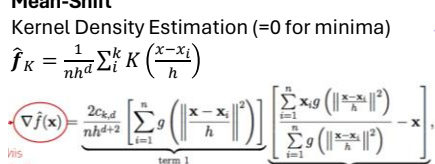
**Kmeans:**
Params: I, or r,g,b, or x,y
- Setting k
- Sensitive to initial centres, outliers
- Assumes spherical clusters
- Assumes means are meaningful

**SLIC**
"#pixels,#superpixels: $n_{tp}; n_{sp}$
initial spacing of each superpixel $s = \sqrt{n_{tp}/n_{sp}}$
Features: $z = [r, g, b, x, y]$
Colour distance $d_c = ||(r_j - r_i, g_j - g_i, b_j - b_i)||$
Spatial Distance $d_s = ||(x_j - x_i, y_j - y_i)||$
Composite Distance= $D = ||(\frac{d_c}{d_{cm}}, \frac{d_s}{d_{sm}})||$
$\Rightarrow D = \sqrt{(d_c^2 + (d_s/s)^2 c^2)}$

**Hyperparams**: $c$(larger=constancy), $n_{sp}$
1. Initialise cluster centres to lowest gradient in 3x3 neighbourhood (avoid edges, noise), $label = -1, d(p) = \infty$
2. Assign, computing $D(p)$ between each ctr and pixel in $2s \times 2s$ neighbourhood.
3. Update cluster centres
4. Test convergence
5. Replace superpixels with average value

**Mean-Shift**
Kernel Density Estimation (=0 for minima)
$\hat{f}_K = \frac{1}{nh^d}\sum_i^k K\left(\frac{x - x_i}{h}\right)$



For each point, until convergence:
Initialise density window $\boldsymbol{x} = \boldsymbol{x_j}$
Compute mean shift vector (term 2)
Shift density window $\boldsymbol{x'} = \boldsymbol{x} + \boldsymbol{m(x)}, \boldsymbol{x} = \boldsymbol{x'}$
Hyperparam: $\boldsymbol{h}$ (bandwidth size)

**Optimisation**
- all points within radius r have same window
- all points within search path have same window

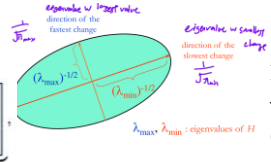
**Filter Banks**
1. k-dim filter bank on m img: output: **(m,y,x,k)**
2. Kmeans and store cluster ctrs (Texton dict **N**-dim)
3. Each test img: filter w same bank for feat rep for each pixel. Assign to nearest cluster. Cluster ID is texton ID.

**Histogram**
Each pixel has closest ID from texton dict.
Hist can also be computed with in windows ctr'd per px

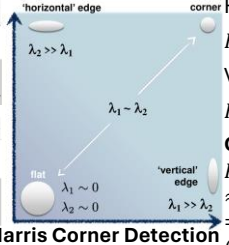
**Gabor equation (sin=real, odd. cos=even)**
$f_{mn} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{m^2+\gamma^2 n^2}{2\sigma^2}\right)\sin\left(\frac{2\pi(\cos(\omega)m + \sin(\omega)n)}{\lambda} + \phi\right)$
$m, n$: 2d coord of kernel. $m>n$, flat, $m<n$, skinny
$\gamma$ – aspect ratio of envelope coverage ($\gamma$=1 means circular, else elliptical
$\sigma$ – rate of decay of exponential envelope (height of peaks)
$\omega$ – orientation of sinusoid in 2D, influences orientation of detected feature (clockwise: 0 for vert, π/2 for hori)
$\lambda$ – wavelength of sinusoid; (wavelength is inversely proportional to frequency); (how spread-out peaks) usually set to be proportional to $\sigma$ so that a constant number of cycles is visible)
$\phi$ – phase of sinusoid; shift from center of kernel

**Texture Boundary (dir edge detection)**
Try all orientations of a disk Θ to find max hist difference
Textons are usually not (locally) precise for segmentation due to aggregation from window. Use spatially smoothed feature response instead.

**Descriptors** (save time complexity of shift_range)
Corner has significant change in all directions on shifting
Taylor expansion: $I(x+u, y+v) \approx I(x,y) + I_x u + I_y v$
SSD: $E(u,v) = \sum_{x,y\in w}[I(x+u, y+v) - I(x,y)]^2$
$\approx \sum_{x,y\in w}[I_x u + I_y v]^2 = Au^2 + 2Buv + Cv^2$
$= [u\ v]\begin{bmatrix} A & B \\ B & C \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$, with $\sum A = I_x^2, \sum B = I_x I_y, \sum C = I_y^2$
Ellipse axes ori determined by the eigenvectors of H
Ellipse axes length determined by the eigenvalues of H



**Harris avoids evec, eval**
Evector x: $Ax = \lambda x$
let $A = \begin{bmatrix} h_{11} - \lambda & h_{12} \\ h_{21} & h_{22} - \lambda \end{bmatrix}$
To get $\lambda$, solve:
$\det(A) = ad - bc = 0.$
Eval: solve $A\begin{bmatrix} x \\ y \end{bmatrix} = 0$
Hori edges: $I_x = 0$, $H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$
Vert edges: $I_y = 0$, $H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$



**Cornerness score $R$**
$R = \min(\lambda_1, \lambda_2)$
$\approx \det(H) - trace(H)$
$= \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$
$(trace(A) = a + d)$

**Harris Corner Detection**
1. Compute gradient at each point
2. Convolve 3 channels w np.ones(window), calculate cornerness score $\det(H) - \kappa\ tr(H)$
3. Find points where R>threshold
4. NMS max points

**Simple NMS**
Search for max values and zero out surrounding window

**Adaptive NMS**
Pick corners where R >>> neighbouring maxima of radius **r**

**Equivariance:** transform img → detection (locations) undergoes similar transformation
**Invariance**: image transformed & detection (score) does not change

**Harris Variance**
Harris location is equivariant. response invariant to translation, rotation, additive intensity change. Variant to intensity scaling, img scaling, smoothing

**LoG (blob detection)**
$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$ roughly equals diff of gaussians
Gaussian: $f_{uv} = \frac{1}{2\pi\sigma^2}\exp(-\frac{u^2+v^2}{2\sigma^2})$
1st line: loc are equivar, 2nd line: descr are invar
Transformation: $I'(x,y) = I(x+\Delta u, y+\Delta v)$ where $\Delta u > 0, \Delta v > 0$
custom, Harris, LoG, SIFT (all are invariant to position offsets)
weighted intensity, weighted gradients, GIST, SIFT (all except GIST)

Transformation: $I'(x,y) = I(x,y) + \Delta u$, where $\Delta u > 0$
custom, Harris, LoG, SIFT (all are invariant to intensity pffset)
weighted intensity, weighted gradients, GIST, SIFT (all except weighted intensity)

Transformation: $I'(x,y)$ is a mirrored version of $I(x,y)$, i.e. flipped over the X axis.
custom, Harris, LoG, SIFT (all are invariant to flipping)
weighted intensity, weighted gradients, GIST, SIFT (GIST and SIFT are not mirror invariant)

Transformation: $I'(x,y) = I(y,x)$
custom, Harris, LoG, SIFT (all are invariant to transpose)
weighted intensity, weighted gradients, GIST, SIFT (GIST and SIFT descriptors are not invar

Transformation: $I'(x,y) = I(0.5 * x, y)$
custom, Harris, LoG, SIFT (none)
weighted intensity, weighted gradients, GIST, SIFT (none)

Transformation: $I'(x,y) = I(0.5 * x, 0.5 * y)$
custom, Harris, LoG, SIFT (scaling transformation)
weighted intensity, weighted gradients, GIST, SIFT (only SIFT)

Transformation: $I'(x,y)$ is a histogram stretched version of to $I(x,y)$.
custom, Harris, LoG, SIFT (none are invariant)
weighted intensity, weighted gradients, GIST, SIFT (none)

Transformation: $I'(x,y)$ is 45 degree clock-wise rotated version of $I(x,y)$.
custom, Harris, LoG, SIFT (all except custom)
weighted intensity, weighted gradients, GIST, SIFT (all except GIST)

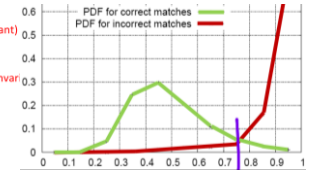
**Orientation Normalisation**
In each window, take mode of grad bins

**Multi-scale Oriented Patches (MOPS)**
1. Take 40x40 window around detected keypoint (e.g. Harris corner)
2. Subsample every 5th pixel.
3. Rotate to horizontal.
4. Normalize intensity values within window $(x - \bar{x})/\sigma$
5. Wavelet transform on 8x8 patch to get a 64-dimensional descriptor

**GIST** (get rough spatial dist of img grad)
1. Divide image (patch) into 4 x 4 cells
2. Apply Gabor filters (directional edge detectors).
3. Compute filter response averages for each cell.
4. Size of descriptor is 4 x 4 x N, where N is size of the filter bank.

**SIFT Descriptor**
1.1 Take 16x16 window around detected kp from image at scale matching to kp.
1.2. Partition window into 4x4 grid of cells
2.1 Compute gradient ori and mag per px;
2.2 Reweight mag according to Gaussian centered on kp, discard pixels w low mag
3.1 Create hist with 8 bins for each cell.
3.2 For rotation-invariance, shift histogram binning by dominant orientation.
3.3 Collapse into vector (16 x 8 = 128 dims).
4. Normalize vector to unit length, clamp values based on threshold, re-normalize again for final descriptor

**Variances**
SIFT invar to rot, trans.
Not invar to affine, project
Not invar to perspective changes
Semi-invar to up to 60deg out of plane rot




**Feature Matching**
ratio dist=$||f1 - f2|| \ / \ ||f1' - f2'||$
(best / next-match)
**Precision:** TP/(TP+FP)
**Recall:** TP/(TP+FN)
**Specificity:** TN/(TN+FP)
**ROC**: TPR against FPR

Filtering: $G(x) = h\{F(x)\}$
(range i.e intensity changes)
Warping: $G(x) = F(h\{x\})$
(domain i.e. location changes)

**Homography (DLT)**
homogeneous coords (add row of 1)
[3x1]=[3x3][3x1]
$P' = HP = [x'\ y'\ z']^T = [x'/w'\ y'/w']^T$
To find $H$, get this eqn ($h_9$ is scaled out)
$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \alpha\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
For $A_i h = 0$:
$A_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$
$h = [h1 \dots h9]^T$

**Solving for H**
Given $\{p_i, p_i'\}$, for all $p$,
1. Create 2x9 matrix $A_i$
2. Concat into 2n*9 matrix
3. Compute SVD of $A = U\Sigma V^T$
4. Store smallest singular $v_i$
5. Reshape to get H

**DLT**
- H has 8 dof, each kp correspondence is 2 eqn, so need 4 matches to solve
- Requires linear transformation (two images may not be projective)
- Sensitive to scaling (need norm)
- Sensitive to outliers

**Normalised DLT**
1. mx, my = arr_mean
2. sx, sy = arr_std * (1 / math.sqrt(2))
3. T = np.array([[1/sx, 0, -mx/sx],[0, 1/sy, -my/sy],[0, 0, 1]]
4. transform $\{p_i, p_i'\}$ with $T \cdot src$ and $T \cdot dst$
5. Run DLT on each $\{p_i, p_i'\}$ to get $\tilde{H}$
6. Denorm with $H = inv(T')\tilde{H}T$.

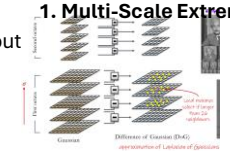
**RANSAC**
1. Sample min 4 points
2. Compute H using DLT
3. Count inliers among sample
4. Keep H if most inliers
5. For best H, recompute using all inliers

**Hyperparams:**
$\delta$ (inlier threshold), $N$ (iterations)

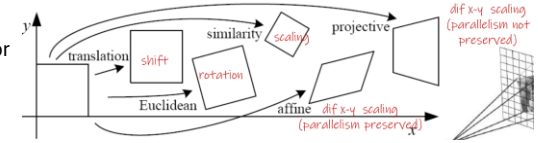
**Iterations**
$N = \frac{log(1-p)}{log(1-(1-e)^s)}$
e: probability outlier
s: num points in sample
p = probability that set has no outliers

**Warping**
**Forward Warping:** for each source (x,y), send intensity to (x',y'). Splat among dest pixels if in between
**Inverse warping:** for each destination, find (x,y) and interpolate value from neighbouring source pixels)

**1. Multi-Scale Extrema (Blob) Detection**



**2. Localization + Scale/Orientation**
• Refine kp loc to sub-pixel accuracy via Taylor expansion across the octaves
• Reject maxima below threshold
• Reject maxima w insuff local curvature



**Optical Flow**

**Colour Constancy Assumption**

(Brightness constancy for intensity images)

Implication: allows for pixel-to-pixel comparison (not image features)

**Small Motion Assumption**

(pixels only move a little bit)

Implication: linearization of the brightness constancy constraint

**Optical Flow**: $(u, v)$

**Displacement**: $(\delta x, \delta y) = (u\delta d, v\delta t)$

$I(x + i\delta t, y + v\delta t, t + \delta t = I(x, y, t)$

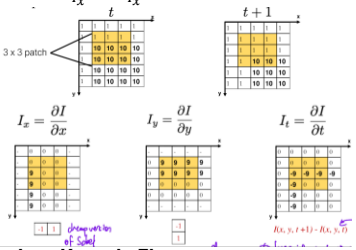
doing Taylor expansion yields:

**Brightness Constancy:**

$\frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0$

$I_x u + I_y v + I_t = 0$

$u = -\frac{I_y}{I_x}v - \frac{I_t}{I_x} \Rightarrow y = mx + b$




**Lukas-Kanade Flow**

Constant Flow Assumption

- all pixels in window have same constant flow

E.g. assume 5x5 patch has constant flow

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$ (25 eqns)

$Ax = -b \Rightarrow \hat{x} = argmin||Ax - b||^2$

$\Rightarrow A^T A\hat{x} = A^T b$

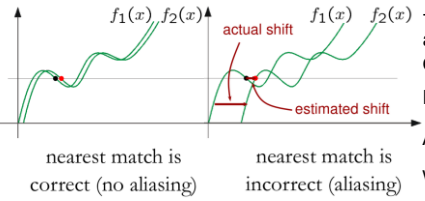
$$\Rightarrow \begin{bmatrix} \sum_{p\in P} I_x I_x & \sum_{p\in P} I_x I_y \\ \sum_{p\in P} I_y I_x & \sum_{p\in P} I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum_{p\in P} I_x I_t \\ \sum_{p\in P} I_y I_y \end{bmatrix}$$

Solvable when $A^T A$ is invertible

- $\lambda_1$and $\lambda_2$ not too small
- $A^T A$ well-conditioned
- $\lambda_1/\lambda_2$ not too large, where $\lambda_1 > \lambda_2$

i.e. corners are good

**If not small motion:**




**Aperture Problem**

When only line but no corners,

$I_x$ or $I_y$ = 0 then unable to recover u or v

**Coarse-to-Fine Estimation (pyramid)**

1. Downsample large motion
2. Compute LK flow $OF_0$
3. Upsample flow field
4. Warp higher res image with LK flow
5. Compute residual LK flow $OF_1$ using warped image
6. Repeat w increasing higher-res img
7. $OF = OF_0 + \cdots \Delta OF_n$

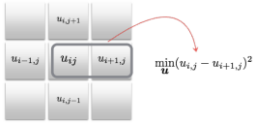
**Horn-Shunck Optical Flow**

Brightness constancy: every pixel,

$\min\left[I_x u_{ij} + I_y v_{ij} + I_t\right]^2$

Smooth flow field: $\min\left(u_{i,j} - u_{i+1,j}\right)^2$

Objective: $\min_{u,v} \sum_{i,j}\{E_s(i-,j) + \lambda E_d(i,j)\}$




Ave of 4 neighbours:

**HS Algorithm**

1. Precompute $I_x, I_y, I_t$
2. Let $u = 0, v = 0$
3. While not converged,

(ave, frac, update eqns)

$\bar{u}_{ij} = \frac{1}{4}\{u_{i+1,j} + u_{i-1,j} + u_{i,j+1}, u_{i,j+1}\}$

$frac = \frac{I_x\bar{u}_{kl} + I_y\bar{v}_{kl} + I_t}{\lambda^{-1} + I_x^2 + I_y^2}$

$\hat{u}_{kl} = \bar{u}_{kl} - frac \times I_x$

$\hat{v}_{kl} = \bar{v}_{kl} - frac \times I_y$

**Hyperparams:**

$\lambda$ small= emphasise smoothness

**Summary:**

**LK Flow** (constant flow), flow constant for all pixels, sparse, local method

- Extracts visual features, tracks over multiple frames
- Sparse motion fields
- Suitable for large motion

**HS Flow** (smooth flow), varies per pixel, dense, global method

- Based on spatio-temporal img brightness variations
- Dense motion fields, sensitive to appearance variations

**Ground Truth**

EndPointError = $\left|\left|\hat{f} - f_{GT}\right|\right|$

Angular Error = $\arccos(\frac{f^T \cdot \hat{f}}{|f||\hat{f}|})$

where $f_{GT} = (u, v, 1), \hat{f} = (\hat{u}, \hat{v}, 1)$

**Apparent Motion**

- Projected Motion field may not be correct e.g. lighting chances
- Constant Motion (LK) / smooth motion (HS) may not hold e.g. obj boundaries
- Brightness constancy may not hold
- Large motions, non-linear assumptions for iterative update

**Tracking**

1. Template matching:

Slow, combinatory, global solution

2. Multi-scale template matching

(norm-cc w smaller img to get peak, then upscale to get coarse location):

Faster, combinatory, locally optimal

3. Local refinement based on initial guess

**Image Alignment**

Use prev frame as template for next frame

Find warp parameter **p** for min SSD

Guess $p$, assume small increment.

Use Taylor series and solve using Least-Squares

Objective: $\min_p \sum_x \left[I\left(W(x;p)\right) - T(x)\right]^2$

$= \min_{\Delta p} \sum_x \left[I\left(W(x;p_0 + \Delta p)\right) - T(x)\right]^2$

$= \min_{\Delta p} \sum_x \left[\nabla I \frac{\partial W}{\partial p}\Delta p - \left(T(x) - I\left(W(x;p_0)\right)\right)\right]^2$ equiv to (Ax-b)

$\Delta p = H^{-1} \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^T \left[\left(T(x) - I\left(W(x;p_0)\right)\right)\right]$

$H = \sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^T \left[\nabla I \frac{\partial W}{\partial p}\right]$

x is coord, W is warp func (2x1) $p_0$ is warp param(2x3 for affine), $\nabla I$ is img grad (1x2), partial is Jacobian, $\Delta p$ is incremental warp (6x1 for affine), T(x) is template intensity

**LK Alignment (fit template to img)**

1. Warp image $I\left(W(x;p)\right)$
2. Compute error image $T(x) - I\left(W(x;p_0)\right)$
3. Compute gradient $\nabla I(x')$
4. Evaluate Jacobian $\frac{\partial W}{\partial p}$
5. Compute Hessian (above)
6. Compute $\Delta p$ (above)
7. Update p (above)

**KLT Algorithm for Feature Tracking (track template across img)**

1. Find corners satisfying $\min(\lambda_1, \lambda_2) > \lambda$ for the Hessian
2. Loop over corners:

- Compute displacement to next frame using the LK align
- Store displacement of each corner, update corner position
- (opt) Add more corner pts every M frames using step 1

3. Returns long trajectories for each corner point

**Problems**

- Feature selection, efficiency
- Accuracy eg rotation, shadow, drift
- Occlusion (add, remove points)

**Template Matching**

1. Compute template descriptor
2. Search for similar descriptor

**Template-based Tracking:**

- Feature selection
  - RGB, gradient, histograms, deep features
- Search for candidates
  - Limit search space based on prev results
- Solve the mode-finding problem
  - Global vs local max
- Update the target's template
  - Fixed vs update template

**Metrics**

Accuracy

- Intersection over union of bounding boxes (eg avg over frames)

Robustness

- Times that tracker fails (lose target)

**Deep Learning Params**

- 200x200 pixels = 40k params per hidden unit

Normalised hist means scaling doesn't affect. But texton dict is affected

DLT for circle

$$[x \quad y \quad 1] \begin{bmatrix} 2x_0 \\ 2y_0 \\ r^2 - x_0^2 - y_0^2 \end{bmatrix} = x^2 + y^2$$

Inlier: $(y - y_i)^2 < \delta$

eg $\left(r - \sqrt{(x - x_0)^2 + (y - y_0)^2}\right)^2 < \delta$

One brightness constancy eqn = 1 pixel = 1 equation

SIFT is robust but not invariant to affine and projective transforms

**MOSSE Filter**

Minimise MSE of corr output

$\boldsymbol{g} = \underset{\boldsymbol{g}}{\text{argmin}} \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{g} \otimes \boldsymbol{x}_i - \boldsymbol{y}_i)^2 + \lambda\|\boldsymbol{g}\|^2$

$\boldsymbol{g} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$

x is a N*hw matrix

y is a N*hw output

x1 is first kernel, x2, x3 are subsequent kernels

**Algorithm**

1. Initialise w several perturbed versions (robust)
2. Optimise Kernel g in DFT
3. Apply kernel g to frame $I_k$
4. Find local maxima to locate target BB
5. Update kernel w new BB

$\hat{\boldsymbol{a}}_{N+1} = (1 - \eta)\hat{\boldsymbol{a}}_N + \eta\,(\hat{\boldsymbol{x}}^*_{N+1} \odot \hat{\boldsymbol{y}}_{N+1})$

$\hat{\boldsymbol{b}}_{N+1} = (1 - \eta)\hat{\boldsymbol{b}}_N + \eta\,(\hat{\boldsymbol{x}}^*_{N+1} \odot \hat{\boldsymbol{x}}_{N+1})$

$\eta$ higher, more the next sample influences update

**Jacboians (rmb to remove cols you don't need). J=[] is just example**

**Translation eg $I'(x,y) = I(x+3, y-2)$**

$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix}$

$J = \begin{bmatrix} \frac{\partial x'}{\partial t_x} & \frac{\partial x'}{\partial t_y} \\ \frac{\partial y'}{\partial t_x} & \frac{\partial y'}{\partial t_y} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

**Euclidean eg $I'(x,y) = I(-y,x)$**

$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos(\theta) - y\sin(\theta) + t_x \\ x\sin(\theta) + y\cos(\theta) + t_y \end{bmatrix}$

$J = \begin{bmatrix} \frac{\partial x'}{\partial t_x} & \frac{\partial x'}{\partial t_y} & \frac{\partial x'}{\partial \theta} \\ \frac{\partial y'}{\partial t_x} & \frac{\partial y'}{\partial t_y} & \frac{\partial y'}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -x\sin(\theta) - y\cos(\theta) \\ 0 & 1 & x\cos(\theta) - y\sin(\theta) \end{bmatrix}$

**Similarity**

$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s\cos(\theta) & -s\sin(\theta) & t_x \\ s\sin(\theta) & s\cos(\theta) & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} sx\cos(\theta) - sy\sin(\theta) + t_x \\ sx\sin(\theta) + sy\cos(\theta) + t_y \end{bmatrix}$

$J = \begin{bmatrix} \frac{\partial x'}{\partial t_x} & \frac{\partial x'}{\partial t_y} & \frac{\partial x'}{\partial s} & \frac{\partial x'}{\partial \theta} \\ \frac{\partial y'}{\partial t_x} & \frac{\partial y'}{\partial t_y} & \frac{\partial y'}{\partial s} & \frac{\partial y'}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & x\cos(\theta) - y\sin(\theta) & -sx\sin(\theta) - sy\cos(\theta) \\ 0 & 1 & x\sin(\theta) + y\cos(\theta) & -sx\cos(\theta) - sy\sin(\theta) \end{bmatrix}$

**Affine eg $I'(x,y) = I(x + 0.3y + 3, 0.5x + y - 2)$**

$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} p_1 & p_3 & t_x \\ p_2 & p_4 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} p_1x + p_3y + t_x \\ p_2x + p_4y + t_y \end{bmatrix}$

$J = \begin{bmatrix} \frac{\partial x'}{\partial t_x} & \frac{\partial x'}{\partial t_y} & \frac{\partial x'}{\partial p_1} & \cdots & \frac{\partial x'}{\partial_4} \\ \frac{\partial y'}{\partial t_x} & \frac{\partial y'}{\partial t_y} & \frac{\partial y'}{\partial p_2} & \cdots & \frac{\partial x'}{\partial p_4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & x & 0 & y & 0 \\ 0 & 1 & 0 & x & 0 & y \end{bmatrix}$

A, D are indicative of the aperture problem – they are all edge regions in which the resulting flow is perpendicular to the edge. B & C are correct; magnitude of the flow vectors seem high but this is likely due to scaling

E – flow is correct – even though wall is not moving, the shadows create an "apparent" motion. This is not so visible due to the scaling in ground truth

F – the swirl pattern of flow is incorrect, this is likely due to strong smoothing constraints from HS flow

Lukas-Kanade has sparse optical flow, HS has dense optical flow with smooth flow fields

•Sparse-looking: B, C, F. Rule out B since flow vectors are still regular; so just C and F remaining. F is noisier and more flow vectors, so has the lower threshold for R;

•Of remaining A, B, D, E for Horn-Shunck; A and D have more iterations – cannot recover the fine movements of the low-gradient areas such as the fabric background without running many iterations

•Smaller lambda, smoother flow-fields so A is large lambda, B locally smoother than E.