Structure from Motion – Part II

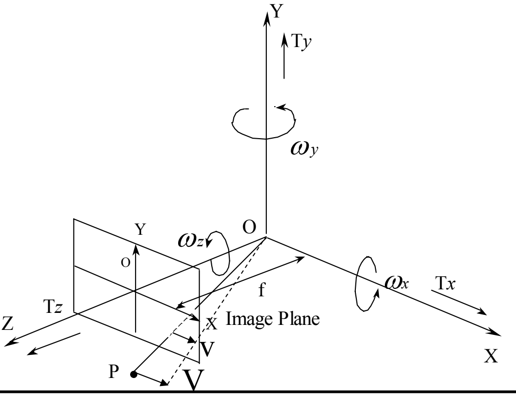
- Given optical flow, recover 3D motion & depth
  - Basic equations
  - Two intuitive but iterative algorithms
  - Closed form algorithm based on parallax
- Appreciate what SFM offers
  - I move, therefore I see
- Appreciate limitations of SFM
  - scale-ambiguity
  - rotation-translation confusion

Structure from Motion – Part II

- Given optical flow, calculate 3D motion and depth
  - Need to relate 3D Motion & depth to 2D optical flow
- Assume a camera moving in a static environment
- Camera motion expressed as a translation and a rotation.

3D Motion of Camera

- **T** = the translational component of the camera motion
- ω = the rotational velocity
- P = the position vector [X Y Z] ^{T}



Relative velocity of P:

$V = -T - \omega \times P$

Relating 3D Motion to 2D Motion Field

$$\text{Perspective Projection}: (x, y) = f\,\frac{(X, Y)}{Z}$$

Taking derivative on both side, we have

x'=f(X'/Z-XZ'/Z^2)

y'=f(Y'/Z-YZ'/Z^2)

$$\text{On LHS, we have } \left(\frac{dx}{dt}, \frac{dy}{dt}\right) \text{ i.e. flow } (v_x, v_y)$$

$$\text{On RHS, we need } \left(\frac{dX}{dt}, \frac{dY}{dt}, \frac{dZ}{dt}\right) \text{ i.e. } (V_x, V_y, V_z)$$

Relating 3D Motion to 2D Motion Field

$$\mathrm{V} = -\mathrm{T} - \omega \times \mathrm{P}, \quad \longleftrightarrow \quad \begin{aligned} V_x &= -T_x - \omega_y Z + \omega_z Y \\ V_y &= -T_y - \omega_z X + \omega_x Z \\ V_z &= -T_z - \omega_x Y + \omega_y X \end{aligned}$$

Substituting,

$$v_x = \frac{T_z x - T_x f}{Z} - \omega_y f + \omega_z y + \frac{\omega_x xy}{f} - \frac{\omega_y x^2}{f}$$
$$v_y = \frac{T_z y - T_y f}{Z} + \omega_x f - \omega_z x - \frac{\omega_y xy}{f} + \frac{\omega_x y^2}{f}$$

Note: In the differential case like motion here, T denotes velocity vector, whereas in the discrete case like stereo, T is a displacement vector.

Relating 3D Motion to 2D Motion Field

$$v_x = \frac{T_z x - T_x f}{Z} - \omega_y f + \omega_z y + \frac{\omega_x xy}{f} - \frac{\omega_y x^2}{f} = (v_x)_{\mathrm{trans}} + (v_x)_{\mathrm{rot}}$$
$$v_y = \frac{T_z y - T_y f}{Z} + \omega_x f - \omega_z x - \frac{\omega_y xy}{f} + \frac{\omega_x y^2}{f} = (v_y)_{\mathrm{trans}} + (v_y)_{\mathrm{rot}}$$

$$(v_x)_{\mathrm{trans}} = \frac{T_z x - T_x f}{Z} \quad ; \quad (v_y)_{\mathrm{trans}} = \frac{T_z y - T_y f}{Z}$$
$$(v_x)_{\mathrm{rot}} = -\omega_y f + \omega_z y + \frac{\omega_x xy}{f} - \frac{\omega_y x^2}{f}$$
$$(v_y)_{\mathrm{rot}} = \omega_x f - \omega_z x - \frac{\omega_y xy}{f} + \frac{\omega_x y^2}{f}$$

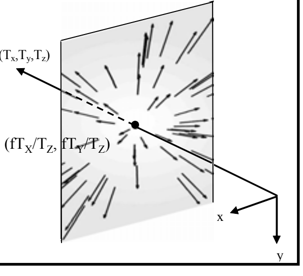
- $(v_x, v_y)_{trans}$ the translational flow contains information about structure of the scene.
- $(v_x, v_y)_{rot}$ the rotational flow is independent of Z.

Pure translation

- When camera motion is only translation, then we have

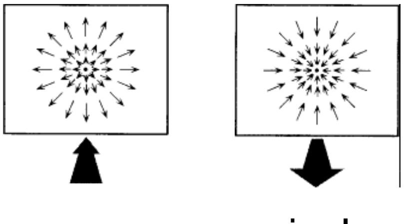
$$\boxed{\omega_x = \omega_y = \omega_z = 0} \longrightarrow \boxed{\begin{aligned} v_x &= \frac{T_z x - T_x f}{Z} \\ v_y &= \frac{T_z y - T_y f}{Z} \end{aligned}} \xrightarrow{\boxed{x_0 = \frac{fT_x}{T_Z}, \; y_0 = \frac{fT_y}{T_Z},}} \boxed{\begin{aligned} v_x &= (x - x_0)\frac{T_z}{Z} \\ v_y &= (y - y_0)\frac{T_z}{Z}. \end{aligned}}$$

- Consider the special point (fT_X/T_Z, fT_Y/T_Z):
  - This is the "image" of the velocity vector onto the image plane. It is located at where the translation vector cuts the image plane.
- The motion at this point must be 0 since the surface point along this ray stays on the ray as the camera moves (our equations evaluate to 0 at this point too)



Pure translation

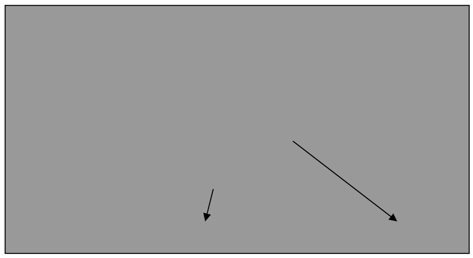
- Consider the direction of the flow $v_x = (x - x_0)\frac{T_z}{Z}$, $v_y = (y - y_0)\frac{T_z}{Z}$ through any point (x,y):
  $v_y/v_x = (y- y_0)/(x- x_0)$
- So this direction must pass through (x_0,y_0). The point (x_0,y_0) is known as the FOE (focus of expansion) or FOC (focus of contraction). All flows emanates from FOE or points towards FOC.



- In stereo context, this point is known as ? epipole.

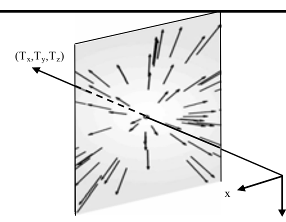
Pure translation

- $T = [0, 0, 1]$; FOE (x_0, y_0) ?
- $T = [1, 0, 0]$; FOE (x_0, y_0) ?



- Where is the FOE given that the 2 flows are purely translational?

Scale ambiguity in T



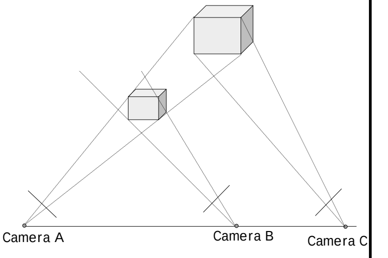
- So if we have optical flow, we can calculate the direction of translation in the form of FOE. But can we recover the absolute magnitude of the 3 components T_X, T_Y, T_Z?
- No, we can only recover T up to a scale ambiguity. This ambiguity is clear from (T_X , T_Y occur in ratio with T_Z).

$$x_0 = \frac{fT_x}{T_Z}$$

$$y_0 = \frac{fT_y}{T_Z},$$

Error in textbook: P185: 5th from bottom: it should be "if $T_Z>0$", not $T_Z<0$, and 4th line from bottom, it should be "if $T_Z<0$", not $T_Z>0$. P186; 3rd line from bottom: it should be "also proportional", not "also inversely proportional".

Scale Ambiguity in Z



- There is also scale ambiguity in Z
  - T_X, T_Y, T_Z occur in ratio with Z.

$$v_x = \frac{T_z x - T_x f}{Z}$$

$$v_y = \frac{T_z y - T_y f}{Z}$$

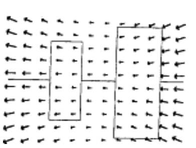
  - Same optic flow field generated by two similar surfaces undergoing similar motions: (T_X, T_Y, T_Z **,** Z) and ($k\, T_X$, $k\, T_Y$, $k\, T_Z$ **,** $k\, Z$).
- If we have computed the FOE of an image sequence then we can compute the (scaled) depth to visible points in the scene

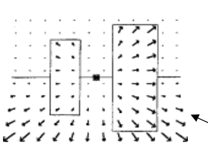
$$v_x = (x - x_0)\frac{T_z}{Z} \Rightarrow \frac{Z}{T_z} = \frac{x - x_0}{v_x}$$

- Since all depths in the scene can only be recovered up to a common scale factor, we sometimes just use $Z/\,T_Z$ (depth scaled by T_Z) as the solution for Z.

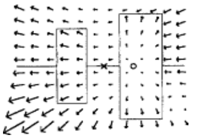
General 3D Motion (SFM)

- So far, we have considered the simple case of pure translation.
  - To solve general 3D motion (with Rotation R and translation T) is a difficult problem!
  - One key problem is the coupling between R & T. Small rotations about the y (x) axis are easy to confuse with translations in x (y).





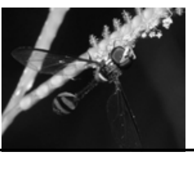
+

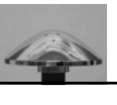


Small rotations about the y axis are easy to confuse with translations in x .

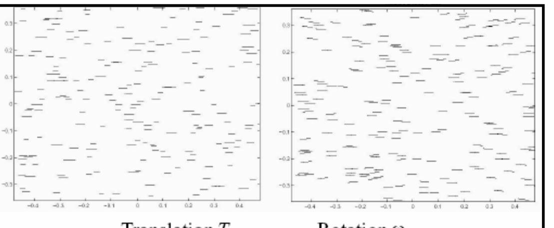
Pure translation along Z-axis.

Can be mistaken as a pure translation heading in a direction slightly off the Z-axis. **NB: FOE is still at the cross.**

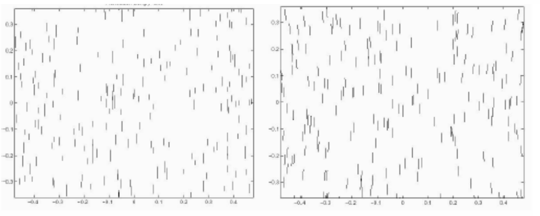




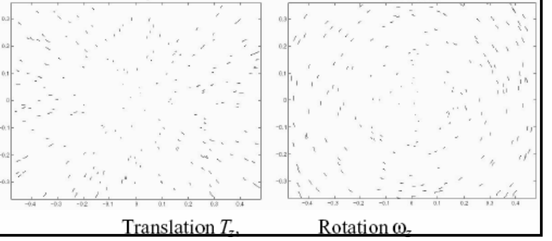
List pair of motions that are hard to separate:



Translation T_x, Rotation ω_y



Translation T_y, Rotation ω_x

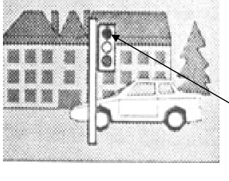


Translation T_z, Rotation ω_z

General 3D Motion (SFM) –

More practical problems in solving SFM

-- Computing optical flow is difficult.

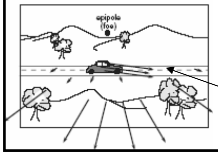


– Optical flow algorithms need to integrate information over small image neighborhoods, assuming smoothness of flow.

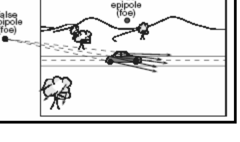
– If those neighborhoods overlap a boundary between an object and the background, smoothness assumptions are violated and the result will be wrong.

-- Independently moving objects confuse 3D motion estimation algorithms

- their motion is inconsistent with the rigid camera motion



Motion field of the moving object is inconsistent with the radial motion field emanating from FOE



Structure from Motion

- What happens if you can't recover the 3D motion perfectly
  - The structure that you perceive will be distorted

Error in Depth Reconstruction . Int'l Journal of Computer Vision, **44** (3), pp 199-217, Aug 2001. © 2001 by Kluwer academic

Behaviour of SFM algorithms . Int'l Journal of Computer Vision, **51** (2), 111-137, 2003. © 2003 Kluwer academic

Solving General SFM

$$\begin{cases} v_x = (x - x_0)\dfrac{T_z}{Z} - \omega_y f + \omega_z y + \dfrac{\omega_x xy}{f} - \dfrac{\omega_y x^2}{f} \\ v_y = (y - y_0)\dfrac{T_z}{Z} + \omega_x f - \omega_z x - \dfrac{\omega_y xy}{f} + \dfrac{\omega_x y^2}{f} \end{cases}$$

- For N image points, there are 2N equations (each point provides 2 optical flow equation) with N+5 unknowns (N depths, 2 for FOE, 3 for rotation).
  - Possible to solve with numerical method but dimension too high.
- Usual method: factor out Z from the 2 equations

$$(v_x - v_x^{Rot},\, v_y - v_y^{Rot}) \cdot (y - y_0, -(x - x_0)) = 0$$

  - N image points; N equations; 5 unknowns $x_0, y_0, \omega_x, \omega_y, \omega_z,$

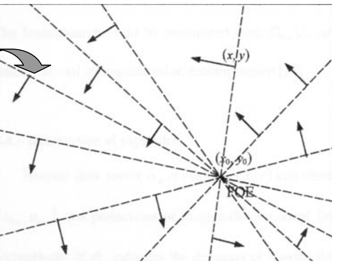
- Essentially, given optical flow, algorithms try to find a set of $x_0, y_0, \omega_x, \omega_y, \omega_z,$ which can minimize

Field of view

$$\sum_{FV}\sum (v_x - v_x^{Rot})(y - y_0) - (v_y - v_y^{Rot})(x - x_0) = 0$$

Simple SFM Algorithm

- Still a five dimensional search. Can further decompose the parameters to reduce the search dimension.
  - 2D search for FOE, obtain rotation in closed form from FOE.
  - 3D search for rotation, obtain FOE in closed form from rotation.
- One way is to first search for the translational parameters (FOE).
  - Each hypothesized FOE defines a set of emanating lines
  - Project optical flow in the direction ⊥ to these lines.
  - It would only contain rotational flow if the FOE is chosen correctly.
  - Fit the 3 rotational parameters (e.g LS) & obtain solution in closed form. Ex: write down the LS equation.
  - check the residual for goodness of fit.



Simple SFM Algorithm II

- Another way is to first search for the 3 rotational parameters. (e.g. Prazdny 80)
  - Given candidate rotation, can remove rotational flow completely

$$(v_x)_{\mathrm{rot}} = -\omega_y f + \omega_z y + \frac{\omega_x xy}{f} - \frac{\omega_y x^2}{f}$$

$$(v_y)_{\mathrm{rot}} = \omega_x f - \omega_z x - \frac{\omega_y xy}{f} + \frac{\omega_x y^2}{f}$$

  - If the rotational parameters are chosen correctly, then after “de-rotation”, all flow field should meet at FOE. Why?
  - Check the intersections of the de-rotated flow and choose rotation such that the dispersion of the intersections is smallest.
- E.g. Given that the rotation is given by (0, 0, 0.1), and the optical flows at the feature points (1,0) and (1,1) is given by (1, -0.1) and (1.1, 0.9) respectively, find the FOE (x0, y0).
- The above methods are conceptually simple, but their solutions require iteration which is time consuming.

Motion Parallax

- Motion parallax: Consider two visual features at different depths whose projections on the image plane are coincident, their relative motion field – motion parallax -- does not depend on the rotational component of motion in 3-D space.
- Relative motion (ie difference) between the 2 flow fields:

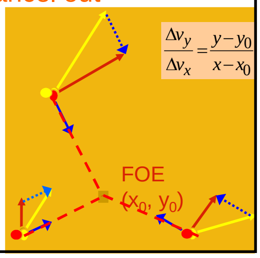
$$\Delta v_x = v_x^{trans} - \bar{v}_x^{trans} = (T_z x - T_x f)\left(\frac{1}{Z} - \frac{1}{\bar{Z}}\right)$$
$$\Delta v_y = v_y^{trans} - \bar{v}_y^{trans} = (T_z y - T_y f)\left(\frac{1}{Z} - \frac{1}{\bar{Z}}\right).$$

the rotational components cancel out

$$\frac{\Delta v_y}{\Delta v_x} = \frac{y - y_0}{x - x_0}$$

direction of motion parallax

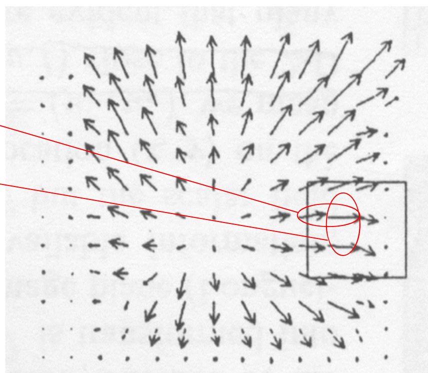
- FOE can be determined.



Motion Parallax

- Problem: not many pairs of points would exactly satisfy the coincidence condition.
- Approximate motion parallax: regard the flow difference between 2 nearby points as noisy estimate of the true motion parallax.

$$\Delta v_{x1}, \Delta v_{y1}$$
$$\Delta v_{x2}, \Delta v_{y2}$$



Approximate Motion Parallax Algorithm

Obtain approximate motion parallax → Compute FOE from approximate motion parallax → Compute rotation & Z from FOE

- At each neighborhood, solve LS Ax =0 using SVD, where x is a unit vector $\perp$ to the parallax $\begin{bmatrix} \Delta v_x \\ \Delta v_y \end{bmatrix}$
- Solve LS $\mathrm{A}' \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \mathrm{b}$ using SVD
- Solve LS $\mathrm{A}'' \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \mathrm{b}'$ using SVD

Motion Parallax

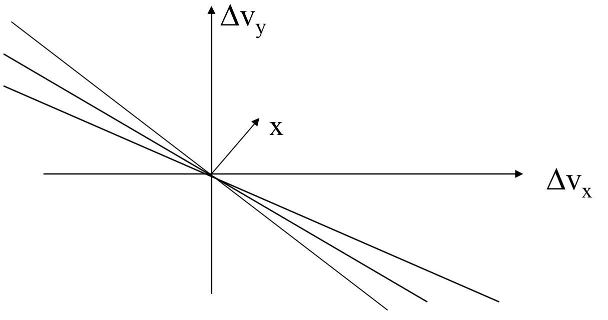
- To find best estimate of parallax ($\Delta v_x, \Delta v_y$) using various noisy estimates ($\Delta v_{xi}, \Delta v_{yi}$). Determine the eigenvalues & eigenvectors of the matrix:

$$\begin{bmatrix} \sum \Delta^2 v_x & \sum \Delta v_x \Delta v_y \\ \sum \Delta v_x \Delta v_y & \sum \Delta^2 v_y \end{bmatrix} \longleftarrow \underline{\text{Design matrix of data fitting problem}}$$

- The eigenvector associated with the greater eigenvalue is the best estimate of the motion parallax within the patch.
  - If rank is 1, data can be fitted perfectly and is reliable. (in other words, degree of freedom/ number of basis / number of principal component is 1)
  - Can use the ratio of the two eigenvalues as a measure of the estimate's reliability.

Interlude: Linear minimization

- Consider the modified problem: finding the direction of x that is most perpendicular to all the parallax



- Form the matrix A, where i^{th} row is given by $(\Delta vx_i, \Delta vy_i)$).
- Amounts to solving Ax=0, for non-zero x.

Interlude: Linear minimization

- Choose x to be the eigenvector associated with the smallest eigenvalue of A^TA. Recall the same result from the section on SVD. Why is this?
- x can only be determined up to a scale, so, choose x to be a unit vector, $||x||=1$.
- We want to find x s.t. $\varepsilon=Ax$ is minimum and $||x||=1$. Lagrange multipliers!
- Define cost $C=|| \varepsilon ||^2 + \lambda (1- ||x||^2)$
- Can be rewritten as $C=x^TA^TAx+ \lambda (1-x^Tx)$
- Find critical points of C, ie, where derivative dC/dx=0

Interlude: Linear minimization

- $dC/dx = 2 A^TAx - 2\lambda x = 0$

  $\Rightarrow A^TAx = \lambda x$
- This is the eigen equation!
- Any eigenvector of A^TA is a solution.
- Choose the eigenvector e_n that minimizes $|| \varepsilon ||^2$

  $|| \varepsilon ||^2 = (e_n^T A^T)(A e_n) = e_n^T (A^TA e_n)$

  $= e_n^T e_n \lambda_n = \lambda_n$

Interlude: Linear minimization

- This is minimized by choosing $x= e_n$ where e_n is the eigenvector associated with the smallest eigenvalue λ_n.
- Our original problem is to find a direction that is most consistent with the direction of the n lines obtained, ie, we want to maximize Ax.
- So to maximize $|| \varepsilon ||^2$, choose $x= e_m$ where e_m is the eigenvector associated with the largest eigenvalue λ_m.

Interlude: Linear minimization

- How to set up A, given n measurements (Δvx_i, Δvy_i), i = 1,2…n ? A consists of n rows, with i^{th} row given by (Δvx_i, Δvy_i).
- We are trying to find a normal x=(a,b) that are perpendicular to these directions. Thus each equation is of the form $a\Delta vx_i + b\Delta vy_i = 0$. (the parallax is in the direction (b,-a).
- This normal x =(a,b) is the eigenvector associated with the smallest eigenvalue of A^TA; the parallax (b,-a) is the eigenvector associated with the largest eigenvalue of A^TA.
- Can solve by eigenvector technique or by solving SVD(A). Columns of V are eigenvectors of A^TA.

$$A^TA \quad \text{is} \quad \begin{bmatrix} \sum \Delta^2 v_x & \sum \Delta v_x \Delta v_y \\ \sum \Delta v_x \Delta v_y & \sum \Delta^2 v_y \end{bmatrix}$$

Back

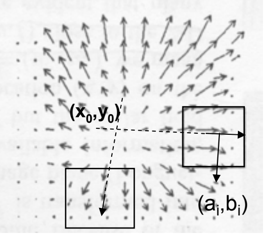
Approximate Motion Parallax Algorithm

1. Obtain approximate motion parallax
   - At each neighborhood, solve LS $Ax = 0$ using SVD, where x is a unit vector $\perp$ to the parallax $\begin{bmatrix} \Delta v_x \\ \Delta v_y \end{bmatrix}$
2. Compute FOE from approximate motion parallax
   - Solve LS $A' \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = b$ using SVD
3. Compute rotation & Z from FOE
   - Solve LS $A'' \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = b'$ using SVD

Motion Parallax

- With several motion parallax computed from N patches, the intersection yields the FOE.

From Block B_i centered at (x_i, y_i), we have used $SVD(A_i)$ to obtain the parallax direction $(b_i,-a_i)$ and consistency measure w_i



parallax direction $(b_i,-a_i)$

- Each block B_i yields an equation $(a_i, b_i) \cdot (x_i - x_0, y_i - y_0)^T = 0$ (the normal to the parallax must be $\perp$ to the emanating lines from FOE).
- Collect equations from all blocks and solve for (x_0, y_0) by LS. Ex. Write down the LS equation.
- Better: use weighted least square. Weight reflects consistency in motion parallax measurement. Each row is weighted (multiplied) by w_i as weight. w_i can be the ratio of the two eigenvalues.

Approximate Motion Parallax Algorithm

Obtain approximate motion parallax

- At each neighborhood, solve LS $Ax = 0$ using SVD, where x is a unit vector $\perp$ to the parallax $\begin{bmatrix} \Delta v_x \\ \Delta v_y \end{bmatrix}$

↓

Compute FOE from approximate motion parallax

- Solve LS $A' \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = b$ using SVD

↓

Compute rotation & Z from FOE

- Solve LS $A'' \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = b'$ using SVD

The rest of the problem (rotation & Z) is easy.
Refer to the intuitive algorithm for the LS equation.

End of Motion Analysis

- Key points:
  - Motion equations relating optical flow & 3D motion & Z
  - Properties of these equations; e.g. scale ambiguity, ambiguity between Rotation & Translation, etc.
  - Given rotation, how to solve FOE, and vice versa
  - Parallax Algorithm

- Follow up Activities: (*: Optional)
  - Revise lecture notes; attempt tutorial.
  - *Additional / self reading: book & classic papers
    - S. Maybank. Theory of Reconstruction from Image Motion, 1993.
    - J.Weng etc. Motion and Structure from Image Sequences, 1993.
    - Longuet -Higgins, H.C. A computer algorithm for reconstructing a scene from two projections. Nature, 293: 133-135, Sept 1981.
  - *Read up Richard Dawkins' book "Climbing Mount Improbable" for next week.

End of Motion Analysis

- Relevant textbook sections: Trucco (8.1 - 8.3, 8.4.1, 8.5.2)
  - Sect 8.5.1 is not examinable but it describes a technique typical of most SFM methods used in our field: mathematically involved and fraught with limitations.

- The following few slides introduce you to key arguments researchers are engaged in at the cutting edge of this field. They are not examinable but I hope they will give you a broader perspective and further fascinate you and maybe you will take up research in this field.

Is reconstruction the right approach?

- So far, vision has been conceived as a problem of creating hierarchical representations.
  - 2-D images -> primal sketch -> $2^1/_2$-D sketch -> object-centered descriptions. Known as "from pixels to predicates"
- Vision is described as the process of creating a complete and accurate representation of the scene.
- Thus, much of the motion analysis research has focused on SFM (complete scene recovery), as well as estimating the 3-D motion parameters.

Is reconstruction the right approach?

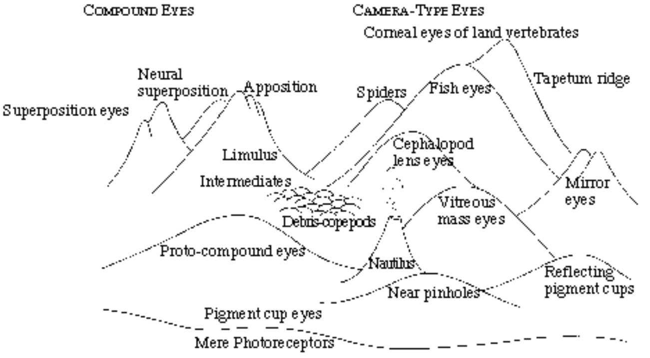
- But complete scene reconstruction results in:
  - more information than is necessary
  - mathematical difficulty, ill-posedness
  - prolonged time needed to solve motion related problems.
- low-level animals, such as anthropods, insects, and mollusks are still able to solve motion analysis problems
  - even they do not possess powerful computational mechanism to perform 3-D scene reconstruction. E.g.
- One fundamental flaw - the study of the visual system is undertaken in isolation from its environment.
  - Given infinite resources, every problem can be solved in principle but resources are finite
  - vision is always purposeful

Is reconstruction the right approach?

- Agent is always engaged in some tasks, subserved by vision
  - Emerging paradigm of purposive vision
- Possible to divide a visual problem into several sub-tasks and solve them without scene reconstruction
- For example, the task of detecting obstacle
  - Not necessary to compute the exact motion
  - But only to recognize certain patterns of flow evolve in a way that signifies collision.
- Instead of reconstructing the world, recognize entities that are directly relevant to task at hand.
  - Does there always exists an appropriate representation to allows us to directly derive the necessary parameters?

Eyes in biological world

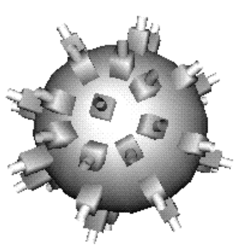
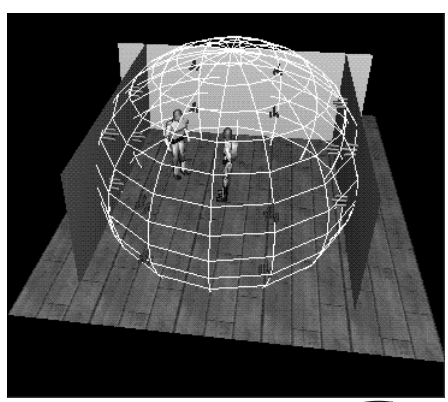
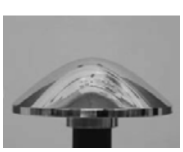
- Must it be camera-type eye?
- Eyes in nature have evolved no fewer than 40 times independently in diverse parts of animal kingdom.
- Eyes "landscape" show 9 basic types of eyes.

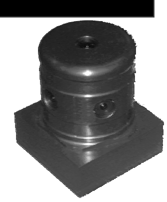


Eyes in biological world

- Why flying animals (insects, birds) have panoramic vision?
  - either as compound eye or having camera-type eyes on opposite sides of head
- Deeper mathematical reasons for having panoramic vision?
  - Resolve the confounding between translation and rotation
  - Insect eyes are not just panaromic! It is built from large collection of ommatidia that can be considered as individual cameras.
    - A large collection of stereo systems?

Non-conventional camera systems

Bio-robotics

- In face of errors in 3-D motion estimates, what motion strategy to adopt?
  - Examples in nature: mantis, locust, <u>wasp</u>