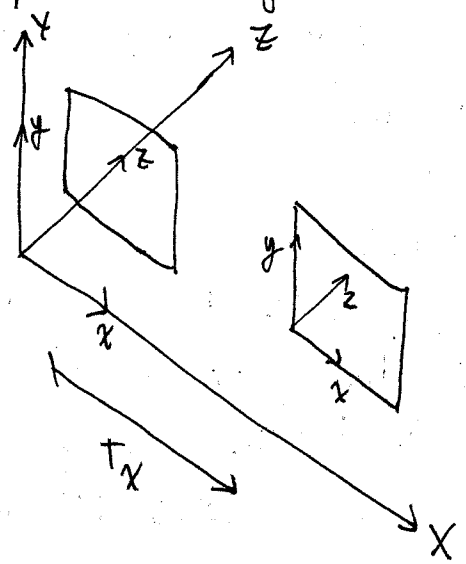
these are just from one view

See: Trimensional, Photometric Stereo

2016-03-01c; CS6475

05-05 (cont.)

- Stereo vision provides for things like triangulation to resolve that ambiguity
- Our eyes are ~60mm apart.
- Parallax: Points at different depths displace differently
  - Nearby points displace more than far ones
- 'Anaglyph' encodes parallax in one picture (needing filters to separate out the two views) - typ. red & cyan filters, so left eye sees left image, right eye sees right, & brain performs normal fusion.
- Simple stereo system:



Left camera at $(0,0,0)$
Right camera at $(T_x,0,0)$
- Just translated right by T_x.
  (T_x = stereo 'baseline')

- From x_l & x_r in view below can we recover depth?

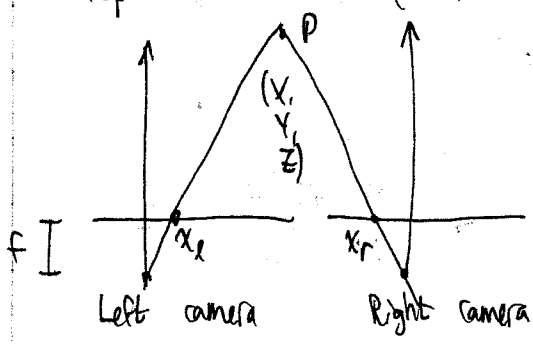
$$x_l = f\frac{X}{Z},\ y_l = f\frac{Y}{Z}$$

$$x_r = f\frac{X-T_x}{Z},\ y_r = f\frac{Y}{Z}$$

Let disparity $d = x_l - x_r$

$$d = f\frac{T_x}{Z},\quad Z = f\frac{T_x}{d}$$

- Top-down view: (XZ)



See CS647b 3B-L2

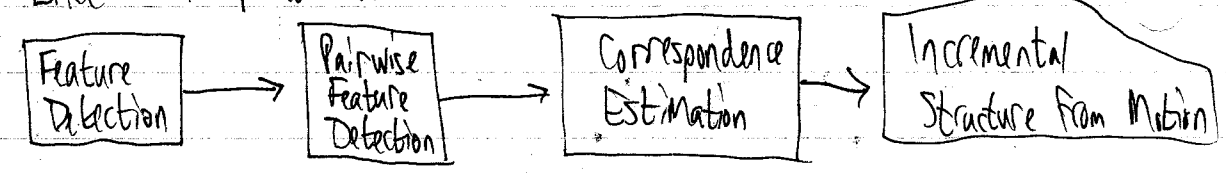
- To compute disparity we need to find point correspondences
  - Epipolar constraint helps with this by restricting search to one line
- Occlusions create regions where no corresponding points exist. Larger patches can smooth results but make these regions & finer regions worse.
- See: Google Project Tango, Amazon Fire w/ 3 Cameras
- Dr. Essa has the Fujifilm camera with two lenses

05-06

- Photosynth
  - Going beyond panoramas', 'photo tourism'

UW & MS Research

  - See: Snavely, Seitz, Szeliski, "Photo tourism: Exploring photo collections in 3D," ACM SIGGRAPH 2006
  - photosynth.net technology preview
    http://phototour.cs.washington.edu
  - Uses iterative bundle adjustment
  - Users can annotate regions of photos and these are automatically translated to other photos of the same areas (properly accounting for different viewpoints)
  - This involves some amount of scene reconstruction & SFM
- Scene reconstruction:
  - Automatically estimate position, orientation, & focal length of camera
  - 3D positions of feature points
  - Like with panorama:



2016-03-01, CS6475

05-06 (cont.)

- Feature detection - might use SIFT
- Pairwise feature matching - match features between each pair of images (yes, this can be a lot of pairs). The Photosynth paper provided Bundler to do this efficiently. Might also use RANSAC to refine.
- Correspondence estimation - link up pairwise matches across several images
- Structure From Motion - can we determine translation & rotation of different views of same scene?
  - The paper's innovation was to do this incrementally, not globally (for every pair), as each 'new' image arrived
- To do: Go to https://photosynth.net and make an account, and make your own
  - Homework refers to this?
- Google Maps uses much the same technique with Street View and the like
- This is a pretty brilliant use of photos to show space & environments, so it sees use in architecture & real estate

See: Kushal, Self, Furukawa, Gallup, Hernandez, Curless, Seitz (2012), "Photo Tours", 3DPTV 2012

2016-03-17

06-01

- Video processing
  - Images over time - $I(x,y,t)$ (continuous) or $I(i,j,t)$ (discrete)
  - File formats: include images, framerates, codec/wrappers

- We don't cover video compression here.
- Persistence of Vision (supposedly) is what makes a video appear as motion to us.
- See Muybridge (1830-1904); he used stop-motion
- Marey (1830-1904) - similar work
- We can see our data now as a 3D 'volume' in x, y, & t
  - But much of this works the same
  - Feature detection is very useful here (features in one frame often are visible in next)
- Direct tracking approaches: Find feature, match it to feature in next frame
- Motion-based approaches: compute motion at pixel (each pixel) between frames - optical flow

See his website:
- Some of Dr. Essa's work: Registering & blending multiple distinct video views of same scene

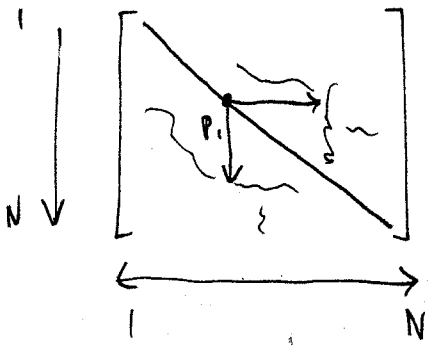
06-02

- Video textures
  - We can detect repeating textures in images & synthesize them. This same concept exists for video too.
  - Simple form: Looping a video that has some periodic motion
    - This can have gaps/jumps though.
  - One approach: Find frames in video that are similar to each other (think of an NxN matrix giving similarity between every frame 1, 2, ... N)
    - What metric for similarity? We can use L2 norm, or Manhattan distance (L1 norm). Both work well.

2016-03-17 (6), CS6475.

06-02 (cont.)

- We can visualize this matrix, too (see video #9)
  - We expect the diagonal to be similar, and nearby the diagonal too. But, others will be similar too.
  - From this we can construct loops:



e.g. play up to frame at p_i, and then jump across row or column to any frame that is similar enough (then jump back to diagonal)

  - Can play forwards or backwards too.
  - We get a sort of state machine, almost. We may loop infinitely; we may choose different rates of transition. Or: A <u>markov chain</u>.
- What about if we have a pendulum, in which it has two near-identical frames but in which pendulum moves left in one & right in the other?
  - Need to model its dynamics somehow or the transitions won't look right.
- Also sometimes need to fade/blend/morph (graphcut textures also solve this).
- See Kwatra, Schödl, Essa (2003) - cuts can generate much cleaner transitions while still having crisp images.
- Video portraits - One can apply this to keep a person animated indefinitely with slight variation in expression

"Controlled Animation of Video Sprites"

- Video Sprites (Schödl & Essa, 2002)
- Cliplets, Cinemagrams (Microsoft Research)
  - Selecting & animating only parts of a video
  - See Youtube links at 2nd to last video

Henrik Sozzi, Tom Freestone

"Panoramic video textures" (SIGGRAPH 2005)
"Selectively De-animating Video" (SIGGRAPH 2012)

2016-03-22

06-03

- Video Stabilization
  - GATech built their own system for this
  - Shaky videos are very common on Youtube
  - See: Grundmann, Kwatra, and Essa (2011) IEEE CVPR 2011
    Grundmann, Kwatra, Castro, and Essa (2012), IEEE ICCP 2012
  - This work is now on Youtube, available when you upload video.
- Older kinds: Steadicam from Garrett Brown (1975), done for Star Wars
  - It doesn't use gyroscopes, but it really just uses counterweights & balance to isolate the camera from body motion
  - Optical/in-camera stabilization (moving lens or sensor to cancel motion) - these get rid of high-frequency perturbations but don't do much for low-frequency
- Post-process stabilization has many benefits - it can handle video from any camera, it can handle low-frequency shake, it can be done with much larger/cheaper distributed processing power

2016-03-22(b), CS6475

06-03 (cont.)

- Main steps:
  - Estimate motion
  - Stabilize camera path
  - Crop & re-synthesize
- The camera person usually has a fairly wide-angle lens, and tries to center the subject - so this helps the process by giving room to crop to a smaller 'virtual' camera.
  - But we must be careful not to let that 'virtual' camera move to outside of the real one giving us borders with no data. (And we don't want inpainting/faking).
  - So we must constrain to the frame.
- Estimating camera motion can use many of our prior techniques - detecting & tracking corners, for instance. We must be able to tell what is foreground & what is background, and focus on the background.
- Motion model:
  - Translation (2 DOF) has still some wobble
  - Similarity (4 DOF) does better, but still has wobble
  - Homography (8 DOF) handles things best
- Next: What are the properties of a <u>stable</u> path, if we want to generate one?
  - May try to turn path to partitions of constant, linear, or parabolic.

- Our crop window defines the "allowed" paths. Smaller crop windows allow more motion.
- Recall CCD & CMOS differences:
  - In CCD: Photosites read in to amplifier at end; global shutter.
  - In CMOS: Photosites are amplifiers; read one scanline at a time.
    - They have rolling shutters, and this causes non-rigid deformation when motion occurs.
- Stabilization needs to handle this too in many cases.
  - The problem is that readout speed varies per-camera
  - Their solution split the image into parts and attempted to estimate it.
- One thing added was adaptive crop sizes, and a fallback mode in case stabilization fails
- Alternatives:
  - Chickens have the ability to keep their heads very still through body motion... Yes, someone tried to make use of this.

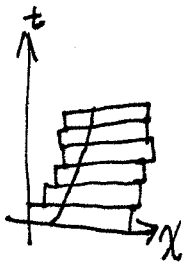
2016-03-23

06-04
- Panoramic Video Textures
  - This combines what we covered already with panoramas, and with video textures
  - PVT: "A video that has been stitched into a single, wide FOV," but the dynamic parts of that video can continue to play indefinitely

2016-03-23(b)

06-04 (cnt.)

- Difficulty: Input captures the dynamics only at one region
- Can either use all video frames, or select N images, to turn video to panorama
- 'Video volume' stacks frames in time, but we may turn this to a global coordinate space
  - We may use what we've discussed elsewhere - RANSAC and such
  - But we must also identify the regions that are dynamic
- See: http://grail.cs.washington.edu/projects/panovidtex
- Map a continuous diagonal slice of the "input video volume" to output panorama
  - Problem: Restricts boundaries to frame, has shearing artifacts



- They addressed this with... cuts.
  - The usual graph cuts & seams. (Agarwala et al., 2005)
  - mincut algorithm.
- Another step to "active photographs" & more immersive imagery; how to store & display this efficiently remains an issue
- See: - Agarwala, Zheng, Pal, Agrawala, Cohen, Curless, Salesin, Szeliski (2005) "Panoramic Video Textures", SIGGRAPH 2005
- Schödl, Szeliski, Salesin, & Essa (2000), "Video textures", SIGGRAPH 2000
- Kwatra, Schödl, Essa, Turk, Bobick (2003), "Graphcut textures: image & video synthesis using graph cuts", SIGGRAPH 2003

2016-04-04

07-01

- Light Field

- Recall that photography is about capturing rays of light. Our analysis thus far has traced these rays through a scene back to pixels.
- But pixels can be very limiting. We can only get so far with processing pixels after capture.
- Recall pinhole cameras.
  We have that single point with a "bundle of light rays" being captured.
- But this point isn't special. Any point at all "sees" all these bundles of rays.
- We can think of each point as a sphere, light accumulating at the center.
- We simply don't have sensors at those points.
- Say P is the intensity distribution at that point. We can parametrize that sphere with θ & ϕ, so $P(\theta, \phi)$. If we're really far away, it's basically $P(x, y)$.
- But light has colour & changes over time. So, we need wavelength λ & time t.
  → $P(\theta, \phi, \lambda, t)$ and $P(x, y, \lambda, t)$
- Now say V_x, V_y, V_z is position of the viewing point (where we see point from).

2016-04-04(6), CS6475

07-01 (cont.)

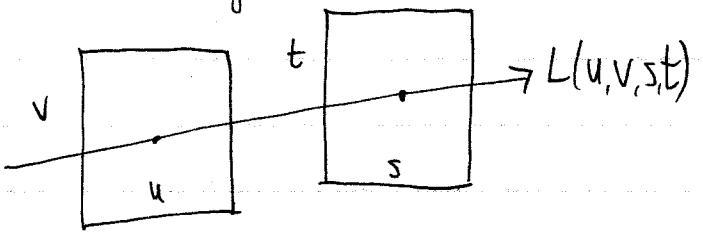
→ "The Plenoptic Function, P_f, is measured in an idealized manner by placing an eye at every possible location in the scene (V_x, V_y, V_z) and recording intensity of light rays, wavelength λ, at time t, at every possible angles (θ, ϕ) (around V_z) or in terms of (x, y).

See: Adelson & Bergsen (1991)

- $P(\theta, \phi, \lambda, t, V_x, V_y, V_z)$ or $P(x, y, \lambda, t, V_x, V_y, V_z)$
- A plenoptic light-field camera: captures a light-field & renders to pixels as needed.
- $P(\theta, \phi, \lambda, t, V_x, V_y, V_z)$ is 7-dimensional; it captures the complete scene, as in holographic video.

5-D

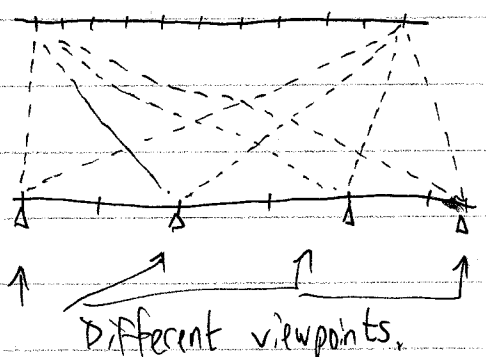
- If we ignore time & color: Capture only viewpoint & direction, $P(\theta, \phi, V_x, V_y, V_z)$ - think static holograms
- But this can also be 4-D if we put some constraints:
  - Put within bounding box (note that space of lines in 2D space is 4D)
  - No occluding objects, but viewpoint & direction
  - Think of a snow globe.
  - "Outside" of the scene, light from the scene does not get occluded by objects - and is represented as a 4D light field. Think of:



Adelson & Wang, 1991

07-01 (cont.)

- This is two-plane parametrization for light slab \ beam.
- Each plane has 2 coordinates. Front is (u,v), back is (t,s). Light flows from uv to st plane.
  - For simplicity, $u,v,s,t \in [0,1]$.
- One way to see:

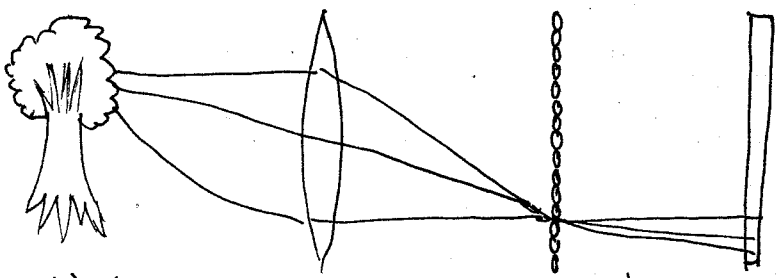


- Rays arriving at one point on the uv plane from all points on st plane.
  - Think of st plane as having various viewpoints of scene?
- Rays leaving at one point in st plane are bound for all points on the uv plane.
  - Think of a whole array of cameras on uv plane
- Simplest form: $P(\theta, \phi)$, 2 dimensions: A panorama over a sphere. View is fixed.
- With two pinholes we may get depth parallax
- With a moving one, motion parallax
- Eccentric aperture sort of encodes depth by offsetting image left\right of optical center.
- A bunch of pinholes after the lens can encode some of this information.
- See: Ng et al. 2005, "A Light-Field\Plenoptic Camera"

2016-04-04c, CS6475

07-01 (cont.)

(That's a tree, not a mushroom cloud.)



- Lenticular arrays may be used in printed cards; cylindrical lenses in a row can act as this.
- Now think of the uv plane at the lens, and st plane at lenticular array
- This is an old area: See 1908, Lippman & integral photography
  1992: Stereo from one lens, Adelson & Wang
  2012: Lytro is commercially available.
- Lytro in some sense does epsilon photography
  - Supplies depth & different focal planes
- Lens w/ array of pinhole cameras can encode both intensity & direction.
- See:
  Gorler, Grzeszczuk, Szeliski, Cohen (1996), "The Lumigraph" (ACM SIGGRAPH 1996)
  Levoy & Hanrahan, (1996) "Light field rendering", SIGGRAPH 1996

2016-04-05

07-02

- Projector-Camera Systems (PROCAMS)
  - Projector can be used as a controlled light source
- Recall pipeline: 3D Scene → Illumination → Optics → Sensor → Processing → Display → User
- A projector has controllable apertures just as a camera does (they're individual pixels here)
- See: Lightstage, USC\ICT
  Trimensional, Schindler
- Lightstage lets us capture from many angles of lighting and then use this in later re-rendering\relight — http://gl.ict.usc.edu/LightStages (Debevec, 2012)
- Trimensional uses the screen of a phone, rather than a projector, to control the lighting, combined with capturing from the front-facing camera.

Both use gray codes on projector

- We could use these techniques also to auto-calibrate a projector to correct keystoning - and to generalize this to N projectors that need to stitch an image
  http://wearables.unisa.edu.au, Marner, Smith, Walsh, Thomas (2014)
  libSAR?
- Lee, Dietz, Aminzade, & Hudson (2004) - projector calibration with 1 pixel sensor for - one at each corner)
  - Also extends to multiple projectors

http://www.cs.cmu.edu/smartheadlight/

- Light that is 'aware' of obstructions: Two projectors & camera, and track obstructions - and use other projector to illuminate where other is blocked.
  - Summet Flagg et al. (2007)
- Programmable car headlights (CMU project)

2016-04-05(b), CS6475

07-02 (cont.)

- RoomAlive (Microsoft Research)
  http://projection-mapping.org/roomalive-uist

07-03

- Coded photography
  - This relates somewhat to epsilon photography: changing parameters (exposure, viewpoint, focus) and fusing
  - It is also similar to light field cameras
  - Coded exposure: Control light in time
  - Coded aperture: Control light near sensor
  - Coded illumination: Control light in the scene
  - Coded sensing: Control intensities (Bayer sensors are already this! We code it so we get R, G, B.)

Raskar, 2009

- "Coded photography encodes the photographic signal & post-capture decoding for improved scene analysis."
  - Epsilon photography tries to do this in multiple images (which isn't always feasible, e.g. for fast-moving subjects).
  - In coded photography, neighboring pixels may have different variations (vs. neighboring images) due to controlling light w/ space & time

07-03 (cont.)

- e.g. how could we turn one image to a depth map? to an infinite-DOF image?
- Point spread function: With a "normal" aperture, we see a point source approximate the shape of the aperture as it moves in & out of focal plane, and get larger as it moves away. We can see all out-of-focus areas this way.
- Depth from defocus tries to infer depth by analyzing this - but it's fundamentally an ill-posed problem.
  - It's also really hard to tell apart smooth parts of a scene from motion blur and either from defocus blur
  - And it's hard to undo defocus blur

"Ringing" happens in deconvolution

(See Lucy-Richardson Deconvolution 1972-1974. You can't remove it because information is lost.)

- Dr. Essa is fond of saying, "What you see is all you get." in contexts like these: You can't fix a bad image.

Levin et al. 2007

- So, how can we do anything?
  1) Explicit prior on natural images to improve deconvolution & depth discrim.
  2) Make defocus patterns different from natural images, & thus easier to discriminate.
- Defocus can be seen as a local depth-dependent convolution (i.e. blur kernel depends on depth). This works okay, provided we know depth or that kernel
- But: Can we put a mask/code in aperture plane, such that we may discriminate defocus patterns?

2016-04-05c, CS7641

07-05 (cont.)

- This 'coded aperture' manipulates our point spread function.
  - It still has ringing artifacts and cannot remove all blur, but does much better than a "conventional" aperture.
  - What structure of mask helps the most with discrimination?
    - Sounds like I need to read Levin paper to know. Relates to information along center & ambiguity?
- See: "depth regularization"
- This can refocus from a single image
  - Looks like it is a DIY-able mod with an SLR lens, if one can acquire a physical mask?
- Pros of coded aperture:
  - Image & depth in one shot!
  - No resolution loss
  - Simple lens modification
  - Extra DOF with deconvolution
- Cons:
  - Depth is coarse
  - Loses some light

Raskar et al. 2006

- Flutter Shutter Camera
  - Instead of changing aperture, control when shutter opens
  - If shutter is open for period of subject motion, we simply get motion blur. But what if we flutter it with predictable patterns open & closed then we get very predictable signatures in the motion blur.
- Traditional camera is a "box filter" with shutter. This convolves to a sinc function ($\sin x / x$). The flutter shutter convolves to sinc functions too but in a way that preserves high frequencies.
- If we try to find inverse function - the filter is unstable for the box function, but is not for the coded exposure. So, the latter deblurs much better. (Though, still has noise & banding.)

07-03 (cont.)

- The shutter pattern matters too. Just alternating is insufficient; fully random doesn't work either. The authors eventually found that alternating long/short patterns with some randomness work best
- See:
- Raskar (2009), "Computational Photography: Epsilon to Coded Photography, Emerging Trends in Visual Computing, Springer 2009.
- Levin, Fergus, Durand, Freeman (2007), "Image and Depth from a Conventional Camera with a Coded Aperture", ACM SIGGRAPH 2007.
- Raskar, Agrawal, Tumblin (2006) "Coded Exposure Photography: Motion Deblurring using Fluttered Shutter" ACM SIGGRAPH 2006.