(4). How do panaromas are recognised? Explain various methods for recognising Panaromas? (U5)

A. → Panaromas are created when img is captured in elongated view.

→ Basically it is the stitching of multiple imgs together. imgs in sequence, first & last img to be stitched (topology inference)

→ A 360° photo is a controllable panaromic img that surrounds the original pt from which shot was taken. (jump from one to other, to create end-to-end overlaps).

Methods:

⇒ Bundle Adjustment:

* register a large no. of imgs is to add new imgs to the panorama one at a time.
* aligning the most imgs already in collection.
* In case of 360°, accumulated error lead to a gap.
* fixed by stretching the alignment of all imgs using (gap closing).
* process of simultaneously adjusting pose parameters for a large collection of overlapping imgs called Bundle Adjustment.

⇒ Feature Matching: [Feature-Based Method]

* Corresponding features from 2 imgs.
* One is source img. Other one is target img.

⇒ Img Matching.

* process of bringing two imgs geometrically into so that pixels of the 2 imgs correspond to same physical.
* Harris corner.
* SIFT (Scale Invariant feature transform)
  ↓
  Detecting local features.

(12) Fourier Transform Alignment: (U4)

* fourier Based Alignment refers to fact that the fourier transform of shifted signal has the same magnitude as original signal but a linearly varying phase.

$$F\{I_1\{x+u\}\} = F\{I_1\{x\}\}e^{-ju.w} = I_1(w)e^{-ju.w}$$

→ w → vector valued Angular frequency.

* Another useful property.
  → Convolution in the Spatial domain to Multiplication in fourier Domain.

* mainly used in the computation of img correlation.
* fourier Based Alignment is mainly used when hierarchical approach fails.
* The fast fourier transform Algo can compute the transform of N×M img in O(NM log NM) op.
* It can also be used to accelerate the sum of squared differences.

⇒ windowed correlation:
* fourier can be performed for many pixels But windowed correlation can be performed even for single pixel.
* AKA ⇒ wighted correlation
* cross-correlation func shd be replaced with windowed cross-correlation func.

$$E_{wcc}(u) = [w_0(x)\, I_0(x)] \bar{*} [w_1(x)\, I_1(x)]$$

$w_0, w_1 \rightarrow$ zero outside the valid range.

⇒ phase correlation.
* Here spectrum of 2 Matching signals are matched & whitened.
* It is noiseless & has perfect cyclic shift.
* perfect picture ⇒ low noise, effective whitening process.

$$\mathcal{F}\{E_{pc}(u)\} = \frac{I_0(\omega) I_1^*(\omega)}{\|I_0(\omega)\| \|I_1(\omega)\|}$$

$$\mathcal{F}\{E_{pc}(u)\} = e^{-2\pi j u \cdot \omega}$$ [noiseless signals with perfect cyclic shift]

=> Rotation & Scales.

* fourier Based Alignment is mostly used to estimate translational shift bet imgs, under Limited Conditions.

* Also use to estimate in-plane rotations & scales.

Consider 2 imgs purely by rotation

$$I_1(\hat{R}x) = I_0(x).$$

=> polar coordinates.

$\bar{I}_0(r,\theta) = I_0(r\cos\theta, r\sin\theta)$

$\bar{I}_1(r,\theta) = I_1(r\cos\theta, r\sin\theta$

$I_0(r,\theta)$ (FFT)

=> log - polar coordinates

$\bar{I}_0(s,\theta) = I_0(e^s\cos\theta, e^s\sin\theta)$

$\bar{I}_1(s,\theta) = I_1(e^s\cos\theta, e^s\sin\theta)$

$I_0(s,\theta)$

-> by a Scale.

$I_0(x)$

$I_0(x)$.

* Both imgs converted to fourier Domain, Only magnitude are retained

14. Normalized Cuts in Detail.

U3.

* Normalized cuts tech introduced by Shi & Malik examines the affinities bet nearby pixels & tries to separate grps that are connected by weak Affinities.

* Consider a simple graph. The pixels in group A are All strongly connected with high Affinities shown as (thick red lines) as are in pixels grps.

* Connections bet these two grps, (thinner blue lines) are much weaker.

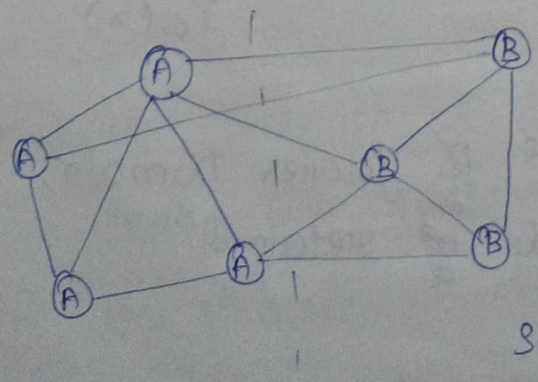
* A normalized cut bet 2 grps. (Dashed line) separates them into clusters.

* The cut bet 2 grp A & B is Defined as the sum of all weights being cut

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

* weights bet 2 pixels. i & j measure their similarity

⇒ Smallest cut ⇒ isolating a single pixel



	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)	

$\Rightarrow$ Normalised cut $= \dfrac{\text{Cut}(A,B)}{\text{assoc}(A,V)} + \dfrac{\text{Cut}(A,B)}{\text{assoc}(B,V)}$

$\Rightarrow$ $\text{assoc}(A,V) = \text{assoc}(A,A) + \text{Cut}(A,B)$

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

$w = [w_{ij}] \Rightarrow$ sum in weight Matrix

$\Rightarrow$ nodes in A come 1st, then B, $\Rightarrow$ sums can be computed.

* These normalised values better reflect fitness of a particular segmentation, since they look for collection of edges that are weak relative to all of the edges both inside & emanating from particular Region.

* Computing the normalised cut is NP-complete.

Let x be indicator vector.

$x_i = +1 \; ; i \in A$

$x_i = -1 \; ; i \in B$.

$d = W1 \quad \Rightarrow D = \text{diag}(d)$.

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

Normalized Affinity Matrix: $N = D^{-1/2} W D^{-1/2}$

13 a) Snakes in Detail.

U3 Drawbacks in using Snakes.

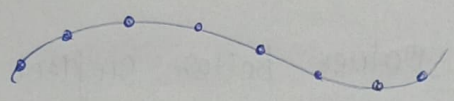
b) Basic Solution to overcome while using Snakes.

A) * The Snakes in CV are also called as Active Contours

* It works like Stretched elastic Band being released.

* It is to Identify & Outline the target Obj.

* Snakes are Configured by use of Spline



* used in medical imaging.

* energy minimizing.

* img Based potential. $\left[E_{img} = w_{line} E_{line} + w_{edge} E_{edge} + w_{term} E_{term} \right]$

* two Dimensional Spline Curve.

$$f(s) = (x(s), y(s))$$

* Major Drawback ⇒ noise Sensitivity

B) * We can use Splines & Shape priors.

→ AKA → B - Spline approximations.

* The Resulting B - Snake can be written as

$$F = Bx$$

$$F = \begin{bmatrix} f^T(0) \\ \vdots \\ f^T(N) \end{bmatrix} \qquad B = \begin{bmatrix} B_0(s_0) & \cdots & B_K(s_0) \\ \vdots & & \vdots \\ B_0(s_N) & \cdots & B_K(s_N) \end{bmatrix}$$

$$X = \begin{bmatrix} x^T(0) \\ \vdots \\ x^T(K) \end{bmatrix}$$

10. Hierarchical motion estimation. (U4)

A * It is a process where an img pyramid is constructed & a search over a smaller no. of discrete pixels is performed.

* The Motion estimate from one level of the pyramid is then used to initialise a smaller local search for next level.

$$I_K(x_i) \leftarrow I_K^{(l-1)}(2x_j)$$

* At the coarest level, we search for best Displacement u & that minimises the diff bet. imgs I_0 & I_1.

* One suitable vector is used to predict the displacement.

* The search over displacement then repeated.

* Alternatively one of the img can be warped by current motion estimate, in which case only small incremental motions need to be computed at finer level.

7. (i) BDRF -> Bidirectional reflactance Distribution function

(ii) with neat Dia. Discuss components into which a typical BDRF is split into ?

A(i) * BDRF => used for Scattering of light.

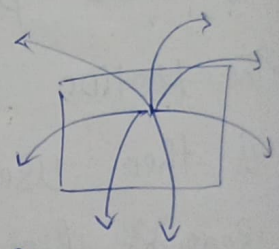
* It has 4 function namely.

$R(a, b, c, d, \lambda)$

each has a separate wavelength in it.

* It has reciprocal formula & that formula is Helmboltz reciprocity.

$R(\hat{a}, \hat{b}, \hat{c}, \hat{d}, \lambda)$



* Scattering of light is isotrophic

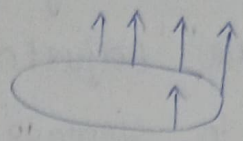
* It will not be in perfect Direction, so, it is isotrophic

(ii) * Basically BRDF can often be Split into their Diffuse & Specular Components

=> Diffuse Component:

* The diffuse Component scatters light uniformly in all Directions mostly associate with shading.

* It often imparts a strong body color since scattering is uniform in all directions. BRDF is const



=) Specular Reflection:

* Depends on Direction of outgoing light.
* Incident Light rays are reflected in direction that is rotated by 180° around the surface normal.

U5 1. What are 2D Motion Models? Explain how they can be used for img stitching?

A * 2D Motion Models are Models that are used to establish mathematical relationships from one img to another for registering & aligning

* Img Stitching Algo for aligning imgs & stitching them into seamless photo-mosaics are among the oldest technique.

* They are used to create high resolution photo-mosaics to produce today's Digital maps & Satellite photo's.

* Also used to create wide angle panaromas.

* Img stitching comes under photogrammatery
* Img stitching acts as a base for Bundle Adjustments
* Used to remove ghosts & parallax.
* Img stitching is faster than older techniques like pixel-to-pixel matching.
  Sparse set of features.

=> STEPS:

proper mathematical model

↓

Pixel to pixel matching using gradient

↓

finally, Aligning imgs

16. Categorise the various technique Developed for Solving pose estimation? (U3)

A. Linear Algo:

Simplest way to recover pose of the camera is to form a set of Linear equations analogous to those used for 2D-Motion

$$x_i = \frac{P_{00} x_i + P_{01} y_i + P_{02} z_i + P_{03}}{P_{20} x_i + P_{21} y_i + P_{22} z_i + P_{23}}$$

$$y_i = \frac{P_{10} x_i + P_{11} y_i + P_{12} z_i + P_{13}}{P_{20} x_i + P_{21} y_i + P_{22} z_i + P_{23}}.$$

Pose Estimation:

* Obj's 3D poso from a set of 2D pts projection.
* extrinsic calibration (opposed to intrinsic)
* prob recovery → perspective - 3 pt prob (P^3P)
* large pts collectively → PnP.

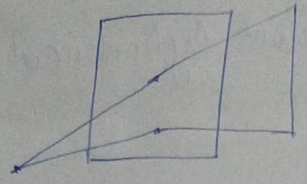
⇒ Linear Algo set of linear eq.

⇒ iterative Algo. accurate & flexible.

is a C.V tech.

found out by key pts.

connection bet 2 pts ⇒ pair.



iterative Algo:

Most Accurate way to estimate pose is to Directly minimise the squared reprojection error for the 2D pts as a func of the unknown pose parameters

$$\Sigma\left(\frac{\delta f}{\delta R}\Delta R + \frac{\delta f}{\delta t}\Delta t + \frac{\delta f}{\delta K}\Delta K - r_i\right)$$

2. Explain the Basic Steps involved in choosing a Composting Surface. (U5)

A. * Compositing is process of getting finally Stitched mosaic img.

* This involves Selecting a final Composting Surface (flat, cylindrical, Spherical etc). & reference img [reference into reference coordinate sys]

* The final Step is to how Optimally bend these pixels to Minimize visible seams, blur ghosting

⇒ Choosing a Compositing Surface:

* The first choice is how to represent img.

* If only few pictures are there, Select a reference img from one of those.

* for few imgs it is Sometimes called as flat ~~Panorama~~ Panorama.

* for larger view we select cylindrical / spherical projection.

=> View Selection

* which part of the scene will be centered in the final view.
* In case of smaller img → one of those pics acts as reference. (geometrically most central).
* In case of larger spherical / cylindrical projection taken

case of 360° => choose the middle img from sequence of i/p

=> Coordinate Transformation:

* next step is mapping of coordinates bet I/p & O/p pixels.
* If final compositing surface is flat, I/p img have no radial distortion.
* If final compositing surface is cylindrical or spherical then I/p has radial distortion.

=> Sampling issues

* Sometimes if all the above said process yield the crt pixel address there may be a problem of low resolution.
* We can use HIP mapping to crt that in order to get high quality imgs.

produce img with higher resolution } super-resolution

19. Explain the various techniques in edge Detection with necessary. (U2)

A. * Edge Detection is a crucial approach in CV

* It refers to a set of Mathematical technique for Detecting edges or curves in a pic when brightness of img abruptly changes or more formal has discontinuities.

* Mathematical way to Define the Slope & direction of a surface $J(x) = \nabla I(x)$

→ J → Direction of Steepest ascent.

* In Application where the accuracy of the edge Orientation is more important, higher order Steerable filters can be used.

* Such filters are more selective for more elongated edge.

* Only disadvantage is more expensive

→ Techniques used:

⇒ Scale selection & blur estimation.

* here the scale parameter is σ.

* given a known img noise level, this tech computes, for every pixel, the minimum scale at which an edge can be reliable detected.

⇒ Color edge Detection

* While most edge Detection techniques have been Developed for grayscale imgs.
* Color imgs provides additional info.
* 3D - color histograms can be used to compare regional statistics & that additional measures such as texture can be considered.

⇒ Combining edge feature cues:

* Combining multiple low-level cue such as brightness, color & texture.
* Separate construction & training is done for the difference in brightness, color & texture.
* outputs of these are combined using various M.L tech.

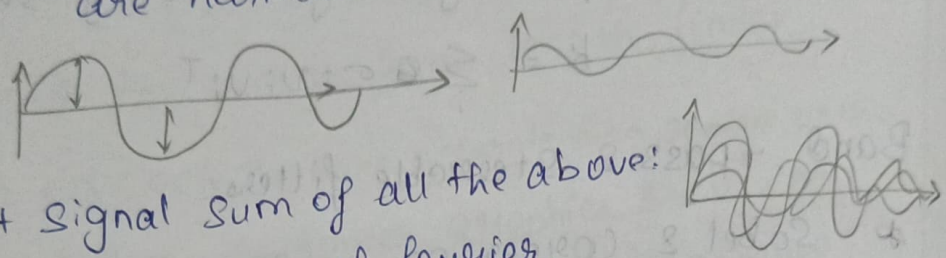
Fourier transforms:

* States that periodic signals can be represented into sum of sines & cosines when multiplied with certain weights.

* Properties:

→ Signals are sines & cosines

→ " are harmonic each other.



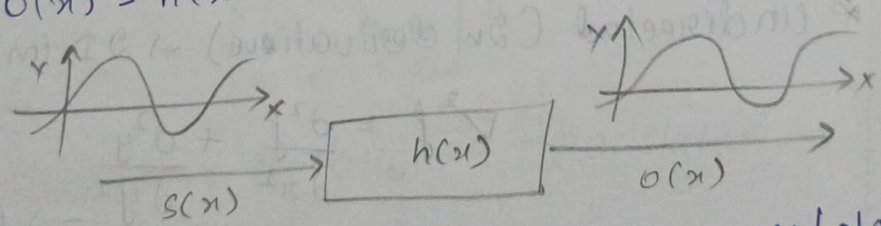
* last signal sum of all the above: this is the idea of fourier.

* $S(x) = \sin(2\pi f x + \phi_i) = \sin(\omega x + \phi_i)$

* Sinusoidal signal $S(x)$ with filter → impulse response $h(x)$.

Same freq → magnitude & phase diff.

$O(x) = h(x) * S(x) = A \sin(\omega x + \phi_0)$.



* new magnitude → gain; phase diff → shift / phase.

* tabulation of magnitude & phase at each freq.

* One-dimenstional ⇒ $H(\omega) = \int_{-\infty}^{\infty} h(x)\, e^{-j\omega x}\, dx$.

Discrete
DFT → $O(N^2)$ OP

fast FFT → $O(N \log_2 N)$ OP.

⇒ 2-D FT.

* just specifying horizontal / vertical freq.

$$s(x, y) = \sin(\omega_x x + \omega_y y)$$

$$H(\omega_x, \omega_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x, y) e^{-j(\omega_x x + \omega_y y)} dx\, dy.$$

FEATURE DETECTORS:

* Patches with large contrast changes are easier to localize.
* Straight line segments at a single orientation suffer from the aperture problem.
* possible to align patches along the direction normal to edge direction.

Feature Detectors:

* Simplest possible matching criterion for comparing two img patches.

$$E_{WSSD}(u) = \sum_i w(x_i)[I_1(x_i+u) - I_0(x_i)]^2 \rightarrow (1)$$

Weighted Sum Sq. Diff

* $I_0, I_1 \rightarrow$ imgs being compared.
* $u = (u, v) \rightarrow$ displacement vector.
* $w(x) \rightarrow$ spatially varying weighting.
* Summation $i \rightarrow$ overall pixels in patch
* Auto-correlation : compute how stable the metric is with respect to small variation in position Δu

$$E_{AC}(\Delta u) = \sum_i w(x_i)[I_0(x_i + \Delta u) - I_0(x_i)]^2 \rightarrow (2)$$

* By Taylor series expansion of img function

$$I_0(x_i + \Delta u) \approx I_0(x_i) + \nabla I_0(x_i) \cdot \Delta u \rightarrow$$

$$E_{AC}(\Delta u) = \sum_i w(x_i)[I_0(x_i + \Delta u) - I_0(x_i)]^2 \rightarrow (3)$$

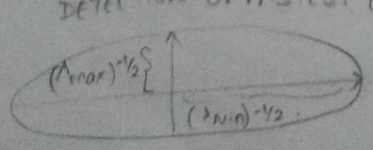
$$= \sum_i w(x_i)[I_0(x_i) + \nabla I_0(x_i) \cdot \Delta u - I_0(x_i)]^2 \rightarrow (4)$$

$$= \sum_i w(x_i)[\nabla I_0(x_i) \cdot \Delta u]^2 \rightarrow (5)$$

$$= \Delta u^T A \Delta u \rightarrow (6)$$

$$\nabla I_0(x_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)(x_i) \rightarrow (7)$$

[img gradient at x]



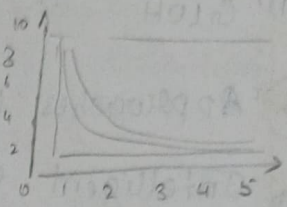
Auto Correlation Matrix : $A = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$

⇒ Förstner - Harris:

* Derived from the auto-correlation Matrix to locate keypoints for the purpose of sparse feature matching.

$$\det(A) - \alpha \, \text{trace}(A^2) = \lambda_0 \lambda_1 - \alpha(\lambda_0 + \lambda_1)^2$$

$\lambda_1 >> \lambda_0$; $\lambda_0 - \alpha \lambda_1$



* Harmonic mean: $\frac{\det A}{\text{tr} A} = \frac{\lambda_0 \lambda_1}{\lambda_0 + \lambda_1}$

$\lambda_0 \approx \lambda_1$ (Smoother func)

* Two eigenvalues are blended to Determine the final interest value.

⇒ Adaptive non-maximal Suppression (ANMS)

* look for local maxima in interest func, lead to uneven distribution of feature pts.
* efficient way to associate suppression radii with all local maxima. by sorting then create a Second list sorted by decreasing suppression radius

⇒ Measuring repeatability:

* Define as the frequency with which keypoints detected in one img are found within ($\epsilon = 1.5$) pixels of corresponding location in transformed img
* Measure the info content available at each detected feature point (entropy of a set of rotationally invariant local grayscale Descriptors)
* Harris operator $\sigma_d = 1$, $\sigma_i = 2$ (Gaussian Derivative)

③ Approaches to locating such Boundary curve in images.

Snakes:

two - ~~Dimention~~ Dimensional generalization of 1D energy-minimizing splines.

Dynamic Snakes & condensation.

linear Dynamic model of shape parameter evolution

Scissors:

User draw a Outline, sys forms a better cu

level sets.

Alternative representation for closed contour

Intelligent Scissors.

* Active contours allows user to roughly specify boundaries. & evolve contour to more accurate loca.
* System optimize the contour in real time. rough outline, sys computes high contrast edges
* To compute optimal curve path (live-wire).
  → first pre-processed to associate low cost with edges.
  (zero crossing, gradient magnitude & orientation)
* user traces a rough curve, recomputes the lowest cost path bet starting seed pt & current Dijkstra's Algo. (breadth-first dynamic Algo) terminates at current target locati
* jumping unpredictable → sys freeze.
* To prevent live wire from freeze, learns intensity profile under current optimized curve
* Watershed region segmentation → Smaller (region Boundaries) graph & meet when nodes are located.
* Instead of recomputing often → snapping.

Level Sets.

* limitation of Active contour → snakes.
* Alternative representation for such closed contours (level sets) → where the zero crossing of a characteristic func defines the curve
* Level set evolve to fit & track by embedding fun
* To reduce computation → fast marching method. (Small strip → location of current zero-crossing)

$$\frac{d\phi}{dt} = |\nabla\phi| \, div\left(g(I) \frac{\nabla\phi}{|\nabla\phi|}\right)$$

$$= g(I) |\nabla\phi| \, div\left(\frac{\nabla\phi}{|\nabla\phi|}\right) + \nabla g(I) \cdot \nabla\phi$$

→ $g(I)$ → generalization version of snake edge potential

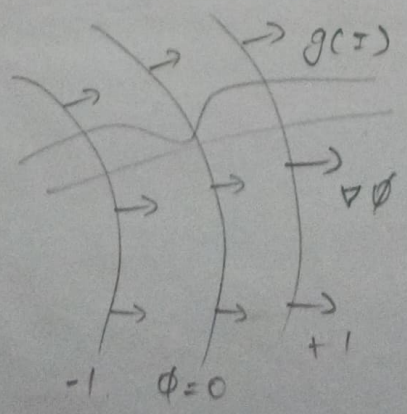
→ ϕ → embedding func. $|\phi| = 1$.

(i) curve in direction of curvature

(ii) curve to migrate towards minima of $g(I)$.

* Change topology.
* local minima ⇒ Based on local measurements (image gradients)

⇒ gray scale image segmentation



2. Explain the Basic Steps involved in choosing a composting surface. (U5)

A- * Compositing is process of getting finally stitched mosaic img.

* This involves selecting a final composting surface (flat, cylindrical, spherical etc). 2 reference img [reference into reference coordinate sys]

* The final step is to how optimally bend these pixels to minimize visible seams, blur ghosting

⇒ Choosing a compositing surface:

* The first choice is how to represent img.

* If only few pictures are there, select a reference img from one of those.

* for few imgs it is sometimes called as flat ~~panorama~~ panorama.

* for larger view, we select cylindrical / spherical projection.

=> View Selection

* which part of the scene will be centered in the final view.
* In case of smaller img → one of those pics acts as reference. (geometrically most central)
* In case of larger spherical / cylindrical projection taken

case of 360° => choose the middle img from sequence of I/p

=> Coordinate Transformation:

* next step is mapping of coordinates bet I/p & O/p pixels.
* If final compositing surface is flat, I/p img have <u>no</u> <u>radial</u> <u>distortion</u>.
* If final compositing surface is cylindrical or spherical then I/p has <u>radial</u> <u>distortion</u>.

=> Sampling issues

* Sometimes if all the above said process yield the crt pixel address there may be a problem of low resolution.
* We can use HIP mapping to crt that in order to get high quality imgs.

produce img with higher resolution } Super-resolution

19. Explain the various techniques in edge Detection with necessary. (U2)

A. * Edge Detection is a crucial approach in CV

* It refers to a set of Mathematical technique for Detecting edges or curves in a pic when brightness of img abruptly changes or more formal has discontinuities.

* Mathematical way to Define the slope & direction of a surface $J(x) = \nabla I(x)$

→ J → Direction of steepest ascent.

* In Application where the accuracy of the edge Orientation is more important, higher order steerable filters can be used.

* Such filters are more selective for more elongated edge.

* Only disadvantage is more expensive

→ Techniques used:

⇒ Scale selection & blur estimation.

* here the scale parameter is σ.

* given a known img noise level, this tech computes, for every pixel, the minimum scale at which an edge can be reliable detected.

⇒ Color edge Detection

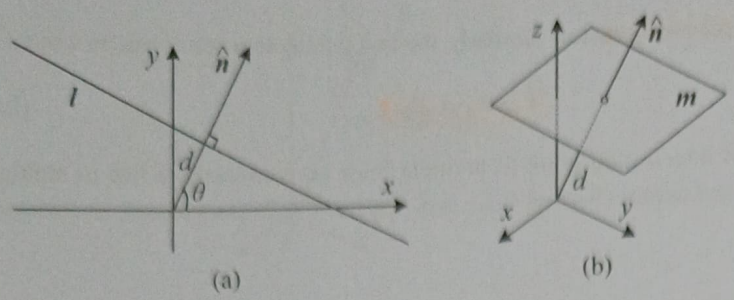
* While most edge Detection techniques have been Developed for grayscale imgs.
* Color imgs provides additional info.
* 3D - color histograms can be used to compare regional statistics & that additional measures such as texture can be considered.

⇒ Combining edge feature cues:

* Combining multiple low-level cue such as brightness, color & texture.
* Separate construction & training is done for the difference in brightness, color & texture.
* Outputs of these are combined using various M.L tech.

2.

FFT	DFT.
* faster version of DFT Algo.	discrete version of FT
* Various fast DFT computation techniques are collectively called as FFT Algo.	Algo that transforms the time Domain to frequency Domain signals.
* Implementation of DFT	relation bet time & frequency domain.
* Applications filtering Algo, Calculating fourier series coefficients	Partial differential applications spectral analysis.



gure 2.2 (a) 2D line equation and (b) 3D plane equation, expressed in terms of the normal $\hat{n}$ and distance to e origin d.

curves (Sections 5.1 and 11.2), surfaces (Section 12.3), and volumes (Section 12.5).

2D points. 2D points (pixel coordinates in an image) can be denoted using a pair of values, $\boldsymbol{x} = (x, y) \in \mathcal{R}^2$, or alternatively,

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}. \tag{2.1}$$

(As stated in the introduction, we use the $(x_1, x_2, \ldots)$ notation to denote column vectors.)

2D points can also be represented using *homogeneous coordinates*, $\tilde{\boldsymbol{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) \in \mathcal{P}^2$, where vectors that differ only by scale are considered to be equivalent. $\mathcal{P}^2 = \mathcal{R}^3 - (0, 0, 0)$ is called the 2D *projective space*.

A homogeneous vector $\tilde{\boldsymbol{x}}$ can be converted back into an *inhomogeneous* vector $\boldsymbol{x}$ by dividing through by the last element $\tilde{w}$, i.e.,

$$\tilde{\boldsymbol{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x, y, 1) = \tilde{w}\bar{\boldsymbol{x}}, \tag{2.2}$$

where $\bar{\boldsymbol{x}} = (x, y, 1)$ is the *augmented vector*. Homogeneous points whose last element is $\tilde{w} = 0$ are called *ideal points* or *points at infinity* and do not have an equivalent inhomogeneous representation.

2D lines. 2D lines can also be represented using homogeneous coordinates $\tilde{\boldsymbol{l}} = (a, b, c)$. The corresponding *line equation* is

$$\bar{\boldsymbol{x}} \cdot \tilde{\boldsymbol{l}} = ax + by + c = 0. \tag{2.3}$$

We can normalize the line equation vector so that $\boldsymbol{l} = (\hat{n}_x, \hat{n}_y, d) = (\hat{\boldsymbol{n}}, d)$ with $\|\hat{\boldsymbol{n}}\| = 1$. In this case, $\hat{\boldsymbol{n}}$ is the *normal vector* perpendicular to the line and d is its distance to the origin (Figure 2.2). (The one exception to this normalization is the *line at infinity* $\tilde{\boldsymbol{l}} = (0, 0, 1)$, which includes all (ideal) points at infinity.)

We can also express $\hat{\boldsymbol{n}}$ as a function of rotation angle θ, $\hat{\boldsymbol{n}} = (\hat{n}_x, \hat{n}_y) = (\cos\theta, \sin\theta)$ (Figure 2.2a). This representation is commonly used in the *Hough transform* line-finding algorithm, which is discussed in Section 4.3.2. The combination (θ, d) is also known as *polar coordinates*.

When using homogeneous coordinates, we can compute the intersection of two lines as

$$\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{l}}_1 \times \tilde{\boldsymbol{l}}_2, \tag{2.4}$$

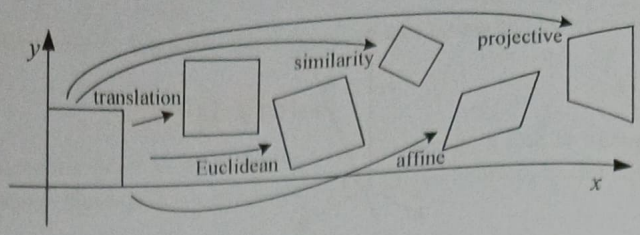


Figure 2.4 Basic set of 2D planar transformations.

3D quadrics. The 3D analog of a conic section is a quadric surface

$$\bar{x}^T Q \bar{x} = 0 \tag{2.13}$$

(Hartley and Zisserman 2004, Chapter 2). Again, while quadric surfaces are useful in the study of multi-view geometry and can also serve as useful modeling primitives (spheres, ellipsoids, cylinders), we do not study them in great detail in this book.

2.1.2 2D transformations

Having defined our basic primitives, we can now turn our attention to how they can be transformed. The simplest transformations occur in the 2D plane and are illustrated in Figure 2.4.

Translation. 2D translations can be written as $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}$ or

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \end{bmatrix} \bar{\boldsymbol{x}} \tag{2.14}$$

where $\boldsymbol{I}$ is the (2×2) identity matrix or

$$\bar{\boldsymbol{x}}' = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \bar{\boldsymbol{x}} \tag{2.15}$$

where $\boldsymbol{0}$ is the zero vector. Using a 2×3 matrix results in a more compact notation, whereas using a full-rank 3×3 matrix (which can be obtained from the 2×3 matrix by appending a $[\boldsymbol{0}^T\ 1]$ row) makes it possible to chain transformations using matrix multiplication. Note that in any equation where an augmented vector such as $\bar{\boldsymbol{x}}$ appears on both sides, it can always be replaced with a full homogeneous vector $\tilde{\boldsymbol{x}}$.

Rotation + translation. This transformation is also known as *2D rigid body motion* or the *2D Euclidean transformation* (since Euclidean distances are preserved). It can be written as $\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}$ or

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \bar{\boldsymbol{x}} \tag{2.16}$$

where

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.17}$$

is an orthonormal rotation matrix with $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$ and $|\boldsymbol{R}| = 1$.

Scaled rotation. Also known as the *similarity transform*, this transformation can be expressed as $\boldsymbol{x}' = s\boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}$ where s is an arbitrary scale factor. It can also be written as

$$\boldsymbol{x}' = \begin{bmatrix} s\boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \bar{\boldsymbol{x}} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \bar{\boldsymbol{x}}, \tag{2.18}$$

where we no longer require that $a^2 + b^2 = 1$. The similarity transform preserves angles between lines.

Affine. The affine transformation is written as $\boldsymbol{x}' = \boldsymbol{A}\bar{\boldsymbol{x}}$, where $\boldsymbol{A}$ is an arbitrary 2×3 matrix, i.e.,

$$\boldsymbol{x}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \bar{\boldsymbol{x}}. \tag{2.19}$$

Parallel lines remain parallel under affine transformations.

Projective. This transformation, also known as a *perspective transform* or *homography*, operates on homogeneous coordinates,

$$\tilde{\boldsymbol{x}}' = \tilde{\boldsymbol{H}}\tilde{\boldsymbol{x}}, \tag{2.20}$$

where $\tilde{\boldsymbol{H}}$ is an arbitrary 3×3 matrix. Note that $\tilde{\boldsymbol{H}}$ is homogeneous, i.e., it is only defined up to a scale, and that two $\tilde{\boldsymbol{H}}$ matrices that differ only by scale are equivalent. The resulting homogeneous coordinate $\tilde{\boldsymbol{x}}'$ must be normalized in order to obtain an inhomogeneous result $\boldsymbol{x}$, i.e.,

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \quad \text{and} \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}. \tag{2.21}$$

Perspective transformations preserve straight lines (i.e., they remain straight after the transformation).

Hierarchy of 2D transformations. The preceding set of transformations are illustrated in Figure 2.4 and summarized in Table 2.1. The easiest way to think of them is as a set of (potentially restricted) 3×3 matrices operating on 2D homogeneous coordinate vectors. Hartley and Zisserman (2004) contains a more detailed description of the hierarchy of 2D planar transformations.

The above transformations form a nested set of *groups*, i.e., they are closed under composition and have an inverse that is a member of the same group. (This will be important later when applying these transformations to images in Section 3.6.) Each (simpler) group is a subset of the more complex group below it.

Co-vectors. While the above transformations can be used to transform points in a 2D plane, can they also be used directly to transform a line equation? Consider the homogeneous equation $\tilde{\boldsymbol{l}} \cdot \tilde{\boldsymbol{x}} = 0$. If we transform $\boldsymbol{x}' = \tilde{\boldsymbol{H}}\boldsymbol{x}$, we obtain

$$\tilde{\boldsymbol{l}}' \cdot \tilde{\boldsymbol{x}}' = \tilde{\boldsymbol{l}}'^T \tilde{\boldsymbol{H}}\tilde{\boldsymbol{x}} = (\tilde{\boldsymbol{H}}^T \tilde{\boldsymbol{l}}')^T \tilde{\boldsymbol{x}} = \tilde{\boldsymbol{l}} \cdot \tilde{\boldsymbol{x}} = 0, \tag{2.22}$$

i.e., $\tilde{\boldsymbol{l}}' = \tilde{\boldsymbol{H}}^{-T}\tilde{\boldsymbol{l}}$. Thus, the action of a projective transformation on a *co-vector* such as a 2D line or 3D normal can be represented by the transposed inverse of the matrix, which is equivalent to the *adjoint* of $\tilde{\boldsymbol{H}}$, since projective transformation matrices are homogeneous. Jim

This suggests that it might be prudent before computing a full essential matrix to first compute a rotation estimate $\boldsymbol{R}$ using (6.32), potentially with just a small number of points, and then compute the residuals after rotating the points before proceeding with a full $\boldsymbol{E}$ computation.

7.2.1 Projective (uncalibrated) reconstruction

In many cases, such as when trying to build a 3D model from Internet or legacy photos taken by unknown cameras without any EXIF tags, we do not know ahead of time the intrinsic calibration parameters associated with the input images. In such situations, we can still estimate a two-frame reconstruction, although the true metric structure may not be available, e.g., orthogonal lines or planes in the world may not end up being reconstructed as orthogonal.

Consider the derivations we used to estimate the essential matrix $\boldsymbol{E}$ (7.10–7.12). In the uncalibrated case, we do not know the calibration matrices $\boldsymbol{K}_j$, so we cannot use the normalized ray directions $\hat{\boldsymbol{x}}_j = \boldsymbol{K}_j^{-1}\boldsymbol{x}_j$. Instead, we have access only to the image coordinates $\boldsymbol{x}_j$, and so the essential matrix (7.10) becomes

$$\hat{\boldsymbol{x}}_1^T \boldsymbol{E}\hat{\boldsymbol{x}}_1 = \boldsymbol{x}_1^T \boldsymbol{K}_1^{-T}\boldsymbol{E}\boldsymbol{K}_0^{-1}\boldsymbol{x}_0 = \boldsymbol{x}_1^T \boldsymbol{F}\boldsymbol{x}_0 = 0, \tag{7.28}$$

where

$$\boldsymbol{F} = \boldsymbol{K}_1^{-T}\boldsymbol{E}\boldsymbol{K}_0^{-1} = [\boldsymbol{e}]_\times \tilde{\boldsymbol{H}} \tag{7.29}$$

is called the *fundamental matrix* (Faugeras 1992; Hartley, Gupta, and Chang 1992; Hartley and Zisserman 2004).

Like the essential matrix, the fundamental matrix is (in principle) rank two,

$$\boldsymbol{F} = [\boldsymbol{e}]_\times \tilde{\boldsymbol{H}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \begin{bmatrix} \boldsymbol{u}_0 & \boldsymbol{u}_1 & \boldsymbol{e}_1 \end{bmatrix} \begin{bmatrix} \sigma_0 & & \\ & \sigma_1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_0^T \\ \boldsymbol{v}_1^T \\ \boldsymbol{e}_0^T \end{bmatrix}. \tag{7.30}$$

Its smallest left singular vector indicates the epipole $\boldsymbol{e}_1$ in the image 1 and its smallest right singular vector is $\boldsymbol{e}_0$ (Figure 7.3). The homography $\tilde{\boldsymbol{H}}$ in (7.29), which in principle should equal

$$\tilde{\boldsymbol{H}} = \boldsymbol{K}_1^{-T}\boldsymbol{R}\boldsymbol{K}_0^{-1}, \tag{7.31}$$

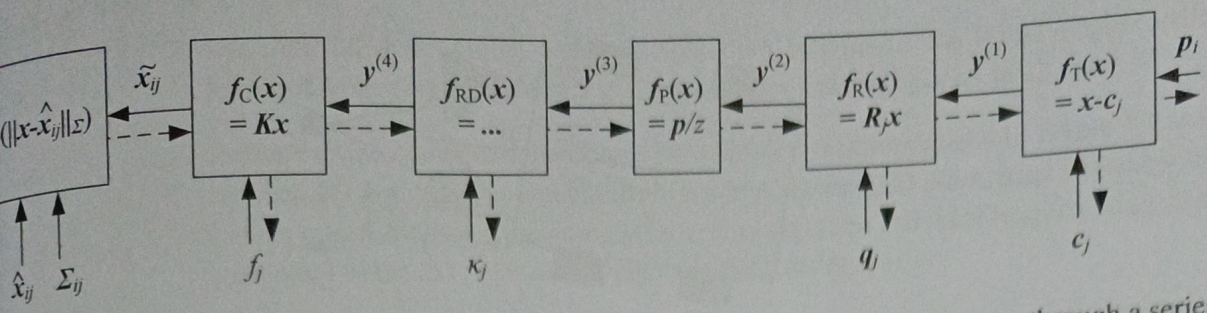
cannot be uniquely recovered from $\boldsymbol{F}$, since any homography of the form $\tilde{\boldsymbol{H}}' = \tilde{\boldsymbol{H}} + \boldsymbol{e}\boldsymbol{v}^T$ results in the same $\boldsymbol{F}$ matrix. (Note that $[\boldsymbol{e}]_\times$ annihilates any multiple of $\boldsymbol{e}$.)

Any one of these valid homographies $\tilde{\boldsymbol{H}}$ maps some plane in the scene from one image to the other. It is not possible to tell in advance which one it is without either selecting four or more co-planar correspondences to compute $\tilde{\boldsymbol{H}}$ as part of the $\boldsymbol{F}$ estimation process (in a manner analogous to guessing a rotation for $\boldsymbol{E}$) or mapping all points in one image through $\tilde{\boldsymbol{H}}$ and seeing which ones line up with their corresponding locations in the other.[7]

In order to create a *projective* reconstruction of the scene, we can pick any valid homography $\tilde{\boldsymbol{H}}$ that satisfies Equation (7.29). For example, following a technique analogous to Equations (7.18–7.24), we get

$$\boldsymbol{F} = [\boldsymbol{e}]_\times \tilde{\boldsymbol{H}} = \boldsymbol{S}\boldsymbol{Z}\boldsymbol{R}_{90^\circ}\boldsymbol{S}^T\tilde{\boldsymbol{H}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{7.32}$$

[7] This process is sometimes referred to as *plane plus parallax* (Section 2.1.5) (Kumar, Anandan, and Hanna 1994; Sawhney 1994).



7.7 A set of chained transforms for projecting a 3D point p_i into a 2D measurement x_{ij} through a series formations $f^{(k)}$, each of which is controlled by its own set of parameters. The dashed lines indicate of information as partial derivatives are computed during a backward pass. The formula for the radial on function is $f_{RD}(x) = (1 + \kappa_1 r^2 + \kappa_2 r^4)x$.

arate texture map from each reference camera and to blend between them during rendering, which is known as *view-dependent texture mapping* (Section 13.1.1) (Debevec, Taylor, and Malik 1996; Debevec, Yu, and Borshukov 1998).

7.4 Bundle adjustment 4m

As we have mentioned several times before, the most accurate way to recover structure and motion is to perform robust non-linear minimization of the measurement (re-projection) errors, which is commonly known in the photogrammetry (and now computer vision) communities as *bundle adjustment*.[13] Triggs, McLauchlan, Hartley *et al.* (1999) provide an excellent overview of this topic, including its historical development, pointers to the photogrammetry literature (Slama 1980; Atkinson 1996; Kraus 1997), and subtle issues with gauge ambiguities. The topic is also treated in depth in textbooks and surveys on multi-view geometry (Faugeras and Luong 2001; Hartley and Zisserman 2004; Moons, Van Gool, and Vergauwen 2010).

We have already introduced the elements of bundle adjustment in our discussion on iterative pose estimation (Section 6.2.2), i.e., Equations (6.42–6.48) and Figure 6.5. The biggest difference between these formulas and full bundle adjustment is that our feature location measurements x_{ij} now depend not only on the point (track index) i but also on the camera pose index j,

$$x_{ij} = f(p_i, R_j, c_j, K_j), \tag{7.49}$$

and that the 3D point positions p_i are also being simultaneously updated. In addition, it is common to add a stage for radial distortion parameter estimation (2.78),

$$f_{RD}(x) = (1 + \kappa_1 r^2 + \kappa_2 r^4)x, \tag{7.50}$$

if the cameras being used have not been pre-calibrated, as shown in Figure 7.7.

While most of the boxes (transforms) in Figure 7.7 have previously been explained (6.47), the leftmost box has not. This box performs a robust comparison of the predicted and mea-

[13] The term "bundle" refers to the bundles of rays connecting camera centers to 3D points and the term "adjustment" refers to the iterative minimization of re-projection error. Alternative terms for this in the vision community include *optimal motion estimation* (Weng, Ahuja, and Huang 1993) and *non-linear least squares* (Appendix A.3) (Taylor, Kriegman, and Anandan 1991; Szeliski and Kang 1994).

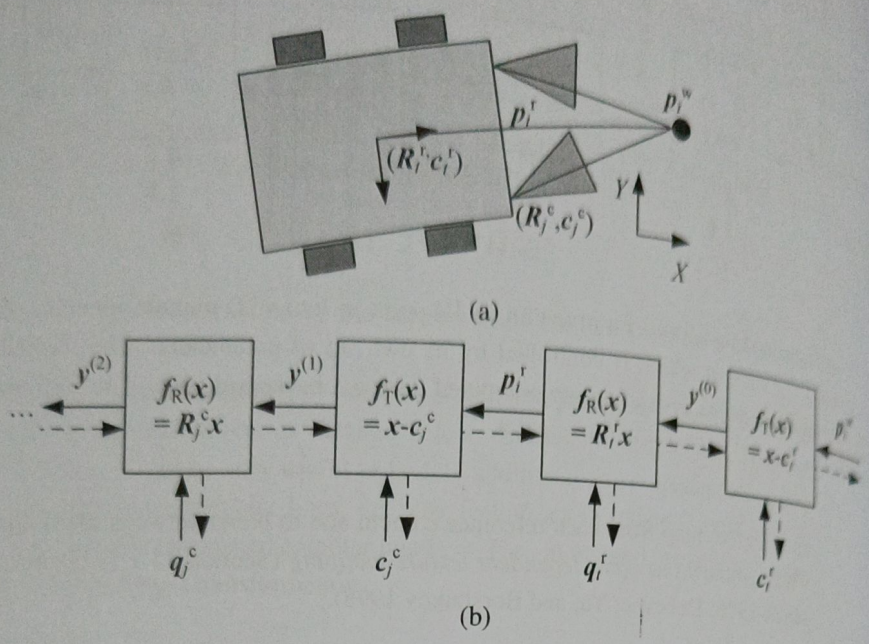


Figure 7.8 A camera rig and its associated transform chain. (a) As the mobile rig (robot) moves around world, its pose with respect to the world at time t is captured by $(\boldsymbol{R}_t^{\mathrm{r}}, \boldsymbol{c}_t^{\mathrm{r}})$. Each camera's pose with respe rig is captured by $(\boldsymbol{R}_j^{\mathrm{c}}, \boldsymbol{c}_j^{\mathrm{c}})$. (b) A 3D point with world coordinates $\boldsymbol{p}_i^{\mathrm{w}}$ is first transformed into rig coordina and then through the rest of the camera-specific chain, as shown in Figure 7.7.

sured 2D locations $\hat{\boldsymbol{x}}_{ij}$ and $\bar{\boldsymbol{x}}_{ij}$ after re-scaling by the measurement noise covariance Σ_{ij}. In more detail, this operation can be written as

$$\boldsymbol{r}_{ij} = \bar{\boldsymbol{x}}_{ij} - \hat{\boldsymbol{x}}_{ij}, \tag{7.51}$$

$$s_{ij}^2 = \boldsymbol{r}_{ij}^T \Sigma_{ij}^{-1} \boldsymbol{r}_{ij}, \tag{7.52}$$

$$e_{ij} = \hat{\rho}(s_{ij}^2), \tag{7.53}$$

where $\hat{\rho}(r^2) = \rho(r)$. The corresponding Jacobians (partial derivatives) can be written as

$$\frac{\partial e_{ij}}{\partial s_{ij}^2} = \hat{\rho}'(s_{ij}^2), \tag{7.54}$$

$$\frac{\partial s_{ij}^2}{\partial \bar{\boldsymbol{x}}_{ij}} = \Sigma_{ij}^{-1} \boldsymbol{r}_{ij}. \tag{7.55}$$

The advantage of the chained representation introduced above is that it not only makes the computations of the partial derivatives and Jacobians simpler but it can also be adapted to any camera configuration. Consider for example a pair of cameras mounted on a robot that is moving around in the world, as shown in Figure 7.8a. By replacing the rightmost two transformations in Figure 7.7 with the transformations shown in Figure 7.8b, we can simultaneously recover the position of the robot at each time and the calibration of each camera with respect to the rig, in addition to the 3D structure of the world.