Image Sequence (or Motion) Analysis

also referred as:

Dynamic Scene Analysis
Input: A Sequence of Images (or frames)

Output: Motion parameters (3-D) of the objects/s in motion  

and/or  

3-D Structure information of the moving objects  

Assumption: The time interval $\Delta T$ between two consecutive or successive pair of frames is small.
Typical steps in Dynamic Scene Analysis

- Image filtering and enhancement
- Tracking
- Feature detection
- Establish feature correspondence
- Motion parameter estimation (local and global)
- Structure estimation (optional)
- Predict occurrence in next frame
Since $\Delta T$ is small, the amount of motion/displacement of objects, between two successive pair of frames is also small.

At 25 frames per sec (fps):
$\Delta T = 40$ msec.

At 50 frames per sec (fps):
$\Delta T = 20$ msec.
**Image Motion**

Image changes by difference equation:

\[ f_d(x_1, x_2, t_i, t_j) = f(x_1, x_2, t_i) - f(x_1, x_2, t_j) = f(t_i) - f(t_i) = f_i - f_j \]

Accumulated difference image:

\[ f_T(X, t_n) = f_d(X, t_{n-1}, t_n) - f_T(X, t_{n-1}); n \geq 3, \]

where, \[ f_T(X, t_2) = f_d(X, t_2, t_1) \]

Moving Edge (or feature) detector:

\[ F_{mov\_feat}(X, t_1, t_2) = \left| \frac{\partial f}{\partial X} \right| f_d(X, t_1, t_2) \]

Recent methods include background and foreground modeling.
Two categories of Visual Tracking Algorithms:
Target Representation and Localization; Filtering and Data Association.

A. Some common Target Representation and Localization algorithms:

Blob tracking: Segmentation of object interior (for example blob detection, block-based correlation or optical flow-KLT)

Kernel-based tracking (Mean-shift tracking): An iterative localization procedure based on the maximization of a similarity measure (Bhattacharyya coefficient).

Contour tracking: Detection of object boundary (e.g. active contours or Condensation algorithm)

Visual feature matching: Registration; RANSAC, DLT

B. Some common Filtering and Data Association algorithms:

Kalman filter: An optimal recursive Bayesian filter for linear functions and Gaussian noise (FTLE).

Particle filter: Useful for sampling the underlying state-space distribution of non-linear and non-Gaussian processes.

Also see: Match moving; Motion capture; Swistrack, BOVW, Steak flow, tracklets, spatio-temporal gradients, LCS, LTDS, MRF, LDA, RFT, LCSS, MDA, DFM, Dynamic textures, BOAW, HFST, SRC based MHOF, LBPTOPS, HOP; also Manifold and Deep learning.
<table>
<thead>
<tr>
<th>Method</th>
<th>Marathon</th>
<th>Mecca</th>
<th>Pedestrians</th>
<th>Pilgrims</th>
<th>Crosswalk</th>
<th>Roundabout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optical flow + DBSCAN</td>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
</tr>
<tr>
<td>Particle flow + FTLE field Segmentation</td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
<td><img src="image9" alt="Image" /></td>
<td><img src="image10" alt="Image" /></td>
<td><img src="image11" alt="Image" /></td>
<td><img src="image12" alt="Image" /></td>
</tr>
<tr>
<td>Streak flow + Watershed Segmentation</td>
<td><img src="image13" alt="Image" /></td>
<td><img src="image14" alt="Image" /></td>
<td><img src="image15" alt="Image" /></td>
<td><img src="image16" alt="Image" /></td>
<td><img src="image17" alt="Image" /></td>
<td><img src="image18" alt="Image" /></td>
</tr>
<tr>
<td>Coherent filtering</td>
<td><img src="image19" alt="Image" /></td>
<td><img src="image20" alt="Image" /></td>
<td><img src="image21" alt="Image" /></td>
<td><img src="image22" alt="Image" /></td>
<td><img src="image23" alt="Image" /></td>
<td><img src="image24" alt="Image" /></td>
</tr>
<tr>
<td>Metatracking</td>
<td><img src="image25" alt="Image" /></td>
<td><img src="image26" alt="Image" /></td>
<td><img src="image27" alt="Image" /></td>
<td><img src="image28" alt="Image" /></td>
<td><img src="image29" alt="Image" /></td>
<td><img src="image30" alt="Image" /></td>
</tr>
</tbody>
</table>
**BLOB Detection** – Approaches used:

- **Corner detectors**
  (Harris, Shi & Tomasi, Susan, Level Curve Curvature, Fast etc.)

- **Ridge detection**, **Scale-space Pyramids**

- **LOG, DOG, DOH** (Det. Of Hessian)

- **Hessian affine, SIFT** (Scale-invariant feature transform)

- **SURF** (Speeded Up Robust Features)

- **GLOH** (Gradient Location and Orientation Histogram)

- **LESH** (Local Energy based Shape Histogram).

Complexities and issues in tracking:

Need to overcome difficulties that arise from **noise, occlusion, clutter, moving cameras, multiple moving objects and changes in the foreground objects or in the background environment.**
DOH - the scale-normalized determinant of the Hessian, also referred to as the Monge–Ampère operator,

$$\det HL(x, y; t) = t^2(L_{xx}L_{yy} - L_{xy}^2)$$

where, $HL$ denotes the Hessian matrix of $L$ and then detecting scale-space maxima/minima of this operator one obtains another straightforward differential blob detector with automatic scale selection which also responds to saddles.

$$\left(\hat{x}, \hat{y}; \hat{t}\right) = \arg\max_{(x, y; t)} \min_{\text{local}}(\det HL(x, y; t))$$

Hessian Affine:

$$H(x) = \begin{bmatrix} L_{xx}(x) & L_{xy}(x) \\ L_{xy}(x) & L_{yy}(x) \end{bmatrix}$$

SURF is based on a set of 2-D HAAR wavelets; implements DOH
SIFT:
Detect extremas at various scales:

Four major steps:
1. Scale-space extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Only three steps (1, 3 & 4) are shown below:

\[ D(x, y, \sigma) = L(x, y, k_i \sigma) - L(x, y, k_j \sigma) \]
\[ L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) \]
\[ \theta(x, y) = \tan^{-1}\left( \frac{L(x, y + 1) - L(x, y - 1)}{L(x + 1, y) - L(x - 1, y)} \right) \]
\[ m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2} \]

Histograms contain 8 bins each, and each descriptor contains an array of 4 histograms around the keypoint. This leads to a SIFT feature vector with \((4 \times 4 \times 8 = 128 \text{ elements})\).
GLOH (Gradient Location and Orientation Histogram)

Gradient location-orientation histogram (GLOH) is an extension of the SIFT descriptor designed to increase its robustness and distinctiveness. The SIFT descriptor is computed for a log-polar location grid with 3 bins in radial direction (the radius set to 6, 11 and 15) and 8 in angular direction, which results 17 location bins.

Note that the central bin is not divided in angular directions. The gradient orientations are quantized in 16 bins. This gives a 272 bin histogram.

The size of this descriptor is reduced with PCA. The covariance matrix for PCA is estimated on 47000 image patches collected from various images. The 128 largest eigenvectors are used for description.

Gradient location and orientation histogram (GLOH) is a new descriptor which extends SIFT by changing the location grid and using PCA to reduce the size.
Motion Equations:

Models derived from mechanics.
Euler's or Newton's equations:

\[ \mathbf{P}' = \mathbf{R} \mathbf{P} + \mathbf{T}, \]
where:

\[
m_{11} = n_1^2 + (1 - n_1^2) \cos \theta \\
m_{12} = n_1 n_2 (1 - \cos \theta) - n_3 \sin \theta \\
m_{13} = n_1 n_3 (1 - \cos \theta) + n_2 \sin \theta \\
m_{21} = n_1 n_2 (1 - \cos \theta) + n_3 \sin \theta \\
m_{22} = n_2^2 + (1 - n_2^2) \cos \theta \\
m_{23} = n_2 n_3 (1 - \cos \theta) - n_1 \sin \theta \\
m_{31} = n_1 n_3 (1 - \cos \theta) - n_2 \sin \theta \\
m_{32} = n_2 n_3 (1 - \cos \theta) + n_1 \sin \theta \\
m_{33} = n_3^2 + (1 - n_3^2) \cos \theta
\]

\[ \mathbf{R} = \begin{bmatrix} m_{ij} \end{bmatrix}_{3 \times 3} \]

\[ \mathbf{T} = \begin{bmatrix} \partial x & \partial y & \partial z \end{bmatrix}^T \]

where: \[ n_1^2 + n_2^2 + n_3^2 = 1 \]

Observation in the image plane:

\[ \mathbf{U} = \mathbf{X}' - \mathbf{X}; \mathbf{V} = \mathbf{Y}' - \mathbf{Y}. \]

\[ X = Fx / z; Y = Fy / z. \]

\[ X' = Fx' / z'; Y = Fy' / z'. \]
Mathematically (for any two successive frames), the problem is:

**Input:** Given \((X, Y), (X', Y')\);

**Output:** Estimate \(n_1, n_2, n_3, \theta, \delta x, \delta y, \delta z\)

First look at the problem of estimating motion parameters using 3D knowledge only:

Given only **three (3) non-linear equations**, you have to obtain **seven (7) parameters**.

Need a few more constraints and may be assumptions too.

Since \(\Delta T\) is small, \(\theta\) must also be small enough (in radians).

Thus \(R\) simplifies (reduces) to:

\[
R \bigg|_{\theta \to 0}
\]

where

\[
Evaluating these \(3 \times 3\) \text{ matrices gives the six parameters.}
\]
Take two point correspondences: \( P'_1 = R.P_1 + T; \ P'_2 = R.P_2 + T; \)

Subtracting one from the other, gives: 
(eliminates the translation component) 

\[
\begin{bmatrix}
\Delta x'_{12} \\
\Delta y'_{12} \\
\Delta z'_{12}
\end{bmatrix} =
\begin{bmatrix}
1 & -\phi_3 & \phi_2 \\
\phi_3 & 1 & -\phi_1 \\
-\phi_2 & \phi_1 & 1
\end{bmatrix}
\begin{bmatrix}
\Delta x_{12} \\
\Delta y_{12} \\
\Delta z_{12}
\end{bmatrix}, \quad \text{Solve for:} \quad \Phi = \begin{bmatrix}
\phi_1 \\
\phi_2 \\
\phi_3
\end{bmatrix};
\]

where, \( \Delta x_{12} = x_1 - x_2, \) \( \Delta x'_{12} = x'_1 - x'_2; \) 

and so on .... for y and z.

Re-arrange to form: 

\[
\begin{bmatrix}
\phi_1 \\
\phi_2 \\
\phi_3
\end{bmatrix}
\begin{bmatrix}
\nabla_{12}
\end{bmatrix}
= \Delta^2_{12} \quad \text{and} \quad \Delta^2_{12} =
\begin{bmatrix}
\Delta x'_{12} - \Delta x_{12} \\
\Delta y'_{12} - \Delta y_{12} \\
\Delta z'_{12} - \Delta z_{12}
\end{bmatrix}
\]
\[
\begin{bmatrix}
\nabla_{12}
\end{bmatrix}
\begin{bmatrix}
\phi_1 \\
\phi_2 \\
\phi_3 
\end{bmatrix} = \Delta^2_{12}
\]

\[
\nabla_{12} = \begin{bmatrix}
\Delta z_{12} & -\Delta y_{12} \\
-\Delta z_{12} & \Delta x_{12} \\
\Delta y_{12} & -\Delta x_{12} 
\end{bmatrix}
\]

and \(\Delta^2_{12}\) = \[
\begin{bmatrix}
\Delta x'_{12} - \Delta x_{12} \\
\Delta y'_{12} - \Delta y_{12} \\
\Delta z'_{12} - \Delta z_{12} 
\end{bmatrix}
\]

\(\nabla_{12}\) is a skew-symmetric matrix.

\(\left| \nabla_{12} \right| = 0\)

So what to do?

Contact a Mathematician?

Interprete, why is it so?

Take two (one pair) more point correspondences:

\[
\begin{bmatrix}
\nabla_{34}
\end{bmatrix}
\begin{bmatrix}
\phi_1 \\
\phi_2 \\
\phi_3 
\end{bmatrix} = \Delta^2_{34}
\]

\[
\nabla_{34} = \begin{bmatrix}
\Delta z_{34} & 0 & \Delta x_{34} \\
\Delta y_{34} & -\Delta x_{34} & 0 \\
0 & \Delta z_{34} & -\Delta y_{34} 
\end{bmatrix}
\]

and \(\Delta^2_{34}\) = \[
\begin{bmatrix}
\Delta y'_{34} - \Delta y_{34} \\
\Delta z'_{34} - \Delta z_{34} \\
\Delta x'_{34} - \Delta x_{34} 
\end{bmatrix}
\]
Using two pairs (4 points) of correspondences:

\[
\begin{bmatrix}
\nabla_{12} \\
\nabla_{34}
\end{bmatrix}
\begin{bmatrix}
\phi_1 \\
\phi_2 \\
\phi_3
\end{bmatrix} = \Delta^2_{12} \quad \begin{bmatrix}
\nabla_{12} \\
\nabla_{34}
\end{bmatrix}
\begin{bmatrix}
\phi_1 \\
\phi_2 \\
\phi_3
\end{bmatrix} = \Delta^2_{34}
\]

Adding:

\[
\begin{bmatrix}
\nabla_{12} + \nabla_{34}
\end{bmatrix}
\begin{bmatrix}
\phi_1 \\
\phi_2 \\
\phi_3
\end{bmatrix} = [\Delta^2_{12} + \Delta^2_{34}] \Rightarrow \begin{bmatrix}
\nabla_{1234}
\end{bmatrix}
\begin{bmatrix}
\Phi
\end{bmatrix} = [\Delta^2_{1234}]
\]

\[
\nabla_{12} = \begin{bmatrix}
0 & \Delta z_{12} & -\Delta y_{12} \\
-\Delta z_{12} & 0 & \Delta x_{12} \\
\Delta y_{12} & -\Delta x_{12} & 0
\end{bmatrix}
\]

\[
\nabla_{34} = \begin{bmatrix}
-\Delta z_{34} & 0 & \Delta x_{34} \\
\Delta y_{34} & -\Delta x_{34} & 0 \\
0 & \Delta z_{34} & -\Delta y_{34}
\end{bmatrix}
\]

And \(\Delta^2 = \begin{bmatrix}
\Delta x'_{12} - \Delta x_{12} \\
\Delta y'_{12} - \Delta y_{12} \\
\Delta z'_{12} - \Delta z_{12}
\end{bmatrix}\) and \(\Delta^2_{34} = \begin{bmatrix}
\Delta y'_{34} - \Delta y_{34} \\
\Delta z'_{34} - \Delta z_{34} \\
\Delta x'_{34} - \Delta x_{34}
\end{bmatrix}\)

\[
\nabla_{1234} = \begin{bmatrix}
-\Delta z_{34} & \Delta z_{12} & \Delta x_{34} - \Delta y_{12} \\
\Delta y_{34} - \Delta z_{12} & -\Delta x_{34} & \Delta x_{12} \\
\Delta y_{12} & -\Delta x_{12} - \Delta z_{34} & -\Delta y_{34}
\end{bmatrix}
\]

; \(\Delta^2_{1234} = \begin{bmatrix}
\Delta y'_{34} - \Delta y_{34} + \Delta x'_{12} - \Delta x_{12} \\
\Delta z'_{34} - \Delta z_{34} + \Delta y'_{12} - \Delta y_{12} \\
\Delta x'_{34} - \Delta x_{34} + \Delta z'_{12} - \Delta z_{12}
\end{bmatrix}\)
\[
\begin{bmatrix}
\n\n\n\n\Delta_1 + \Delta_2 + \Delta_3 + \Delta_4 \\
\n\Delta_1 + \Delta_2 + \Delta_3 + \Delta_4 \\
\n\Delta_1 + \Delta_2 + \Delta_3 + \Delta_4 \\
\n\end{bmatrix}
\]

Condition for existence of a unique solution is based on a geometrical relationship of the coordinates of four points in space, at time \( t_1 \):

\[
\nabla_{1234} = \begin{bmatrix}
-\Delta z_{34} & \Delta z_{12} & \Delta x_{34} - \Delta y_{12} \\
\Delta y_{34} - \Delta z_{12} & -\Delta x_{34} & \Delta x_{12} \\
\Delta y_{12} & -\Delta x_{12} - \Delta z_{34} & -\Delta y_{34}
\end{bmatrix}
\]

and \( \Delta^2_{1234} = \begin{bmatrix}
\Delta y'_{34} - \Delta y_{34} + \Delta x'_{12} - \Delta x_{12} \\
\Delta z'_{34} - \Delta z_{34} + \Delta y'_{12} - \Delta y_{12} \\
\Delta x'_{34} - \Delta x_{34} + \Delta z'_{12} - \Delta z_{12}
\end{bmatrix} \]

This solution is often used as an initial guess for the final estimate of the motion parameters. Find geometric condition, when: \( \nabla_{1234} = 0 \)
A point in 3D space:

\[ X_0 = \begin{bmatrix} kx_o & ky_o & kz_o & k \end{bmatrix}; k \neq 0, \text{ an arbitrary constant.} \]

Image point:

\[ X_i = \begin{bmatrix} wx_i & wy_i & w \end{bmatrix} \quad \text{where,} \quad X_i = PX_0 \]

Assuming normalized focal length, f = 1:

\[
\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x_o/z_o \\ y_o/z_o \end{bmatrix} \quad \text{....(1)}
\]

Assuming \textit{linear} motion model (no acceleration), between successive frames:

\[
\begin{bmatrix} x_o(t) \\ y_o(t) \\ z_o(t) \end{bmatrix} = \begin{bmatrix} x_o + ut \\ y_o + vt \\ z_o + wt \end{bmatrix} \quad \text{....(2)}
\]

Combining equations (1) and (2):

\[
\begin{bmatrix} x_i(t) \\ y_i(t) \end{bmatrix} = \begin{bmatrix} (x_o + ut)/(z_o + wt) \\ (y_o + vt)/(z_o + wt) \end{bmatrix} \quad \text{....(3)}
\]
Assume in equation (3), that \( w (=dz/dt) < 0 \).

\[
\begin{bmatrix}
x_i(t) \\
y_i(t)
\end{bmatrix} = \begin{bmatrix}
(x_o + ut) \\
(y_o + vt) \\
(z_o + wt)
\end{bmatrix} \quad \text{....(3)}
\]

In that case, the object (or points on the object) will appear to come closer to you. Now visualize, where does these points may come out from?

\[
\lim_{t \to -\infty} \begin{bmatrix}
x_i(t) \\
y_i(t)
\end{bmatrix} = \begin{bmatrix}
\infty \\
\infty
\end{bmatrix}
\]

This point “\( e \)” is a point in the image plane, known as the:

**FOE (Focus of Expansion).**

The motion of the object points appears to emanate from a fixed point in the image plane. This point is known as **FOE**.

Approaches to calculate FOE are based on the exploitation of the fact that for constant velocity object motion all image plane flow vectors intersect at the FOE.

Plot the vectors and extrapolate them to obtain FOE.
Image Plane

FOE
FOE may not exist for all types of motion – say pure ROTATION, as shown below.

Multiple FOE’s may exist for multiple object motion and occlusion.
Depth from Motion

Time varying distance, $D(t)$, in the image plane, is the distance of an image point from the FOE:

$$D(t) = \| X_i - e \| = \sqrt{[x_i(t) - \frac{u}{w}]^2 + [y_i(t) - \frac{v}{w}]^2}$$

$$Lt \quad D(t) = 0 \quad t \to -\infty$$

Rate of change of distance $D(t)$ is:

$$V(t) = \frac{d[D(t)]}{dt}$$

Derive this to obtain (home assignment):

$$V(t) = \frac{d[D(t)]}{dt} = -\frac{w.D(t)}{z_o(t)}$$

This helps to define, TIME-TO-ADJACENCY equation:

$$T_A = \frac{D(t)}{V(t)} = -\frac{z(t)}{w}$$
Assuming \( z \) is +ve and \( w \) is –ve. \( D(t) \) is different for different pixels.

This equation holds for each corresponding object and image point pair.

Consider two object points, \( z_1(t) \) and \( z_2(t) \). Then:

\[
T_A = \frac{D(t)}{V(t)} = -\frac{z(t)}{w}
\]

\[
\frac{D_1(t)}{V_1(t)} = -\frac{z_1(t)}{w}, \quad \frac{D_2(t)}{V_2(t)} = -\frac{z_2(t)}{w}; \implies z_2(t) = z_1(t) \begin{bmatrix} \frac{D_2(t)}{D_1(t)} & \frac{V_1(t)}{V_2(t)} \end{bmatrix}
\]

\( D_i(t) \) and \( V_i(t) \) values for any object point can be obtained from the image plane, once \( e \) (FOE) is obtained.

Hence it is possible to determine the relative 3-D depths:

\[
\frac{Z_2(t)}{Z_1(t)}
\]

This is the key idea of Structure from motion (SFM) problem, and you are able to extract the shape (structure) information of the object in motion up to a certain scale factor, from a single perspective view only.
Another important idea of optical flow is based on the Horn’s (Horn-Schunk, 1980) equations. A global energy function is sought to be minimized, whose functional form is given as:

\[
f = \int ((\nabla I \cdot \vec{V} + I_t)^2 + \alpha (|\nabla V_x|^2 + |\nabla V_y|^2 + |\nabla V_z|^2)) \, dx \, dy \, dz
\]

\[
V_{x}^{k+1} = \frac{I_x V_{x}^{k}}{\alpha^2 + I_x^2 + I_y^2 + I_z^2} - \frac{I_y V_{y}^{k}}{\alpha^2 + I_x^2 + I_y^2 + I_z^2} + \frac{I_z V_{z}^{k}}{\alpha^2 + I_x^2 + I_y^2 + I_z^2} + I_t
\]

KLT tracker:

\[
\frac{\partial I}{\partial x} V_x + \frac{\partial I}{\partial y} V_y + \frac{\partial I}{\partial z} V_z + \frac{\partial I}{\partial t} = 0 \quad \text{or,} \quad \nabla I \cdot \vec{V} = -I_t
\]

\[
\begin{bmatrix}
V_x \\
V_y \\
V_z
\end{bmatrix} = \begin{bmatrix}
\sum I_{x_i}^2 & \sum I_{x_i} I_{y_i} & \sum I_{x_i} I_{z_i} \\
\sum I_{y_i} I_{x_i} & \sum I_{y_i}^2 & \sum I_{y_i} I_{z_i} \\
\sum I_{z_i} I_{x_i} & \sum I_{z_i} I_{y_i} & \sum I_{z_i}^2
\end{bmatrix}^{-1} \begin{bmatrix}
-\sum I_{x_i} I_{t_i} \\
-\sum I_{y_i} I_{t_i} \\
-\sum I_{z_i} I_{t_i}
\end{bmatrix}
\]


Concepts from above are left for self-study .........
KLT- difference between $F(x + h)$ and $G(x)$

$$
F'(x) \approx \frac{F(x + h) - F(x)}{h} = \frac{G(x) - F(x)}{h}
$$

$$
\begin{cases}
  h_0 = 0 \\
  h_{k+1} = h_k + \frac{\sum_x w(x) [G(x) - F(x + h_k)]}{\sum_x w(x) F'(x + h_k)}
\end{cases}
$$

$$
\begin{cases}
  h_0 = 0 \\
  h_{k+1} = h_k + \frac{\sum_x w(x) F'(x + h_k) [G(x) - F(x + h_k)]}{\sum_x w(x) F'(x + h_k)^2}
\end{cases}
$$
\[ h \approx \left[ \sum_x [G(x) - F(x)] \left( \frac{\partial F}{\partial x} \right) \right] \left[ \sum_x \left( \frac{\partial F}{\partial x} \right)^T \left( \frac{\partial F}{\partial x} \right) \right]^{-1}, \]

\[ G(x) = F(Ax + h), \]

\[ F(x(A + \Delta A) + (h + \Delta h)) \approx F(Ax + h) + (\Delta Ax + \Delta h) \frac{\partial}{\partial x} F(x). \]

https://www.ces.clemson.edu/~stb/klt/
Motion Analysis using rigid body assumption

Rigid Body Assumption:
\[ \|x_i - x_j\|^2 = c_{ij}, \forall t, \forall (i, j), \text{where } c_{ij} \text{ are constants.} \]

Motion Equation:
\[ X(t_2) = M . X(t_1), \text{where, } M = \begin{bmatrix} m_{11} & m_{12} & m_{13} & l_1 \\ m_{21} & m_{22} & m_{23} & l_2 \\ m_{31} & m_{32} & m_{33} & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, m_{ij} = f(n_1, n_2, n_3, \theta). \]

\[ X(t_i) = \begin{bmatrix} x(t_i) \\ y(t_i) \\ z(t_i) \\ 1 \end{bmatrix}^T \]

Form matrix \( A(t_i) \), using four points as:
\[ A(t_i) = \begin{bmatrix} X_1(t_i) & X_2(t_i) & X_3(t_i) & X_4(t_i) \end{bmatrix} \]

Thus obtain the matrix \( M \), using:
\[ M = A(t_j) . A(t_i)^{-1} \]

Points must be selected in such a fashion that \( A(t_i) \) is a non-singular matrix.

Can you guess, when \( A(t_i) \) will be singular?
After $m_{ij}$’s are obtained:

$$\cos(\theta) = \frac{m_{11} + m_{22} + m_{33} - 1}{2}; \quad \sin(\theta) = \frac{m_{32} - m_{23}}{2n_1}.$$

$$n_1 = \sqrt{\frac{m_{11} - \cos(\theta)}{1 - \cos(\theta)}}, \quad n_2 = \frac{m_{21} + m_{12}}{2n_1(1 - \cos(\theta))}, \quad n_3 = \frac{m_{31} + m_{13}}{2n_1(1 - \cos(\theta))}.$$

This is fine in an ideal case. In noisy situations (or even with numerical errors):

$$A(t_i) = M.A(t_j) + N_{ij}.$$ 

Need to formulate an optimization function to minimize a cost function, to satisfy equations in the least square sense:

$$X_k(t_i) = M.X_k(t_j), \quad k = 1,2,\ldots,K$$

For example, minimize:

$$\sum_{k=1}^{K} \left[ X_k(t_i) - M.X_k(t_j) \right]^2$$

along with **Rigidity constraint**.

Use linearized solution as your initial estimate.

Heard about **SA or GA**?
Work out the following (2nd method):

\[ X_k(t_i) = R.X_k(t_j), \quad k = 1, 2, \ldots, K \]

where,

\[ R = \begin{bmatrix} R_p & T \\ 0 & 1 \end{bmatrix} \]

\[ R_p = R_\alpha R_\beta R_\lambda \]

Again, we have 12 unknown elements in \( R \), which are functions of six unknown parameters.

First obtain the 12 unknown elements and then get the six parameters.
Motion Analysis

using

Image plane coordinates of the features of the moving object.
\[ P_k(t_i) = R P_k(t_j), \quad k = 1, 2, \ldots, K \]

\[ P_k = \begin{bmatrix} x_k & y_k & z_k & 1 \end{bmatrix} \]

\[ x_{k2} = r_{11} x_{k1} + r_{12} y_{k1} + r_{13} z_{k1} + t_x; \]

\[ y_{k2} = r_{21} x_{k1} + r_{22} y_{k1} + r_{23} z_{k1} + t_y; \]

\[ z_{k2} = r_{31} x_{k1} + r_{32} y_{k1} + r_{33} z_{k1} + t_z; \]

**Projective Equations:**

\[ X_k(t) = \frac{x_k}{z_k} \]

\[ x_{k2} = (r_{11} X_{k1} + r_{12} Y_{k1} + r_{13}) z_{k1} + t_x; \]

\[ Y_k(t) = \frac{y_k}{z_k} \]

\[ y_{k2} = (r_{21} X_{k1} + r_{22} Y_{k1} + r_{23}) z_{k1} + t_y; \]

\[ Z_k(t) = \frac{z_k}{z_k} \]

\[ z_{k2} = (r_{31} X_{k1} + r_{32} Y_{k1} + r_{33}) z_{k1} + t_z; \]

\[ X_{k2} = \frac{x_{k2}}{z_{k2}} = \frac{(r_{11} X_{k1} + r_{12} Y_{k1} + r_{13}) z_{k1} + t_x}{(r_{31} X_{k1} + r_{32} Y_{k1} + r_{33}) z_{k1} + t_z}; \]

\[ Y_{k2} = \frac{y_{k2}}{z_{k2}} = \frac{(r_{21} X_{k1} + r_{22} Y_{k1} + r_{23}) z_{k1} + t_y}{(r_{31} X_{k1} + r_{32} Y_{k1} + r_{33}) z_{k1} + t_z}; \]
\[ X_{k2} = \frac{x_{k2}}{z_{k2}} = \frac{(r_{11}X_{k1} + r_{12}Y_{k1} + r_{13})z_{k1} + t_x}{(r_{31}X_{k1} + r_{32}Y_{k1} + r_{33})z_{k1} + t_z}; \]
\[ Y_{k2} = \frac{y_{k2}}{z_{k2}} = \frac{(r_{21}X_{k1} + r_{22}Y_{k1} + r_{23})z_{k1} + t_y}{(r_{31}X_{k1} + r_{32}Y_{k1} + r_{33})z_{k1} + t_z}; \]

Solve for \( Z_{k1} \), from the above two equations to obtain:

\[
\begin{bmatrix} X_{k2} & Y_{k2} & 1 \end{bmatrix} \begin{bmatrix} X_{k1} \\ Y_{k1} \\ 1 \end{bmatrix} = 0;
\]

After manipulation obtain:

\[ \overrightarrow{c}^T \mathbb{e} = -1 \]

where:

\[ \overrightarrow{c} = [X_{k2}X_{k1}, X_{k2}Y_{k1}, X_{k2}, Y_{k2}X_{k1}, Y_{k2}Y_{k1}, Y_{k2}, X_{k1}Y_{k1}, X_{k2}Y_{k2}] \]

\( \mathbb{e} \) is a vector (matrix) of 8 essential parameters, called the essential matrix. Solve to get \( \mathbb{e} \) first.

Hence we require 8 point correspondences in this case to obtain the (i) essential parameters first and then (ii) the motion parameters.
Environment similar to the Human Vision System (minus brain-power)
Motion equations for dynamic stereo:

Let $A_l$ and $A_r$ be the perspective transformation matrices for the pair of stereo cameras. $R$ be the composite motion matrix with 12 unknown elements.

$$I_i^l(t_2) = A_lRX_i(t_1); \quad i = 1, 2, ..., N$$
$$I_i^r(t_2) = A_rRX_i(t_1);$$

Each point provides: **eight (8) equations**

Thus $N$ points provide: **$8N$ equations** (linear).

Number of unknowns: **$12 + 3N$**

To provide a solution: $N \geq 3$ Solve for obtaining the optimal solution
Recent Advances:

- Super-resolution from Video
- MPEG standardization and representation
- CVBR – spatio-temporal analysis
- Video Organization - VOD
- Active Vision and Surveillance
- Multi-camera correspondence and depth
- Global motion compensation
- Pose and gesture
- Adaptive Background models – shading and shadows
- Multiple flows (optical flow)
- 3-D Kalman filter, Particle filtering
- Motion patterns
- Deformable motion analysis
- Sensor and view planning
- Kinematics and Rigidity
- Shape from motion
That’s all for now –

Let’s MOVE ON