

**Linear Filters** *Impulse Noise*: Random occurrences of white pixels

*Gaussian Noise*: Variations in intensity drawn from a Gaussian normal distribution

**Filters:**

*Linear*: Obeys Superposition, Shift-Invariant, Casual, and Stable

*Smoothing Filter*: Values positive,  $\sum F = 1$ , amount of smoothing proportional to mask size, remove high-frequency components; low-pass filter

*Low-Pass*: Smoothing, removing noise, blurring

*High-Pass*: Sharpening, accentuates edges

*Derivative Filter*: Opposite signs to get high response in areas of high contrast,  $\sum F = 0$ , High absolute value when high contrast

*Correlation Filter(Cross-Correlation)*: Pixel is linear combination of surrounding pixels,  $G = H \otimes F$ .

*Gaussian Filter*: Linear, smoothing,  $\sigma$ =variance, kernel=size of mask

*Sharpening Filter*: Accentuates differences with local average, subtraction

*Convolution*: Linear, Flip the filter in both dimensions (bottom to top, right to left), then apply cross-correlation,  $G = H \star F$ ,  $\frac{\partial}{\partial x}(H \star F) = (\frac{\partial}{\partial x}H) \star F$

*Shift Invariant*: Operator behaves the same everywhere; the value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood

*Superposition*: The response to a sum of inputs is the sum of the responses to the individual inputs

*Seperability*: A 2D filter is separable if it can be written as the outer product of two 1D filters

*Median Filter*: Non-linear, no new pixel values, removes spikes, good for impulse + salt & pepper noise, edge preserving

*Laplacian Filter*: Hybrid images, Unit impulse - gaussian  $\approx$  laplacian of gaussian

**Edge Detection** *Edge*: rapid change in image intensity, extrema of the first derivative, zero-crossings of the second derivative

*Steps*: 1. Smooth/Suppress noise, 2. Edge enhancement/Filter for contrast, 3. Edge localization /Local maxima/Threshold/Thinning

**Canny Edge Detector:**

*Algorithm*: 1. Filter image with derivative of Gaussian then get magnitude/direction of gradient, 2. Non-maximum suppression, 3. Linking and thresholding (hysteresis)

*Property*: Threshold strong edges and weak edges using two thresholds, then keep weak edges only if connect to strong edges.

*Property*: Filtering a signal  $f$  with a Gaussian and then calculating its gradient is the same as filtering the signal  $f$  with the first order derivative of the Gaussian.

**Seam Carving** *Energy*:  $\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

*Seam*: Path of least energy, 8-connected, 1 pixel wide

*Cumulative Energy Map*:  $M(i,j) = \text{Energy}(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$

**Template Matching** *Template*: Uses filters that match effect intended to search

*Algorithm*: Use normalized(Controls relative brightness) cross-correlation score to find template in image

**Binary Image Analysis** *Steps*: 1. Threshold image into binary form, 2. Clean using morphological operations, 3. Extract separate blobs/Connected components, 4. Describe blobs with region properties

*Otsu's Method*: Find threshold that minimizes intra-class variance

**Morphological Operations:**

*Dilation*: Closes gaps and fills holes,  $I \oplus [1]_{i \times i}$

*Erosion*: Erode connected components, Shrink features, Removes bridges/branches/noise,  $I \ominus [1]_{i \times i}$

*Opening*: Erosion then dilation, removes noise/small objects, keeps original shape

*Closing*: Dilation then erosion, closes small holes inside or small black points on the object.

**Texture Segmentation**: Analyze/represent texture, group image regions with consistent texture

*Synthesis*: Generate new texture patches/images

*Filter Banks*: Collection of (d) features, d-dimensional vector, contain combination of scales, orientation, and patterns

*First-order Markov Random Field*: Each pixel depends on its neighbors,  $P(X|A,B,C,D)$

*Single Pixel Synthesis*: To synthesize  $x$ , pick one matching window at random, assign  $x$  to be the center pixel of that window, find the best matches using SSD error and randomly choose between them, preferring better matches with higher probability

*Window size*: Larger  $\rightarrow$  More uniform, Smaller  $\rightarrow$  More randomness

*Image Quilting*: Synthesize texture by stitching together overlapping patches from a sample image

**Optical Flow** *Description*: Distribution of the apparent velocities of objects

**Hough Transform** *Algorithm*: 1. Get edges of image, 2. For each edge point vote for possible parameters(Increment accumulator array), 3. Threshold accumulator array to find matches

*Pros*: All points are processed independently, so can cope with occlusion/gaps, Some robustness to noise: noise points unlikely to contribute *consistently* to any single bin, Can detect multiple instances of a model in

a single pass

*Cons:* Complexity of search time increases exponentially with the number of model parameters, Non-target shapes can produce spurious peaks in parameter space, Quantization: can be tricky to pick a good grid size

**RANSAC** *Description:* Choose  $s$  (minimum size for fitting) samples, Fit a model to the samples, Count inliers, Repeat then take model with largest set of inliers  
*Adaptive RANSAC Algorithm:* Let  $N=\infty$ ,  $S_{IN}=\emptyset$  and  $iter=0$

While  $N > iter$ :

Estimate parameters  $a_{tst}$  from a random  $n$ -tuple from  $S$

Determine inlier set  $S_{tst}$ , i.e., data points within a distance  $t$  of the model  $y = f(x; a_{tst})$   
If  $|S_{tst}| > |S_{IN}|$ :

Set  $S_{IN} = S_{TST}$ ,  $a = a_{tst}$ ,  $w = \frac{|S_{IN}|}{|S|}$  and  $N =$

$\frac{\log(1-p)}{\log(1-w^n)}$  with  $p = 0.99$

end-if

iter++

end-while

*Pros:* Robust iterative method for estimating the parameters of a mathematical model from a set of observed data containing outliers, Separates the observed data into inliers and outliers, Can be applied in an iterative manner to obtain multiple models

*Cons:* Not perfect

**Robust Fitting** *Description:* Non-linear optimization problem that must be solved iteratively, Least squares solution can be used for initialization, Adaptive choice of scale: approx. 1.5 times median residual

**Active Contours-Snakes Internal Energy:**

*Formula:*  $E_S = \alpha \left| \frac{dv}{ds} \right|^2 + \beta \left| \frac{d^2v}{ds^2} \right|^2$

*Elasticity:* 1st-order term, Membrane,  $\alpha$  controls tension along spline, Stretching balloon /elastic band, Ideal curve is point

*Stiffness:* 2nd-order term, Thin plate,  $\beta$  controls rigidity of spline, Bending thin plate /wire, Ideal curve is circle

*Cons:* Sensitive to initial position/parameters, Small capture range, Fails to detect concavities

**Level Sets** *Description:* Embed curve in one higher dimension; curve is given by zero level set of implicit function (i.e., intersection of function with  $z=0$ )

*Pros:* Curve may change topology and form sharp corners ('weak solutions'), Discrete grid and finite differences approximate solution, Intrinsic geometric properties are easily determined (normal vector, curvature), Formulation is same for 2D or 3D

**Grouping** *Bottom-Up:* Pixels belong together because they look similar

*Top-Down:* Pixels belong together because they are from the same object

**Bottom-Up Segmentation via Clustering:**

*Description:* Separate image into coherent objects, Group together similar-looking pixels for efficiency of further processing

*Algorithms:* Mode finding and mean shift: k-means, expectation maximization, mean-shift, Graph-based: normalized cuts

*Features:* Color, texture, Quantization for texture summaries

*Clustering:* Unsupervised learning, Detect patterns in unlabeled data, Useful when don't know what you're looking for, Requires data, but no labels

**K-means:**

*Algorithm:* 1. Randomly initialize  $k$  cluster centers, 2. Assign each point to nearest cluster center, 3. Solve for mean of each cluster, 4. Repeat until convergence

*Pros:* Simple, Fast to compute, Converges to local minimum of within-cluster squared error  
*Cons:* Simple, Non-deterministic, Requires initial means and  $k$ , Sensitive to initial centers, Sensitive to outliers, Detects spherical clusters, Assuming means can be computed

**Mean Shift:**

*Description:* Cluster: all data points in the attraction basin of a mode, Attraction basin: the region for which all trajectories lead to the same mode, Find features (color, gradients, texture, etc), Initialize windows at individual feature points, Perform mean shift for each window until convergence, Merge windows that end up near the same "peak" or mode

*Pros:* Does not assume shape on clusters, One parameter choice (window size, aka "bandwidth"), Generic technique, Find multiple modes

*Cons:* Selection of window size, Does not scale well with dimension of feature space

**Normalized Cuts:**

*Pros:* Generic framework, flexible to choice of function that computes weights ("affinities") between nodes, Does not require model of the data distribution

*Cons:* Time complexity can be high, Preference for balanced partitions

**Harris Corner Detector** *Algorithm:* Compute  $M$  matrix for each image window to get their cornerness scores, 2. Find points whose surrounding window gave large corner response ( $f_t$  threshold), 3. Take the points of local maxima, i.e., perform non-maximum suppression

*Properties:* Rotation invariant, Not scale invariant

**Scale Invariant Feature Transform (SIFT)** *Properties:* Rotation+Scale invariant, Partially Illumination+Camera Viewpoint+Occlusion invariant