UNIT - III

Active Contours

Image segmentation means partitioning the input image, by clustering pixel values of the image. It is mainly used for identifying various sufaces or living or non living objects from an image.

Active Contour is a segmentation method that uses energy forces and constraints to separate the pixels of interest from a picture for further processing and analysis.

Contours are the boundaries that define the region of interst in an image. A contour is a collection of points that have been interpolated (insert into something else).

The interpolation proedure might be linear, Splines, or polynomial, depending on how the curve in the image is described.

The use is to define smooth shapes in images and to construct closed contours for regions.

It is mainly used to identify uneven shapes in images.

Contour models define the object borders or other picture features to generate a parametric curve or contour. The energy fn is always related to the image's curve.

External energy is described as the sum of forces caused by the picture that is specifically used to control the location of the contour on the image, and internal energy, which is used to govern deformal (capable of being reshaped) changes.

The contour segmentation constraints for a certain image are determined by the needs. The desired shape is obtained by defining the energy function.

A collection of points that locate a contour is used to describe contour deformation. This shape corresponds to the desired image contour, which was defined by minimizing the energy function.

Snake Model.

The function of snake model is to identify and outline the target object for segmentation. It requires some prior knowledge of the target objects shape, especially for complicated things.

Snake models are configured by the use of spline focused on minimizing energy, followed by various forces governing the image.

Equation

A simple snake model can be denoted by a set of n points, V_i for $i = 0 \ldots n-1$, the internal elastic energy term $E_{internal}$ and external edge based energy term $E_{external}$. The internal energy term's aim is to regulate snake's deformations, while the exterior energy term's function is to control the contour's fitting onto the image.

The external energy is a combination of forces caused by the picture E_{image} and contraint forces imposed by the user E_{con}.

Course Code: Disa

the snake is nothing but a vector of (x,y) points with some constraints, its final goal is to surround the object and describe its shape (contour) and then to track or represent the object by its shape.

Internal energy is the average of pixel intensities inside the curve and external energy demonstrates the average of pixel intensities outside the curve.

$$E^{*}_{snake} = \int_{s=0}^{1} E_{snake}(v(s))\,ds$$

$$= \int_{s=0}^{1} E_{internal}(v(s)) + E_{image}(v(s)) + E_{con}(v(s))\,ds$$

$v(s)$ → set of x and y co-ordinate of the points in the snake $s \in [0,1]$

Advantages:

In the field of medical imaging the snake model is used to segment one portion of an image that has unique characteristics when compared to other regions of the picture.

Disadvantage.

- Noise sensitivity
- erroneous contour detection in high complexity objects.

Internal energy regulates the shape of contour, controlling its curvature, shape regulatory etc.

Internal forces help to smooth the data set, and external forces push the contours near to the region of interest.

Eint → controls the natural behavious of the snake and hence the arrangement of the snake points.

Eimage – attracts the snake to chosen low level features (such as edge points)

Econ – allows higher level information to control the snake's evolution.

The aim of the snake is to evolve by minimizing the equation.

Dynamic Snakes.

Internal image energy is defined to be a weighted summation of first-and second order derivates aroud the contour.

$$E_{int} = \alpha(s)\left|\frac{dv(s)}{ds}\right|^2 + \beta(s)\left|\frac{d^2v(s)}{ds^2}\right|^2$$

The first order differential, $dv(s)/ds$, measures the energy due to stretching which is the elastic energy since high values of this differential imply a high rate of change in that region of the contour.

The second order differential $d^2v(s)/ds^2$ measures the energy due to bending, the curvature energy.

$\alpha(s)$ controls the contribution of the elastic energy due to point spacing.

Choice of the values for α an β controls the shape of snake aims to attain.

Low values for α imply the points can change in spacing greatly, whereas higher values imply that the snake aims to attain evenly spaced contour points.

Low values of β imply that curvature is not minimized and the contour can form corners in its perimeter, whereas high values predispose (inclined to a specified attitude/action or condition) the snake to smooth contour.

The image energy attracts the snake to low level features such as brightness or edge data.

Lines, edges and terminations could contribute image energy.

$$E_{image} = w_{line} E_{line} + w_{edge} E_{edge} + w_{term} E_{term}$$

The line energy can be set to the image intensity at a particular point. If black has a lower value than white, then the snake will be extracted to dark features. Altering the sign of w_{line} will attract the snake to brighter feature.

The edge energy can be computed by application of an edge detection operator, the magnitude, say, of the output of the Sobel edge detection operator

The termination energy can include the curvature of level image contours.

Dynamic Snakes and Condensation

The condensation algorithm (conditional Density propagation) application is to detect and track the contour of objects moving in a cluttered (lot of disorganized stuff in one place) environment.

It does not compute on every pixel of the image, rather pixels to process are chosen at random, and only a subset of the pixels end up being processed.

The condensation algorithm seeks to solve the problem of estimating the conformation of an object described by

a vector x_t at time t. given observations $z_1, \ldots, z_t$, of the detected features in the images up to and including the current time.

The algorithm outputs an estimate to the state conditional probability density $P(x_t | z_1 \ldots, z_t)$ by applying a nonlinear filter based on factored sampling.

The conditional density of the object at the current time $P(x_t | z_1 \ldots z_t)$ is estimated as a weighted, time-indexed sample set $\{s_t^{(n)}, n = 1, \ldots N)$ with weight $\pi_t^{(n)}$. N is a parameter determining the number of sample set chosen.

Iterative Procedure.

sample with replacement N times from the set $\{s_0^{(n)}, n = 1 \ldots, N\}$ with probability $\{\pi_0^{(n)}, n = 1 \ldots N\}$ to generate a realization of $P(x_t | z_1 \ldots z_t)$

2. Apply the learned dynamics $P(x_t|x_{t-1})$ to each element of this new set, to generak a new set $\{s_t^{(n)}\}$.

3. To Take into account the current observation z_t, set $\pi_t^{(n)} = \frac{P z_t|s^{(n)}}{\sum_{j=1}^{N} P(z_t|s^{(j)})}$ for each element $\{s_t^{(n)}\}$.

This algorithm outputs the probability distribution $P(x_t|z_1 \cdots z_t)$ which can be directly used to calculate the mean position of the tracked object, as well as the other moments (shape) of the tracked object

SCISSORS

It is the "live-wire" path selection tool. The live-wire tool allows the user to interactively select the desired optimal path from the entire collection of optimal paths (one for each pixel of the image) generated from a specified seed point.

The optimal path from each pixel is determined at interactive speeds by computing an optimal spanning tree of the image using an efficient implementation of Dijkstra's graph searching algorithm.

The basic idea is to formulate the image as a weighted graph where pixels represent nodes with directed, weighted edges connecting each pixel with its adjacent neighbours.

Local cost

If p and q are 2 neighboring pixels in the image then $l(p,q)$ represents

the local cost on the directed link(edge) from p to q.

The local cost function is a weighted sum of component cost functions on each of the following image features.

Image feature	Formulation
Laplacian Zero-Crossing	f_Z
Gradient Magnitude	f_G
Gradient direction	f_D
Edge pixel value	f_P
Inside pixel value	f_I
Outside pixel value	f_O

Combining these feature components into a local cost function gives:

$$l(p,q) = w_Z \cdot f_Z(q) + w_G \cdot f_G(q) + w_D \cdot f_D(p,q) + w_P \cdot f_P(q) + w_I \cdot f_I(q) + w_O \cdot f_O(q).$$

where each w is the weight of the corresponding feature function. Empirically weights of $w_Z = 0.3$ $w_G = 0.3$ $w_D = 0.1$ $w_P = 0.1$ $w_I = 0.1$ & $w_O = 0.1$

Graph Search algorithm.

The graph search algorithm is initialized by Placing a start or seed point s, with a cumulative cost of 0, on an empty list, L (active list). A point P, is placed on the active list in sorted order based on its total or cumulative cost, g(P).

All other points in the image are initialized with infinite cost. After initialization the graph search then iteratively generates a minimum cost spanning tree of the image, based on the local cost function L

In each iteration, the point or pixel P with the minimum cumulative cost is removed from L and 'expanded' by computing the total cost to each of P's unexpanded neighbours. For each neighbor q of P, the cumulative cost to q

is the sum of the total cost to p plus the local link cost from p to q that is

$$g_{tmp} = g(p) + l'(p,q)$$

If the newly computed total cost to q is less that the previous cost (ie $g_{tmp} < g(q)$) then g(q) is assigned a new, lower cumulative cost and an optimal point pointer is set from q back to p.

After computing the cumulative cost to p's unexpanded neighbors and setting any necessary optimal path pointers, p is marked as expanded and the process repeats until all image pixels have been expanded.

Live-wire

Once the optimal path pointers are generated, a desired boundary segment can be chosen dynamically via a "free" point. Interactive movement of the free point by the mouse cursor causes the boundary to behave like a live-wire as it adapts the minimum cost path.

Level sets.

The limitation of snakes, is if the shape changes dramatically, curve reparameterization may also be required. based on Parametric curve of region f(s)

An alternative representation for such closed contours is to use a level set, where the Zero crossing (It is a point where a sign of a mathematical function changes from positive to negative, zero value) of a characteristics or signed distance function define the curve. zero crossing

Level sets evolve to fit and track objects by modifying the underlying embedding function (2D function) $\phi(x,y)$ instead of the curve $f(s)$.

To reduce the amount of computation required, only a small strip (frontier line) around the locations of the current zero crossing needs to updated at each step, which results in what are called fast marching methods.

split and merge.

Used to segment an image. The image is successively split into quadrants based on a homogeneity (same kind) criterian and similar regions are merged to create the segmented result. This technique incorporates a quadtree (is a tree data structure in which each internal nodes has exactly four children) data structure, meaning that there is a parent child node relationship. The total region is a parent and each of the four splits is a child.

Algorithm

1. Define the criterion to be used for homogeneity.
2. Split the image into equal size regions.
3) Calculate homogeneity for each region
4) If the region is homogeneous, then merge it with neighbor.

5. The process is repeated until all regions pass the homogeneity test.

Homogeneity.

. Ways to define homogeneity.

1) **Uniformity** – the region is homogeneous if its gray scale levels are constant or within a given threshold.

2) **Local mean vs global mean** – if the mean of a region is greater than the mean of the global image, then the region is homogeneous.

3. **Variance** – the gray level variance is defined as (black to white, ie) amount of light (ie) it carries only intensity information.

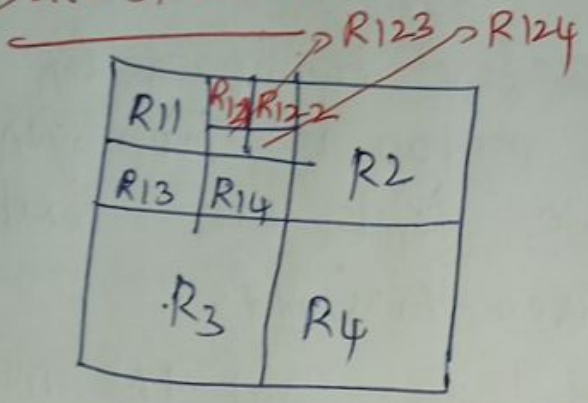
$$\sigma^2 = (1/(N-1)) \sum_{(r,c)\in R} [I(r,c) - \bar{I}]^2$$

r and c are row & column, N is the no. of pixels in the region and

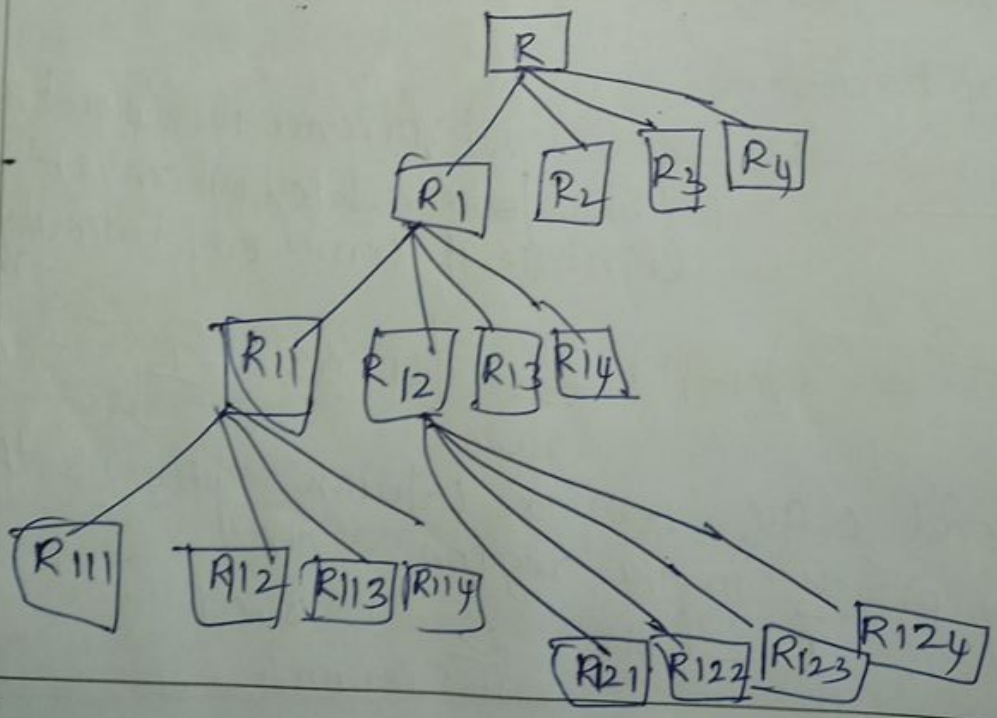
$$\bar{I} = (1/N) \sum_{r,c \in Region} I(r,c)$$

Variance of a region be less than a specified value in order to be considered homogeneous.

Data structure.

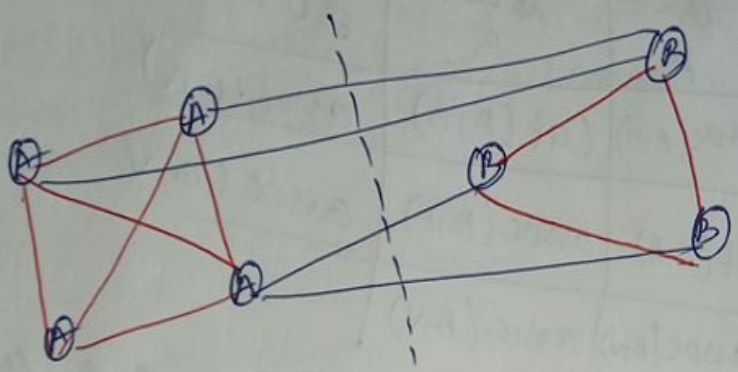


Each level of partitioning can be represented in a tree-like structure.



Normalized cuts

: Examines the affinities (similarities) b/w nearby pixels and tries to separate groups that are connected by weak affinities.



The pixels in group A are all strongly connected with high affinities, as are the pixels in Gp B. The connections b/w these groups are much weaker. A normalized cut b/w the 2 gps, shown in dashed line, separates them into 2 clusters.

The cut b/w 2 gps A & B is defined as the sum of all the weights being cut.

$$Cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

weights b/w 2 pixels (or regions) i and j measure their similarity. Using a minimum cut as a segmentation criterion, does not result in ~~segmentation~~ reasonable clusters, since the smallest cuts usually involve isolating a single pixel.

measure of segmentation is the Cut which is defined as

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

	A	B	sum
A	assoc(A,A)	cut(A,B)	assoc(A,V)
B	cut(B,A)	assoc(B,B)	assoc(B,V)
sum	assoc(A,V)	assoc(B,V)	

The assoc and cut entries are computed as area sums of the associated weight matrix W. Normalizing the table entries by the row or column sum produces normalized associations and cuts Nassoc and Ncut.

In equation

$$assoc(A,A) = \sum_{i \in A, j \in A} w_{ij}$$

is the association (sum of all the weights) within a cluster and assoc(A,V) = assoc(A,A) + cut(A,B) is the sum of all the weights associated with nodes in A.

The table shows how the cuts and associations can be thought of as area sums in the weight matrix $W = (w_{ij})$, where the

where $N = D^{-1/2} W D^{-1/2}$ is the normalized affinity matrix. $z = D^{1/2} y$. Because these eigenvectors can be interpreted as the Large modes of vibration in a spring-mass system, normalized cuts is an example of a spectral method for image segmentation.

Graph cuts and energy based methods.

A common theme in image segmentation algorithms is the desire to group pixels that have similar appearance (statistics) and to have the boundaries b/w pixels in different regions be of short length and across visible discontinuities.

If we restrict the boundary measurements to be b/w immediate neighbors and compute region membership statistics by summing over pixels we can formulate this as a classic pixel based energy function using either a variational formulation.

An example of a discrete labeling problem that combines both region-based and boundary based energy terms is, using minimum description length (MDC) coding

entries of the matrix have been arr. so that the nodes in A come first and nods in B come second

Computing a real valued assignment of nodes to gps.

Let x be the indicator vector where $x_i = +1$ iff $i \in A$ and $x_i = -1$ iff $i \in B$.
Let $d = W1$ be the row sums of the symmetric matrix W and $D = diag(d)$ be the corresponding diagonal matrix.

Minimizing the normalized cut over all possible indicator vectors x is equivalent to minimizing

$$\min_y \frac{y^T (D - W) y}{y^T D y}$$

$y = (1+x) - b(1-x)/2$ is a vector consisting of all 1s and -bs such that $y \cdot d = 0$.

Minimizing the Rayleigh quotient is equivalent to solving the Generalized eigenvalue system $(D - W)y = \lambda D y$ which can be turned into a regular eigenvalue problem $(I - N)z = \lambda z$.

to derive the energy function being minimized

The energy corresponding to a segmentation problem can be written as

$$E(f) = \sum_{i,j} E_r(i,j) + E_b(i,j)$$

where the region term

$$E_r(i,j) = E_s(I(i,j); R(f(i,j)))$$

is the negative log likelihood that pixel intensity (or color) $I(i,j)$ is consistent with the statistics of region $R(f(i,j))$ and the boundary term

$$E_b(i,j) = s_x(i,j)\,\delta(f(i,j) - f(i+1,j)) + s_y(i,j)\,\delta(f(i,j) - f(i,j+1))$$

measure the inconsistency b/w N_4 neighbors modulated by local horizontal and vertical smoothness terms $s_x(i,j)$ and $s_y(i,j)$

Region statistics can be something as simple as the mean gray level or color in which case

$$E_s(I; \mu_k) = \|I - \mu_k\|^2.$$

For Smoothness (boundary) terms it is common to make the strength of the smoothness $s_x(i,j)$ inversely proportional to the local edge strength.

Energy based segmentation problems were optimized using iterative gradient techniques, which were slow and prone to getting trapped in local minima.

Next is applying binary MRF optimization alg to binary object segmentation.

In this approach the user first delineates pixels in the background and foreground regions using a few strokes of an image brush. These pixels then become the seeds that tie nodes in the S-T graph to the source and sink labels S and T. Seed pixels can also be used to estimate foreground and background region statistics (intensity or color histograms).

The capacities of the other edges in the graph are derived from the region and boundary energy terms. (ie) pixels that are more compatible with the foreground or background region get stronger connections to the respective source or sink. adjacent pixels with greater smoothness also get stronger links.

Once the minimum cut/maximum flow problem has been solved using a polynomial time algorithm, pixels on either side of the computed cut are labeled according to the source or sink to which they remain connected.

Another extension of binary segmentation alg is Grabcut. It iteratively re-estimates the region statistics, which are modeled as a mixtures of Gaussians in color space. This allows their system to operate given minimal user input, such as a single bounding box. the background color model is initialized from a strip of pixels

around the box outline. The foreground color model is initialized from interior pixels by quickly converges a better estimate of the object.

The user can also place additional strokes to refine the segmentation as the solution progresses.

Another major extension to the original binary segmentation formulation is the addition of directed edges, which allows boundary regions to be oriented. e.g to prefer light to dark transitions or vice versa.

2D and 3D feature-based alignment

Once we have extracted features from images, the next stage in many vision algorithms is to match these features across different images.

An important component of this matching is to verify whether

converges

the set of matching features is geometrically consistent, (e.g) whether the feature displacements can be described by a simple 2D or 3D geometric transformation.

The computed motions can then be used in other applications such as image stitching or augmented reality.

1) 2D and 3D feature based alignment.

Feature based alignment is the pbm of estimating the motion b/w 2 or more sets of matched 2D or 3D points.

2D alignment using least squares.

Given a set of matched feature points $\{(x_i, x_i')\}$ and a planar parametric transformation of the form

$$x' = f(x; p)$$

Transform	Matrix	Parameters P	$\frac{2}{3}$
1. translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	t_x, t_y	$\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$

2. Euclidean $\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$ (t_x, t_y, θ) $\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$

3. Similarity $\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$ (t_x, t_y, a, b) $\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$

4. affine $\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$ $\begin{pmatrix} t_x, t_y, a_{00}, a_{01} \\ a_{10}, a_{11} \end{pmatrix}$ $\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$

5. projective $\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$ $(h_{00}, h_{01} \cdots h_{21})$

To produce the best estimate of the motion parameters P is to use least squares. (ie) to minimize the sum of squared residuals.

$$E_{LS} = \sum_i \|r_i\|^2 = \sum \|f(x_i; P) - x_i'\|^2$$

where

$$r_i = f(x_i; P) \Rightarrow x_i' = \hat{x}_i' - \bar{x}_i'$$

is the residual b/w the measured location $\hat{x}_i'$ and its corresponding current predicted location $\bar{x}_i' = f(x_i; P)$

Many of the motion models like translation, similarity and affine, have a linear relationship b/w the amount of motion $\Delta x = x' - x$ and the unknown parameters P,

$$\Delta x = x' - x = J(x) P.$$

where $J = \partial f / \partial P$ is the Jacobian of the transformation f with respect to the motion parameters P.

A simple linear regression (linear least squares problem) can be formulated as

$$E_{LS} = \sum_i \| J(x_i) P - \Delta x_i \|^2$$

$$= P^T \left[\sum_i J^T(x_i) J(x_i) \right] P - 2 P^T \left[\sum_i J^T(x_i) \Delta x_i \right] + \sum_i \| \Delta x_i \|^2$$

$$= P^T A P - 2 P^T b + c$$

The minimum can be found by solving the symmetric positive definite (SPD) system of normal equations

$$A p = b,$$

where

$$A = \sum_i J^T(x_i) J(x_i)$$

is called the Hessian and $b = \sum_i J^T(x_i) \Delta x_i$

Pose estimation

A particular instance of feature based alignment, which occurs often, is estimating an object's 3D pose from a set of 2D point projections. The Pose estimation problem is also known as extrinsic calibration, as opposed to the intrinsic calibration of internal camera parameters such as focal length.

The problem of recovering pose from three correspondences, which is the minimal amount of information necessary is known as the Perspective 3-point problem (P3P)

with extensions to large numbers of points Collectively known as PnP.

1. Linear algorithms -

The simplest way to recover the Pose of the Camera is to form a set of linear equations alalogous to those used for 2D motion estimation from the Camera matrix form of perspective projection

$$x_i = \frac{P_{00}X_i + P_{01}Y_i + P_{02}Z_i + P_{03}}{P_{20}X_i + P_{21}Y_i + P_{22}Z_i + P_{23}}$$

$$y_i = \frac{P_{10}X_i + P_{11}Y_i + P_{12}Z_i + P_{13}}{P_{20}X_i + P_{21}Y_i + P_{22}Z_i + P_{23}}$$

where (x_i, y_i) are the measure 2D feature locations and (X_i, Y_i, Z_i) are the known 3D feature locations.

In the case where the camera is already calibrated, (ie) the matrix K is known. we can perform pose

estimation using as few as three points.

The basic observation that these linear PnP (Perspective n-Point) algorithms employ is that the visual angle b/w any pair of 2D points $\hat{x}_i$ and $\hat{x}_j$ must be the same as the angle b/w their corresponding 3D points P_i and P_j.

Given a set of corresponding 2D and 3D points $\{(\hat{x}_i, P_i)\}$ where the $\hat{x}_i$ are unit directions obtained by transforming 2D pixel measurements x_i to unit norm 3D directions $\hat{x}_i$ through the inverse calibration matrix

$$\hat{x}_i = N(K^{-1}x_i) = K^{-1}x_i / \|K^{-1}x_i\|.$$

the unknowns are the distances d_i from the camera origin C to the 3D points P_i where

$$P_i = d_i \hat{x}_i + C$$

1) Split & Merge types.

1) Watershed

A Technique related to thresholding. This technique segments an image into several Catchment basins, which are the regions of an image (interpreted as a height field or landscape).

* An efficient way to compute such regions is to start flooding the landscape at all of the local minima and to label ridges wherever differently evolving components meet.

The whole algorithm can be implemented using a priority queue of pixels and breadth first search.

Since images rarely have dark regions separated by lighter ridges, watershed segmentation is usually applied to a smoothed version of the gradient magnitude image, which also makes it usable with color images.

Watershed segmentation associates unique region with each local minimum which can lead to over-segmentation.

Watershed segmentation is therefore often used as part of an interative system, where the user first marks seed locations (with a click or a short stroke) that correspond to the centers of different desired components.

2. Region Splitting (divisive clustering)

* First computes a histogram for the whole image and then finds a threshold that best separates the large peaks in the histogram. This process is repeated until regions are either fairly uniform or below a certain size.

3. Region merging (agglomerative clustering)

Pixel based merging combines adjacent regions whose average color difference is below a threshold or whose regions are too small

superpixels Segmenting the image into such which are not semantically meaningful

can be a useful pre-processing stage to make higher-level algorithms such as stereo matching, optic flow and recognition.

4) Graph based segmentation.

Merging algorithm that uses relative dissimilarities between regions to determine which ones should be merged. They start with a pixel-to-pixel dissimilarity measure $w(e)$ that measures for example, intensity differences between N_8 neighbors.

For any region R, its internal difference is defined as the largest edge weight in the region's minimum spanning tree

$$Int(R) = \min_{e \in MST(R)} w(e)$$

For any 2 adjacent regions with at least one edge connecting their vertices, the difference between these

regions is defined as the minimum weight edge connecting the 2 regions

$$Dif(R_1, R_2) = \min_{e=(v_1,v_2) \mid v_1 \in R_1, v_2 \in R_2} w(e)$$

The algorithm merges any 2 adjacent regions whose difference is smaller than the minimum internal difference of these 2 regions. Kruskal's minimum spanning Tree alg is used.