SIFT (Scale Invariant Keypoint descriptor)

1. Take a 16×16 window around the detected point.
2. Compute the edge orientation for each pixel.
3. Divide into a 4×4 grid of cells
4. compute the histogram in each cell.

Orientation assignment (for rotation invariance)

1. Create histogram of gradient directions, within a region around a keypoint, at selected scale.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

magnitude → $m(x,y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$

angle/phase → $\theta(x,y) = \tan^{-1}\left(\frac{(L(x, y+1) - L(x, y-1))}{L(x+1, y) - L(x-1, y)}\right)$

- The reason for orientation assignment is to compute the main direction among the pixels. For this, we will build a histogram (from 0 - 2π)

→ Histogram entries are weighted (i) gradient magnitude, (ii) a Gaussian function with σ equal to 1.5 times the scale of the keypoint (max at centre) for more weight.

→ We can calculate the keypoint by visualising the histogram bins. We set a threshold of ~80% and we can have a main edge orientation of the point.
If, there are multiple orientations in the point of interest, we will generate duplicate keypoints. limit to 2 peak.

→ To create a local histogram (Keypoint descriptors)
1. Each histogram entry is weighted by
   (1) a gradient magnitude
   (2) a gaussian function with σ equal to 0.5 times the width of the descriptor window.

→ matching SIFT features :-

$$SSD(f_1, f_2) / SSD(f_1, f_2') < t \quad \text{if } t = 0.8$$

SIFT main steps

1. Scale-space extrema detection → Extract scale and rotation invariant interest points. (i.e. keypoints)
2. Keypoint localization → Determine location and scale for each interest point
   → Eliminate weak keypoints
3. Orientation assignment → Assign one or more orientation to each keypoint
4. Keypoint descriptor → use local image gradients at the selected scale.

Continental
The Future in Motion

Feature matching, Tracking and optical flow.

→ Feature matching → If we have 2 images and we have to match their features we follow a simple approach.

Step 1 → Detect corners in the image with fn eg. Harris detector.

Step 2 → Compute similarity / correlation scores. Take a point in the original image and find a point at the same location or in a proximity with the help of disparity window.
with the help of all the point, draw a correlation curve on the intensity neighborhoods.

Step 3 → select the point with the highest score.

⇒ usually during template matching, many wrong matches (10-50%) are also done. But even with this error rate, we can start computation "motion". All thanks to outliers rejection techniques such as RANSAC

→ with the help of invariant descriptors, we can improve the "similarity measure" of features.

invariant (to light, scale, viewpoint)

features (transform (image)) = feature (image)

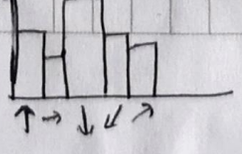
→ Scale Invariant Feature Transform (SIFT) → Advantages

① Locality → features are local, robust to occlusion and clutter.
② Distinctiveness → individual features can be matched to a large database of objects
③ Quantity → many features can be generated for even small objects
④ Efficiency → close to real-time performance.

Algorithm → ① Scale-space peak selection → Potential location for finding features.
The scale-space of an image is produced from the convolution of Gaussian Kernel (Blurring) at different scales with the original image.
Scale space is separated into octaves and the no. of octaves and scale depends on the size of the original image.
Once the image is blurred, the pixel is compared with 8 neighbours. Also compared with the nine pixels in previous and next scales. (Total 26)
If it is local extrema, it is a potential keypoint.

② Keypoint localisation → If the intensity at this extrema is less than the threshold value, they are rejected.
→ Edges are removed using eigen values and their ratios. We only need features
→ Removal of low contrast keypoints and edge keypoints. Left with only strong interest points.

③ Orientation Assignment → Orientation is assigned to each keypoint.
→ orientation histogram is created. Highest peak in the histogram and any peak above 80% is considered.
→ creates keypoints with same location and scale but different direction.



④ Keypoint descriptor → we start with a 16×16 window around detected interest point we compute edge orientation for each pixel. sub-group 16×16 window to 4×4 windows Compute histograms. so, 4×4 descriptors over 16×16 sample array. 4×4×8 → 128 bin values.

→ Histogram entries are weighted by: (i) gradient magnitude (ii) A gaussian function (more at centre.) with σ equal to 1.5 times the scale of the keypoint.

→ Partial Voting → distribute histogram entries into adjacent bins. Each entry is added to all bins, multiplied by a weight of 1-d, (d is the distance from the bin it belongs to).

→ matching SIFT features → accept the match if $SSD(f_1, f_2) / SSD(f_1, f_2') < t$

$t = 0.8$ has given good results in object recognition.

→ Advantages / Properties of SIFT

① can handle changes in viewpoint → upto 60° out of plane rotation.
② can " " " in illumination → 1/4 even day-night
③ Fast and efficient → can run in real-time.

Optical flow

→ A vector field function of the spatio-temporal image brightness variations. It basically shows how does every pixel move from one image to another. (over time and space).

✓ motion field → It is the 2d ~~representation~~ reprojection of 3d-motion of the object in the scene into the image. True motion of the object.

✓ optical flow → apparent motion of brightness patterns in the image.

→ optical flow ≠ motion field. (sphere-sun) example.

Feature Tracking → extract visual features and track them over multiple frames (eg. corners, textured areas)

Optical flow → Recover image motion at each pixel from spatio-temporal image brightness variations.

Solving these 2 problems with one method

Basic Assumptions: Brightness consistency, Spatial coherence, Temporal persistance.

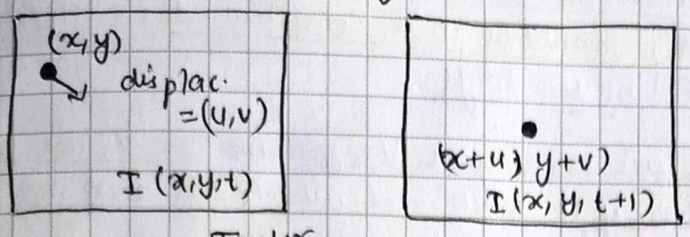
✓ Brightness Consistency → Image measurement (brightness) in a small region remain the same although their location may change.

✓ Spatial coherence → Neighbouring points in the scene typically belong to the same surface and hence typically have similar motions. (consider a small patch)

→ Since they also project to the nearby pixels in the image we expect spatial coherence in the image flow.

✓ Temporal Consistency → The image motion of a surface patch changes gradually over time.

→ **The brightness constancy constraint** :-



→ The brightness constancy equation:-

$$I(x, y, t) = I(x+u, y+v, t+1)$$

→ Taking the Taylor expansion of $I(x+u, v+y+t+1)$ at (x,y,t) to linearize the right side.

$$I(x+u, y+v, t+1) \simeq I(x,y,t) + I_x \cdot u + I_y \cdot v + I_t$$

(I_x: Image derivative along x; I_t: Difference over frames.)

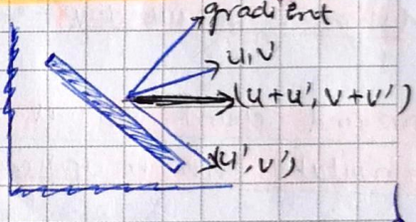
$$I(x+u, y+v, t+1) - I(x,y,t) = I_x \cdot u + I_y \cdot v + I_t \simeq 0$$

$$\therefore \simeq \boxed{\nabla I \cdot [u\ v]^T + I_t \simeq 0}$$

Brightness constancy equation

✓ Since it is a scalar equation with 2 unknown, it is not solvable → Solvable? NO

✓ **Aperture problem** → This problem states that if we are looking only at a single pixel, we cannot ~~identify its motion~~ i.e. Different motions classified as similar. or differentiate b/w the motion similar motions classified as different.



⇒ To solve this problem :- we have 2 methods:
(A) Local method → Lukas and Konade
(B) Global method → Horn and Schunck

→ **Local method** → We try to get more equations for 1 pixel. How? By spatial coherence constant.
→ Assuming the pixel's neighbours to have the same displacement (u,v)
→ for eg. if we use a 5×5 window, that gives us 25 equations/pixel.

$$\Rightarrow \boxed{0 = I_t(p_i) + \nabla I(p_i) \cdot [u\ v]} \Rightarrow \underbrace{\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ I_x(p_3) & \\ \vdots & \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix}}_{25\times 2\ (A)} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{2\times 1\ (d)} = -\underbrace{\begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}}_{25\times 1\ (b)}$$

⇒ **Now the Problem**: we have more equations than unknowns.

$$\boxed{\text{minimize } \|Ad - b\|^2}$$

Solution → Solve least square problem

$$(x = (A^TA)^{-1}A^Tb$$

in MATLAB $x = A \backslash b$

minimum least squares solution given by solution of: $(A^TA)d = A^Tb$ (2×2 · 2×1 = 2×1)

$$\begin{bmatrix} \Sigma I_x I_x & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \Sigma I_x I_t \\ \Sigma I_y I_t \end{bmatrix}$$

A^TA · d · A^Tb

→ Conditions for solvability:
1. A^TA should be invertible
2. A^TA should not be too small due to noise i.e. the eigen values of λ_1 and λ_2 of A^TA should not be too small.
3. A^TA should be well-conditioned i.e. λ_1/λ_2 should not be too large.

Same as Harris Corner detector

Observation: ① Can measure sensitivity by just looking at one of the images
② This tells us which pixels are easy to track and which are hard.

Errors in Lucas-Kanade: Potential causes of errors:- ① Suppose A^TA is easily invertible ② suppose there is not much noise in the image.

→ When are our assumptions violated?
1. Brightness constancy is not satisfied.
2. The motion is not small.
3. A point does not move like its neighbours.
   - → window size is too large
   - → what is the ideal window size?

→ Implementation issues ① Window size → Small window size more sensitive to noise and may miss larger motions.
→ Large window more likely to cross an occlusion boundary (and is also slower)
→ Typical size 15×15 – 31×31

② weighting the window → common to apply weights so that the center matters more

⇒ Horn and Schunk global method (Dense optical flow)

① Formulate error in optical flow constraint: should be as small as possible

$$e_c = \iint_{image} (I_x u + I_y v + I_t)^2 \, dx \, dy$$

② Adding the smoothness constraint:

$$e_s = \iint_{image} (u_x^2 + u_y^2) + (v_x^2 + v_y^2) \, dx \, dy$$

⇒ usually motion field varies smoothly in the image. Penalize departure from smoothness

③ Find (u,v) at each point that minimizes: $e = e_c + \lambda e_s$ (global error; λ: Regulation term)

→ What happens if the motion is not small? (i.e. much larger than 1 pixel)
→ 2nd order terms dominate. Aliasing problem occurs.
→ Aliasing → Temporal aliasing causes ambiguities in optical flow because images can have many pixels with the same intensity.
→ Solution → coarse to fine estimation i.e. Reducing the Resolution

→ Dealing with larger movements: Iterative refinement

1. Initialize $(x', y') = (x, y)$
2. Compute (u, v) by: $\begin{bmatrix} \Sigma I_x I_x & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \Sigma I_x I_t \\ \Sigma I_y I_t \end{bmatrix}$
   - $I_t = I(x', y', t+1) - I(x, y, t)$
   - ↳ displacement
3. Shift the window by (u, v): $x' = x' + u$ $y' = y' + u$
4. Recalculate I_t
5. Repeat the steps 2-4 until small change

Iterative Refinement → Iterative KLT algorithm.

1. Estimate displacement at each pixel by solving L-K equations
2. Wrap I_t towards I_{t+1} using estimated flow field (Interpolation)
3. Repeat until convergence.

→ Tracking over many frames:-

1. Select features in the first frame
2. For each frame:
   1. update the position of tracked features:
      By Discrete search or Lukas-Kanade method
   2. Terminate in consistent tracks:
      Compute similarity with the corresponding feature in the previous frame or in the first frame where its visible.
   3. Find more features to track