CV Part 1

Monday, January 13, 2025

5:26 PM

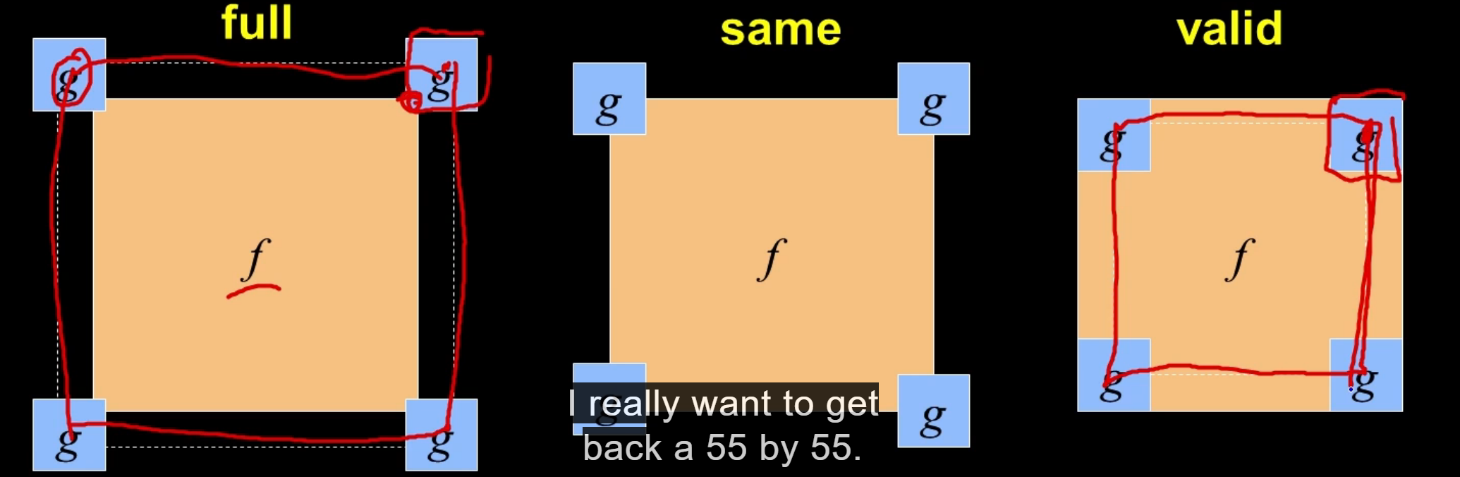
Images as functions

* Images can be blended using weighted average
* Brightness can be increased/decreased by multiplying image by a scaler
* Can plot an image using a 3d plot. Brighter parts of the image will be taller.
* Smoothing the image makes the peaks and valleys of the 3d image smoother, and this results in a blurry image.
* Blurring and image is the same as smoothing a function.
* An image is a function f(x, y) that outputs the intensity at the (x, y) position. For images, the values are bound over some range with a min and max intensity and a min/max (x, y) (width and height)
* Taking the absolute value of the difference of two images can lead to unexpected results due to clipping caused by the data types used for the image.

Filtering

* Cross correlation is sliding a window over a region. Region is a rectangle which can be specified "odd" size. Odd sizes are used so there is a "center" pixel. This sliding window region is called a kernel.
* Simple blurring or "filtering" is averaging values inside the kernel
* Smarter/better blurring uses a gaussian inside the kernel and that is called gaussian filtering. The amount of blurriness is specified by the sigma parameter. A larger kernel size can typically be better. Pick something that is large enough but not too small.

Linearity and convolution

* Additivity: H(f1 + f2) = H(f1) + h(F2)
* Multiplicative Scaling: H(a \* f1) = a \* H(f1)
* If you can sum up a bunch of things to make an image, then apply a linear operator to that whole image, it is the same as the sum of applying the linear operator to each of the pieces of the image
* Impulse - building block of a function
  + Discrete case - an impulse is a single point with a value one
  + Continuous case - small signal whose area is one
* Impulse response - "put in" an impulse" to a black box operator, the response is the impulse response
* Convolution Boundary Issues:
* 
* Methods of boundary padding:
  + 0 padding
  + Wrap around (copy from opposite side)
  + Copy edge (duplicate edge values)
  + Reflect(Duplicate edge values in a range)
* Correlation - slide kernel over image
* Convolution - flip kernel and then slide it over image
* Sharpening filter - double reference point - blur filter
* Median filter - sort the pixels and replace outliers with median

Edge Detection: Gradients

* Can compute decent edge images by summing the squared gradient in the x and y directions and square rooting that
* There are different ways to compute the gradient of an image usually represented by a well-formed kernel that is convolved over the image.
* Can take derivative of the kernel and apply that to the image instead of the derivative of the whole image and then apply the regular kernel to save computation due to associative and differentiation properties

Edge Detection: 2D operators

* Canny Threshold:
  + Apply a high threshold to detect strong edge pixels
  + Link those strong edge pixels to form strong edges
  + Apply a low threshold to find plausible edge pixels
  + Extend the strong edges to follow weak (low value) edge pixels

Hough Transform: Lines

* A parametric model can represent a class of instances where each is defined by the value of the parameters (e.g., lines, circles, or template)
* For a cartesian coordinate way of accessing pixels in an image, a point in image space is a line in hough space and vice versa.
* For a polar coordinate way of accessing pixels in an image (this is used to not have numerical problems such as lines with 0 slopes). A point in image space is a sinusoid in hough space

Hough Transform: Circles

* Finds center with radius R. Votes for circles with radius R, the intersection point is the predicted center point.
* Typically uses a known radius, for cases without a known radius:
  + Hough space has three dimensions (a, b, and r) instead of just a and b for the equation of a circle
  + The votes end up being a cone shape and it is difficult to vote in many dimensions with this algorithm
* Hough pros:
  + All points are processed independently so can cope with occlusion
  + Some robustness to noise
  + Can detect multiple instances of the a model in a single pass
* Hough Cons:
  + Complexity increases exponentially with model parameters
  + None target shapes (like squashed ellipses) can make voting hard
  + It's hard to select a good bin size for voting

Generalized Hough Transform

* Does not use a specific shape, uses an arbitrary shape
* Computes a vector from the boundary points to some center/reference point, a displacement vector

Fourier Transform

* Convert an image into a basis set using frequencies (sinusoids) (Spatial to frequency conversion)
* Decomposes into an even and an odd computing represented by complex numbers
* Can invert the Fourier transform

Convolution in the frequency domain

* Convolution in the spatial domain is multiplication in frequency domain. The inverse is also true. That is, multiplication in the spatial domain is convolution in the frequency domain.
* Small gaussian in space has a big gaussian in frequency and vice-versa
* High frequency represents more of edges and sharp structures, low frequency represents more blurry masses

Aliasing

* Microphone is a metal coil surrounded by magnets. Metal coil is on a springy mechanism. When you speak, it causes a pressure wave that causes the metal coil to move up and down. Moving a metal coil through a magnetic field creates voltage and this is recorded to record the audio.
* Aliasing is a signal that travels "looking" like other frequencies. Cannot distinguish between the signal being low or high frequency
* Antialiasing - how to prevent aliasing
  + Sample more often
  + Make the signal less "wiggly"
* Less often we sample in space (impulses farther apart), the more often we sample in frequency (impulses are closer together
* Antialiasing can reduce the "bad bluriness" associated with directly downsampling an image. Can do this antialiasing using a gaussian
* Antialiasing is important when you want to operate on smaller images to reduce computational expense. The downsampled images must be antialiased to get good results

Cameras and Images

* Images are 2d projection of 3d points
* Cameras are a device the project 3d points to some medium that records the light pattern
* That is, images inherently lose information.
* Some cameras work through an aperture. Light penetrates a small hole and shows up on a wall on the other side, upside-down
* Lense based cameras refract and focus light onto a film. Lenses also have an aperture.
* Large aperture makes depth of field smaller, that is only objects close to the focal length will be in focus
* Small aperture makes depth of field larger, so objects at varying distance will be focused (e.g., background and foreground).
* Field of view is a function of focal length
* Longer focal length is less robust to "shaking", needs more stability like a tripod
* Zooming and moving are not the same thing
* There is a significant effect of width of lenses on perspective (perspective distortion is the idea but not necessarily the correct word)
* Given the lens and camera, we can use tools like photoshop to correct distortions as lenses are not perfect
* Chromatic aberration is the red artifacts that occur when zooming
* Vignetting - lighter in center of image, darker at the edges

Camera Obscura would be a cool museum to visit

Perspective Imaging

* Academic computer vision typically assumes we have a pinhole camera, but is hard to achieve in reality, but we can mimic it.
* Vanishing point between two parallel lines (somewhat perpendicular to the image plane) in an image is due the projection
* Lines that are aligned with the image plane remain perpendicular
* Sets of parallel lines on the same plane (like the ground) have collinear vanishing points (for example, the horizon).
* Our brains can automatically try to undo perspective transformations
* Orthographic projection is a special case of perspective projection where you "smash" the world onto the image plane.
* Anamorphic images require the view to view the image from a specific perspective in order to see the intended image

Stereo Geometry

* Stereo is a special case of having multiple views of an image. There is a relationship between two views which can be used to recover depth.
* Humans are able to perceive depth. The fact that we have two eyes and therefore two images helps this.
* Having two eyes is called "stereo"
* Shadows can be a que for depth
* This is why we use makeup for shading, it changes the apparent depth of the face, changing the viewed shape.
* Recover surface height from changes in texture
* We can use how an image changes focus as you change the aperture to estimate depth.
* The key of stereo is the two images are just a little different
* Stereo is how VR googles work, you have two slightly different images and block the eye from seeing the other eyes view. This causes us to see a 3d image.
* Anaglyph stereo uses red and blue imagery to produce a stereo scenario from 1 position. This is how old 3d movies work. The blue lens causes the one eye to only see the blue imagery, while the red lens causes the other eye to only see the red imagery
* The principal is two different images are shown to each of the eyes separately
* "Twitching" between two slightly different images quickly allows us to see three dimensions/depth
* No change between the left and right image for one target point has a disparity of zero. The change is otherwise called the disparity. The farther away the target point is, the less the disparity is due to perspective. This represents the inverse relationship between depth and disparity. If disparity is zero the depth is infinite.
* You need two calibrated cameras to get matched images. Then, you can compute the depth.

Epipolar Geometry

* Epipolar geometry tells us the relationship between the two aligned/calibrated cameras. We use the constraints provided by this relationship to match points in the image. The matched points can directly be used to compute disparity, and therefore depth.
* Rays in the left image that project to a ray in the right image - the ray in the right image is the epipolar line. The same is true from the right image to the left image.
* All epipolar lines in an image converge at the epipoles. Though, we would need a very large image to observe this effect
* The epipolar constraint reduces the correspondence problem (matching points) to a 1d search along an epipolar line
* We can use verged cameras (cameras pointing slightly towards each other). Or parallel cameras, cameras that are parallel which are a special case.
* The epipoles of the verged cameras converge at some finite distance. In the parallel camera case, the epipoles are at infinity.

Stereo correspondence

* Epipolar constrains the solution set but does not solve the problem (finding point in other image)
* Find most similar pixel in other image, like sum squared differences
* Can have issues where image patches have similar textures, could use larger windows to held mitigate this, but there are downsides to a larger window that result in a less granular result. Could use a large window and only compare the middle pixel
* Occlusion can prevent us from detecting the pixel as the pixel could be blocked in the second image
* Transparent objects cause issues too
* Can use dynamic programming to match two 1d signals, which can result in jagged lines
* Can also use a 2d approach that utilizes the fact that nearby pixels should shift similar amounts, which use an energy minimization, large shifts should be penalized

Extrinsic Camera Parameters

* Stereo correspondence requires some understanding of the geometry between cameras. Calibration is used for this.
* Full relation of coordinates in world to coordinate in image is geometric camera calibration
  + Extrinsic parameters is about relating the coordinate system of the world to the camera (aka camera pose)
  + Camera coordinates in 3d camera plane to 2d image coordinates is intrinsic parameters

Intrinsic Camera Parameters

* Pixels are not actually always square, but they are more square today than historically. They may be slightly rectangular
* Extrinsic Parameters
  + Translation T of the optical center from the origin of the world coordinates (3 params)
  + Rotation R of the camera system (3) params
* Intrinsic
  + Focal length f (based on things like lens size), aspect a (or pixel size s\_x, s\_y), principle point (x', y') (center of camera sensor), and skew (angle of camera sensor)
* Above 6 extrinsic params and 5 intrinsic params let us relate world coordinate to image coordinate

Calibrating Cameras

* Fundamental calibration is we have some points in the world, and its location in the image, then, we can recover the 11 camera parameters
* Resectioning - Estimating the camera calibration matrix from known 3d points. Useful for mapping and cartography
* Direct Linear Calibration
  + Advantages
    - Simple to formulate and solve. Take points, make A matrix, do SVD, M is final column of that
    - Minimizes algebraic error
  + Disadvantages
    - Doesn't directly tell you camera parameters
    - It is approximate
    - Hard to impose contraints like known focal length. Can buy an expensive camera with a very precise known focal length
    - Doesn't actually minimize the right error function. Really want to minimize geometric error
* Geometric error - minimize error between a set of known points and where a particular M matrix (the camera params) predicts the location of the points
* Gold standard way of doing this is to normalize the numbers between 0 and 1 for numerical stability, and doing the direct linear calibration to get a starting point, then minimize the geometric error using some non-linear optimization method, then denormalize if you normalized.
* Now, M matrix represents extrinsic and intrinsic parameters. Can use M matrix to find location of camera in the world.
* Cool way of doing calibration of a camera to a robots coordinate system. Put a checkerboard on a robot arm, move it around, calibrate using OpenCV. Then camera is calibrated to robot arm

Image to image projections

* Projecting from one camera view to another
* Can have translations, rotations, similarity transforms (Can be rotated, translated, and scaled), affine transform (can map any three points to any other three points. Results in ability to translate, rotate, scale, and skew the image. Maintains parallel lines and ratios of areas, 6 degrees of freedom)
* General Projective Transform (or homography) - more general transform. Just preserves lines 8 degrees of freedom
* Translation is a 1 point transformation with 2 unknowns. A 2d point gives 2 equations/knowns.

Homographies and mosaics

* Image reprojections - determine where point in one image is in the other image. Don't need to know where the point is in the world. Need to know the camera center, the location of the point in image 1, then intersect the point into image plane 2. So, it is just a 2d image warp.
* Panoramas are created using homographies
  + Take a sequence of images from the same position, but rotate about optical center
  + Compute image transform between first and second image, transform the second image to overlap with the first, then blend the images together. For more images, repeat.
  + Basically projected everything down to a common plane.
  + In practice, panorama images are done by projecting the image onto the cylinder, then unwrapping the cylinder. The benefit of the plane is it gave you an image as if it came from a super wide camera
* To compute the homography for the panorama, we need to map four points in the first image to four points in the second image
* RANSAC - can be used to segment out uncommon objects with a common background
* Content-based manipulation/coding - can segment a tennis player out of the image using RANSAC, then stitch together their backgrounds using a panorama, and have the tennis player moving on the wide background. Can also use this for airplane-based images of cars
* Image rectification - Can take a slanted image and make it rectangle using other rectangles in the image, then can measure items in the image
* Can rectify a tiled floor using the tiles to measure it
* Can get an overhead view of a football field using this technique and the rectangles formed by the yard lines. Only the players feet are on the plan, so the players are stretched out in the result image
* There is forward and backward image warping. Inverse is the better way. Given a pixel in the output warped image, find the pixel in the original image. This is the color to use in the output.
* Guessing a point between two known points is called interpolation and is important for inverse warping. Bilinear interpolation interpolates linearly in two directions

Projective Geometry

* Projective geometry is useful to determine the relationship between multiple views. Projective geometry is a convenient way to represent the projection operator which is how images are made.
* Points and lines are a dual in projective space. Given any formula, can switch the meanings of the points and lines to get another formula
* Line joining two points is the cross product of the points (when the points are defined in terms of projective coordinates)Y
* Intersection between two lines - cross product of the two lines
* A point is on a line if P dot L is 0
* Projective geometry generalizes naturally to 3D
* 2d projective geometry has a duality between points and lines as both are represented by 4 dimensional vectors
  + Projective transform is represented by a 3x3 matrix
* 3d projective geometry has a duality between points and planes
  + Projective transform is represented by a 4x4 matrix

Essential Matrix

* Represents the relationship between two calibrated cameras, that is, we know the translation and rotation matrix between the two cameras
* The essential matrix relates the point in one frame to the point in the other frame, in world points

Fundamental Matrix

* Represents the relationship between two not calibrated cameras
* The fundamental matrix defines the relationship between the points. If we have enough image point pairs, then we can solve for the fundamental matrix (F)
* F transpose times a point in one image gives the epipolar line in the second image
* That is, F gives us the epipolar constraint between two views
* F Matrix can also give us the epipoles by solving Fp' = 0 and F.transpose() \* P = 0, which is the epipoles in the corresponding image
* Fundamental matrix is a 3x3, but is actually singular (rank is two and not three)
* F matrix is more general than the essential matrix as it removes the need to know the intrinsics. Since the intrinsics are not needed, the F matrix can relate arbitrary camera types
* F matrix can relate views taken from arbitrary camera centers. For instance, can take one image, walk forward, and still relate the views using the F matrix. In this case, epipole is in the center of image-ish
* You can rectify two plans onto a common plane parallel. Pixel motion is horizontal after the transformation

Introduction to "Features"

* Not the features from machine learning
* Idea is to find reliably detectable and discriminable points in a scene
* Features are things that we compute about some local spot. Related to feature vectors in ML but not quite
* Precisely determine point in one image in another related image

Finding Corners

* Compute gaussian derivatives at a pixel
* Find second moment matrix M in a gaussian window around each pixel
* Compute Corner response function R
* Threshold R
* Find local maxima of response function (nonmaximum suppression)

Scale Invariance

* Harris corner detector is invariant to rotation
* Harris is mostly invariant to additive and multiplicative intensity changes
* Harris corner detector is not invariant to image scale
  + Consider regions of different sizes around a point
  + Corresponding sizes will look the same in both images
  + Importantly, the scale invariant region sizes are done independently for images
  + Need some function to use over the region that provides a min or max for a good scale of the region. Average intensity over the region was an example but isn't that great
  + Could use a laplacian of a gaussian. Can estimate the laplacian by subtracting two gaussians where the second gaussian's sigma is multiplied by some factor

SIFT Descriptor

* Now that we have interest points, we need to match them so we can compute homographies
* Need a descriptor to match points
  + Descriptors should be invariant, should be almost the same in both images
  + Trade-off between being invariant and distinctive

Matching Feature Points

* Matching key points in one image to another
* Can use nearest neighbor (best bin first, modification to KD tree), wavelet-based (filter) hashing, locality sensitive hashing
* SIFT is translation rotation scale invariant
* Sony Aibo used SIFT detectors for commands given from a card. The card looked fairly strange so it wouldn't be a common thing to occur in the environment

Robust Error Functions

* Compute putative matches
* Try to keep the best matches
* Take ratio of 1st nearest neighbor to second nearest neighbor. Larger ratio is better
* Taking one image and fitting it to the other is really model fitting, a relationship between entities

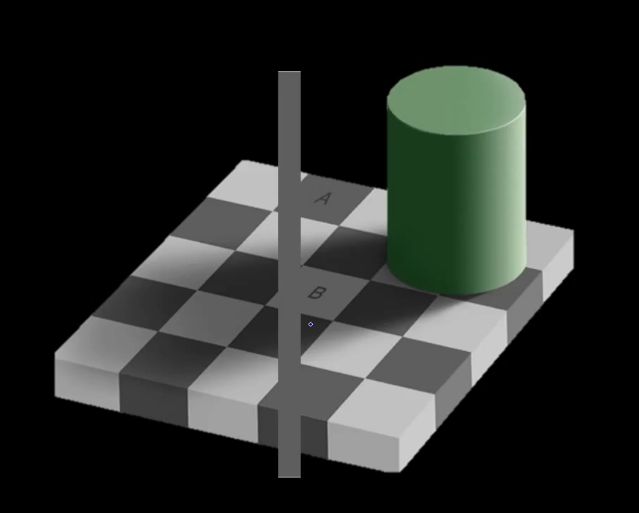
RANSAC

* RANdom SAmple Consensus
* If we have a proposed line, we could probably guess which points belong to that model (inliers)
* Repeat until you get a good line (one with lots of inliers)
* This works with a large number of outliers
* Minimal Set - number of points required to compute model
  + Translation - 1
  + Affine Transform - 3
  + Homography - 4
  + Fundamental Matrix - 8
* RANSAC Params
  + Initial number of points (typically minimum set)
  + Distance threshold t, need enough distance for probability of inliers to be high.
  + Number of samples N. Make N large enough so that at least one random sample set is free from outliers
  + Smaller t requires bigger N since smaller t means the probability of being an inlier is smaller. On the other hand, smaller t can be better results
* RANSAC is fairly scale invariant to the number of points in the image

Photometry

* How the physics of light interacts with objects with respect to making images
* Reflection in windows
* Refractions, bending light in water and similar
* Interreflections - multiple reflections. Ray tracing - as rays of lights bounce off, create illumination in multiple places
* Scattering - light changes due to moisture or fog in the air
* Radiometry - measurement of light
* Radiance (light coming out of surface) - energy carried by a ray, measured in watts per square meter per "steradian" (measure of solid angle such as a cone (angular area of the cone) where a radian is a measure of angle) of an area perpendicular to the light source
* Irradiance (light landing on surface) - energy at the surface. Measured in watts per meters squared. How "focused" the light is. If the light source was even with the target area, the irradiance would be zero since the same amount of light is spread over a larger area.
* Light hitting camera from source is model using a Bidirectional Reflectance Distribution Function (BRDF). Measured as radiance from "viewed" surface divided by "irradiance" at surface.
* Reflection Models
  + Surface of the material that light hits impacts radiance. Diffuse reflection is a matte appearance (non-homogeneous medium)
  + Specular reflection - glossy appearance, more light bounces off
* Image intensity is a combination of body reflection + specular reflection
* Physics - angle in is angle out
* Glossy - The more glossy something is, the more light spreads when reflecting off of it
* The BRDF of many surfaces can be approximated by Lambertian + Specular model

Lightness

* Brain makes assumptions automatically to determine lightness. Brain compensates for shadows, for example. Depending on where shadows are, parts of the image can be the same intensity but our brain interprets them as different.
* 
* Retinex algorithm can remove some shadows
* Humans are good at seeing what color an object would be with a brighter white light source, so can remove the effect of darkness
* Still a difficult, open problem. Computational methods still aren't good enough.

Shape from shading

* Recover something about the shape of something from shading
* Gaussian Sphere represents all possible surface normals
* Shape-from-shading - doesn't really work in the real world
* Photometric stereo - take more images, and can extract shape. Has real application
  + Take pictures of the same object but with different lighting (e.g., different times of day, recovers 3d shape
  + With enough lights and careful calibration, can also recover reflectance function, which can be used to normalize an image
* Makeup is an application of shape from shading

Introduction to Motion

* Give computers an intuition on motion/videos, represented as several images with a time component
* Shot boundary detection (movie switches scenes), background detection (segment out moving portion)
* Can use motion to make a large mosaic image from a video
* Improve video quality (motion stabilization)
* Multiple approaches to motion estimation
  + Feature based methods, extract visual features and track them
    - Sparse motion fields, but more robust tracking
  + Direct, dense methods
    - Recover image motion at each pixel from spatio-temporal image brightness variations (how appearance changes over time)
  + Best methods today use both feature based and direct methods to estimate motion

Dense Flow: Brightness Constraint

* Optic flow is the apparent motion, needs some features in the image to detect this
* Color Constancy, if we knew how a pixel moved between two frames, the two pixels would be the same color
* Between frames, we assume the points don't move very far
* Aperture problem, can only tell the motion locally in the direction perpendicular to the edge
* The assumption is that ideally the whole entire image moves the same amount in the x and y direction. Not perfectly true, but we still want to minimize the number of pixels that move differently. We use this as a constraint to solve flow
* It is probably better to impose these constraints locally, which is what is done in the next lecture

Dense Flow: Lucas and Kanade

* Assuming that over a window, motion is smooth (i.e. the same in the window
* Errors/assumptions in Lucas-Kanade
  + Large motion - Taylor expansion doesn’t hold
  + Difference in brightness - pixels change slightly
* Has an iterative version that helps with errors
* In practice, useful to take a low pass filter of the images before motion estimation
* Summing over a window gives use the ability to create a least squares solution, with two large enough eigenvalues

Hierarchical Lucas-Kanade:

* Iteratively reduce the number of pixels in the image until a good result is achieved
* This reduces the motion between the pixels
* Subsampling - smooth image before removing pixels
* Band-pass approximate filtering are difference of several gaussian filtered images which estimate a Laplacian
* Separable filters can be applied in the x and y direction separately
* Reduce
  + Simply remove every other pixel
* Expand
  + Use half of adjacent pixels
* Sparse LK - Apply LK only to "good" feature locations

Motion Models

* Add additional constraints based on how we understand motion will occur in the environment. Such as focusing on rotating objects
* Homography is also known as a perspective transform. Lines no longer remain parallel. In an affine transform, lines remain parallel.
* Determining how motion is occurring in the image is a transform from world physics to image physics
* Affine Motion
* Layered Motion
* Motion segmentation to grab pixels that move together
* Feature based motion estimation is good at estimating motion for a small number of pixels but can handle large movements. Direct motion is good at handling many pixels motion but only a small distance

Tracking As Inference

* Raw measurement is noisy
* State changes in each state
* Hidden state is the true parameters we care about
* Estimate most likely (or distribution) of the hidden state
  + Given all observations so far
  + Knowledge of dynamics of state transitions (physics, for example)
* Prior - previous state belief
* Measurement - new noisy estimate
* Posterior - new belief given the prior and measurement
* Prediction - Predicted state given the prior, not current state
* Correction - predicted state given the prior including current measurement
* Markovian assumption for simplicity - current prediction only requires previous immediate prediction (Dynamics Model) (also an independence assumption)
* Probability of current measurements only depend on current state (Observation Model, this is typically the most suspect assumption in the model)
* Given corrected estimate for frame t
  + Predict for current frame t + 1
  + Correct for frame t + 1
* Corrected measurement is the probability of the measurement times the prediction per bayes rule

The Kalman Filter

* State undergoes linear transformation plus gaussian noise
* Kalman filter tracks a linear system with assumed gaussian noise. Tracks state using a gaussian, mean and covariance
* Can add measurement noise to reduce certainty based on knowledge of sensor quality
* Kalman loop is predict, take new measurement, correct based off measurement, then time advances
* Correction is a weighted average of the prediction and measurement (weighted based on variances) (this is integrating two sources of information, which is basically what we want to do).
* Edge cases
  + If you are sure of prediction (variance of measurement is zero, measurement is ignored), then just use prediction
  + If you are sure of measurement (variance of prediction is zero, prediction is ignored), then just the measurement is used.
* Kalman gives mechanism to tradeoff how much you pay attention to prediction vs how much you pay attention to measurement based on uncertainties of both.
* If kalman gain is zero, measurements are ignored. If kalman gain is one, predictions are ignored
* Corrected mean is predicted mean plus kalman gain times residual (of predicted mean and measurement)
* Kalman gain is between zero and one in the scalar case and is a matrix in the multidimensional case
* Predictions increase uncertainty, while incorporating measurements after a kalman step reduce uncertainty
* Information can only reduce uncertainty
* Kalman Pros
  + Simple updates, compact and efficient
* Kalman Cons
  + Unimodal Distribution (can only predict one point at a time)
  + Restricted class of motions defined by a linear model
    - Extensions to improve this called extended kalman filtering. EKF uses jacobians to approximate nonlinear dynamics, while base KF is restricted to linear motion

Bayes Filters

* Allow for perturbations in movement, movement is not just linear
* Particle filters are a sample based method, usually weighted samples
* Bayes filter is a more general form. What is the expected change, not assumed to be linear. Particle Filters is one way to represent the more flexible distribution

Particle Filters

* Want to have an arbitrary density rather than represent state with a gaussian as done with kalman filters
* Have a set of particles, use weight of particles and several particles to represent the density.
* Sample particles, then do action, then add uncertainty to that result

Particle Filters for Localization

* odometry - how you think you moved based on how your wheels turned
* Use odometer or similar for "deterministic" portion, but measure using other sensors to improve the result
* If the weights have a certain amount of uniformity, probably do not need to sample again
* Better to overestimate noise and let measurements narrow things down than think measurements are more accurate than they are.
* Each iteration, add some new particles in case things changed unexpectedly to help recover from failure

Particle Filtering for Real

* Condensation - Conditional Density Propagation for Visual tracking - brought particle filtering to CV domain
* Can get dynamics by cheating and measuring the dynamics in an easier system

Tracking Considerations

* Mean-Shift Tracking - use hill climbing to climb in the direction of maximum match
* Need similarity between two histograms to do mean-shift tracking. Can use Bhattacharyya coefficient
* Could also use this similarity function as a sensor model for particle filtering

Introduction to Recognition

* Verification - is this a certain thing? A lamp?

Classification: Generative Models

* Generative - use training data of A to build a model of class A
* Discrimative - given a's, b's and c's, differentiate between them
* If you remove illuminate, all skin is a similar color

Principal Component Analysis

* Generative models work better in low dimensional space
* PCA represents the direction of maximum variance (the most change)
* Principle Eigenvector is eigenvector with maximal variance
* Next principle Eigenvector is perpendicular as eigenvectors are perpendicular to each other
* The direction that captures the maximum covariance of the data is the eigenvector associated with the largest eigenvalue of the data covariance matrix
* Faces represent a small space of pictures
* Finding eigenvalues directly is very computationally expensive, so better ways of computing the principle components have been developed. Can compute eigenvalues of a lower dimensional matrix to compute them for a higher dimensional matrix
* Subtract out the mean to get number of eignevectors possible. So it is M points minus 1 (M-1) is on the final
* The direction of maximum variance is not always good for classification

Appearance-Based Tracking

* Eigentracking - distance from face space
* PCA can be used for tracking with eigenvalues

Discriminative Classifiers

* Examples are boosting and SVMs
* Model decision boundaries between classes
* We assume that we know what all the classes are in advance
* Xbox kinect did their skeleton tracking things using random forests

Boosting and Face Detection

* Boosting works by iteratively weighting incorrect classifications highly so that they are more likely to be correctly classified with the next weak learner
* Used prespecified kernels to create weak learners
* This face detection method works very well and is used in a lot of the face detectors today
  + Rectangular features and integral image
  + AdaBoost for feature selection
  + Cascade

Support Vector Machines

* Support vector machines separate classes linearly but can project the data to a higher dimensional space to make it easier to separate linearly
* They separate with a maximal margin
* Kernel Functions are also known as similarity functions that correspond to an inner product in some expanded feature space
* Slack variables allow some points to be misclassified to favor a better separator
* Gaussian RBF kernel - allows the surface to move nicely near support vectors
* Histogram intersection kernel is a common kernel in computer vision. It is a very simple kernel, sum up the min values from the two histograms for each bin
* Support vectors are the "harder" points near the boundaries. The call the support vectors closest to the boundary the "support faces"
* Couple messy ways to apply SVMs to multiclass as SVM is naturally binary.
  + One vs All - Pick the value furthest from their decision boundary (N SVMs)
  + One vs One - learn an SVM for all pairs of classes. Count votes (more than N SVMs)
* Only need to compute for support vectors at inference time

Bag of visual words

* Punitive matches are possible matches
* When we have close points in feature space, the images might have similar descriptors and therefore may have similar content. Would have millions of patches to search.
* Can improve this problem using indexing. We want to find all images where a feature occurs. We map our features to "visual words"
* There are way more documents than words out there
* Summarize entire image based on its distribution (histogram) of word occurrences

Introduction to Video Analysis

* Background subtraction with a good estimate of the background
* Frame differencing - compare current frame to previous frame
* Mean filtering - compare to last N frames (average). Not great because mean is not robust to outliers
* Median Filtering - Take median value - can be expensive since median needs a sorted list
* Adaptive Background Mixture Models for real-time tracking - mixture of gaussians for detection
* Special cameras that provide RGBD (Depth). Can do background subtraction on the depth channel
* As camera moves to the left, objects move to the right
* Epi-image analysis can be used when slicing a video volume in different directions to pull out objects, perpendicular lines tend to be objects

Activity Recognition

* Model based action recognition - use human body tracking and pose estimation to create action descriptions. Easy if your training data from place and camera where you want to do recognition
* Model based Activity recognition - given some lower level detection of action and recognize the activity by comparing structural representation of the activity
* Activity as motion, space time appearance patterns
* Activity recognition is typically done in video, can do "activity recognition/pose estimation" in single images
* Motion history image - decay parts that decayed long ago
* Image Moments -
* Hu Moments - translation, rotation, and scale invariant
* Virtual PAT (Personal Aerobics Trainer) - uses MHI recognition. Portable IR background subtraction system

Hidden Markov Models

* Learning - Construct a model from a bunch of samples
* Inference - given a set of models, which model most likely generated the data
* Markov Model - predict tomorrow based on today - first order markovian. Second order if today and yesterday for tomorrow, and so on
* Hidden Markov Model - Cannot observe state like weather. Can only see observables like the clothes people are wearing that day.
* Emission Probabilities - probability of seeing a symbol given some state
* 3 problems of HMMs
  + Evaluation - classification/recognition problem
  + Decoding - optimal state sequence to produce observation sequence. Useful if states have some meaning, like in genome work
  + Learning - determine model given a training set of observations. Maximize likelihood of getting observations (EM, expectation maximization)
* Naïve - sum over all possible paths and all possible state sequences O(TN^T) complexity
* Efficient - O(N^2\*T) - uses recursion and only uses previous state from markov property
* Forward - compute likelihood of being in state I at time t given a sequence from the start until t, given trained HMM
* Backward - recursive algorithm could compute the likelihood of being in a state I and time t and observing the remainder of the observed sequence given trained HMM
* If we know HMM, we can use the forward and background algo to get an estimate of the distribution over which state the system is at time t
* With those distributions and having actually observed output data, I can determine the emission probabilities that would maximize the probability of the sequence
* Given distribution about state can also determine the transition probabilities to maximize probability
* With the new state can get a new estimate of the state distributions, and repeat
* Baum welch algo
* HMM are generative models of time series
* Can train using forward-backward algos
* Conditional Random Fields are better than HMMs with big data for segmentation
* Chain Code - entire counter is a sequence of discrete steps (HMMs)
* Can combine multiple HMMs for a sequence of gestures, natural extension
* HMMs - learning paradigm. Recognition is fast, training is slow. Not as good for some segmentation as newer methods (CRF)
* Works well when the problem is easy (when regularities are clear) Then need to understand where the difficulty is.

Color Spaces

* 6-7 million cones in the retina - responseible for high resolution vision and discriminating color. There are Red, Green, and Blue cones. Not all light can be made with RGB
* People who are missing one of the three cones are colorblind. Almost all people who are color blind are men
* Color can be the same but luminence is different making things appear different colors
* YUV and lab space
* HSL and HSV space
* HCL space is prof's favorite. Basically a rotated RGB space
* Differences in compatible color spaces can make images come out different from printers or appear different on monitors and tvs.
* Outside color gammit of printer
* Can plot colors in 3d space. May be able to use this separation to segment
* Pixels that are similar in color could be grouped
* Getting darker means RGB pixels get closer together, getting lighter means rgb pixels get closer together
* Green contributes the most to luminance
* YUV space - Y looks like black and white image u and V space contain color information. Represents color in the two channels
* Why YUV
  + Easier clustering of pixels
  + Human vision is more sensitive to luminance than color
  + Can compress imagery using more bits for y channel than u and v channel
    - YUV422 - 2 bytes for y, 1 byte each for U and V. Hard for humans to distinguish the difference
    - YC\_bC\_r / YP\_bP\_r - video transmission, compression
    - CIE L \* a \* b\*
      * Based on human perception
      * Intensity channel: L \* = brightness
      * Color component: a\* = red-green, b\* = blue-yellow
* Looking at just blue channel of video has a lot of noise in it. Because they dedicate hardly any bits to encoding blue - doesn't matter to human vision

Segmentation

* Segmentation is a form of clustering
* Figure Ground Segmentation - pull out foreground from the background
* Superpixels - small selections of similar regions
* Cluster histogram into groups
* Good clusters minimize sum of squared differences
* Average is the value that minimizes sum of squared differences
* Taking average generates the value that would provide maximum likelihood
* Having to specify the number of clusters with k-means is a problem
* Clustering is carving a continuous space into buckets, aka quantization
* Can also cluster in rgb space
* Cluster on intensity and position to separate similar clusters in different locations
  + Also can do color + position
* K Means is guaranteed to converge
* K means is memory intensive
* Need to pick k - problem
* Sensitive to initialization and outliers
* Only finds spherical clusters

Mean Shift Segmentation

* Mean shift algorithm seeks modes or local maxima of density in the feature space
* Adding texture features can help reasonable segmentation
* Use Textons from a bank of filters and cluster twice, cluster in big feature space (textons) and also cluster on color space

Segmentation by Graph Partitioning

* Think of segmentation problem as a graph where each pixel is connected to adjacent pixels with weighted edges between
* Segmenting the image becomes breaking the graph into segments. Delete links between segments (those with lowest affinity)
* Graph cut algorithm cuts a graph (min cut or max flow algorithm to cut the graph at the lowest affinity)
* Normalizes cuts fix the bias of min cut that pulls off little elements
* Need to use feature matching or similar to compute affinities

Binary Morphology

* Improving blobs - morphology
* Connected components analysis - find connected blobs, number of the object to which it belongs
* Dilation expands objects - fills holes and gaps
  + Place origin of structuring element at each pixel, take OR between binary image and structuring element if an element in result goes to 1, origin is set to 1
  + Circle with plus is dilation
* Erosion - shrink the shape. Can remove partially connected objects
  + Place origin of structuring element on each pixel, take AND, all 1s in structuring elements must touch a 1 or pixel is set to 0
  + Circle with minus is Erosion
* Want to disconnect parts while not reducing in increasing shape size
* Open and Closing are two of the most useful binary morphology operations
* Open - Erosion followed by dilation
* Closing - Dilation followed by erosion
  + Fills concave corners but leaves convex ones
* Idempotent - repeat operation and it makes no change - both closing and opening are idempotent, only works once
* Open follow by close can give interesting results
* Other basic morphological algorithms
  + Boundary Extraction
    - Take object and subtract out erosion of it
  + Region Filling
  + Extraction of connected components
  + Convex hull
  + Thinning - 1 pixel wide thin band
  + Skeletons
  + Pruning
* Circle with a star inside is a hit or miss operator
* Can detect faults in gears using morpology

3D Perception

* Depth sensing, what is the physical 3D structure of this object
* Color is not always discriminative of what is there
* Passive 3d sensing
  + Work with naturally occurring light, does not add its own light
  + One option is to (de)focus, vary focal plane to figure depth
* Active 3d sensing
  + Project something into the environment to recover depth
  + Recover shape with known light sources
  + Time of flight like with LIDAR (light, laser)/SONAR (sound)
  + Lidar only gives slices of depth so you sweep the laser up and down
  + Structured light, send out light and use camera to detect where that light hits on the image plane. Uses bends to determine depth
  + Use infrared structured light. Its nice because it isn't detectable by human vision, so does not mess with what we see.
  + Orientation as a function of distance
* Kinect - projects specled IR dots into the room and recovers depth using properties of the light
  + Two kinects would confuse one another because theres no way to distinguish the kinect source and could be placed in different positions, uses different ir colors
  + IR dots would be washed out by sunlight so it can only work indoors
  + Can be confused by mirrors/light bouncing
* Depth sensor provides multiple forms of information. With a dense depth map, it does not only tell you the depth of different objects, it also can help separate objects as there will be significant gaps between depths of objects at different locations in the image. The depth also represents free space as the ray can move from image source to destination by definition
* Cons
  + Depth image is viewpoint dependent
  + Doesn't capture physical geometry
  + Need to know where camera is to represent where geometry is in space
* Point Clouds
  + Voxel
* Biggest advantage of point clouds is the point cloud library (PCL) that provides many operations for point clouds
* Can use feature matching to align points in 3d space using descriptors of points
* RANSAC to find the right sets of matches
* Kinect can be used to get point clouds

The Retina

* 30,000 nerve fibers for hearing, 1 million for vision. Demonstrates that vision is more complex
* 30% of the cortex has something to do with vision
* Humans have about 200 degree field of view. Non-Predator animals tend to have wider field of view to detect predators but has the cost of less stereo vision
* How does the brain remove overlapping information that is provided by the individual eyes
* Hemifield neglect, after a stroke on one half of the brain there would be a hole on the opposite. This person wouldn't have a great big black hole on that side of the field of view, the field of view would simply be smaller
* We tend to use planer retinas but the eye has a spherical retina which has nice properties
* Lens is pulled on by fibers to focus near and far in the eye
* Degradation of these lens fibers can cause us problems seeing close as we age
* Area where blood vessels and nerves come together in the eye is the optic nerve and is a blind spot for us
* The human eye is an inverted retina where the light passes through other objects before hitting our cones
* The clear gunk light passes through should place shadows on our eye. Our brain can do a cool think and remove anything that is perfectly stable in the eye
* Rods are more light sensitive than cones
* Cones allow us to discriminate color, responsible for high resolution vision
* Eyes to adjust to a room means that photoreceptors in your eye are adjusting to the amount of light, not pupil retracting as that’s near instant
* Rods and cones are cells with a nucleus
* Electromicrograph to picture them
* More red and green cones than blue
* Pixels in eye are in hexagon grids
* Periphery of high resolution is constructed by the brain, periphary is actually low res. Does this to detect motion so eye can be pointed there
  + Fovea is high res and periphery is low res
* High dynamic range features on phones for example blend together a series of pictures to automatically select how much each region in the image should be dialed up or down to get a smooth visual representation
* Candellas, unit of energy

Vision in the Brain

* Data is combined
* Single cells can fire for specific objects or collection of objects
* Can use dies and use an fMRI to see what parts of the brains light up when seeing specific objects