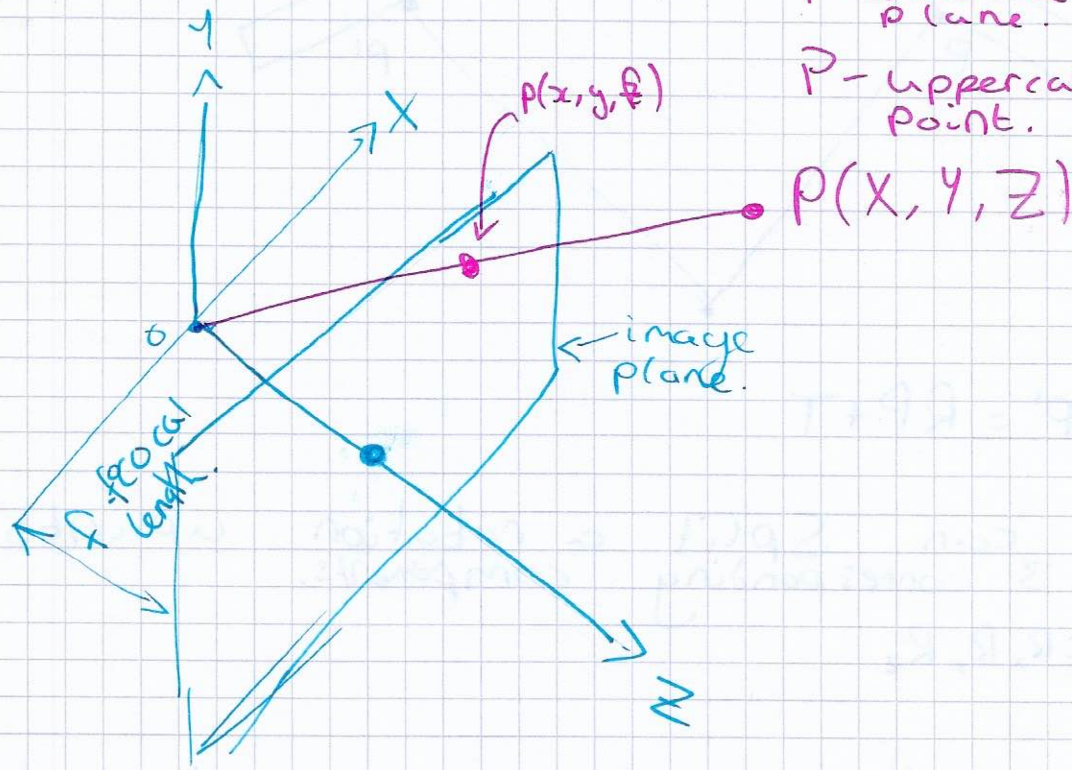
IPCV. - Motion.

p - lowercase in image plane.

P - uppercase - actual point.

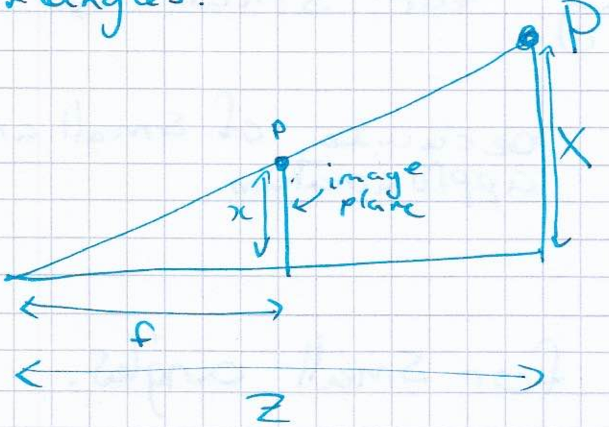


3D point $P = (X, Y, Z)$

is projected onto point in the image plane

$p = (x, y, f)$.

You can relate these points with similar triangles.



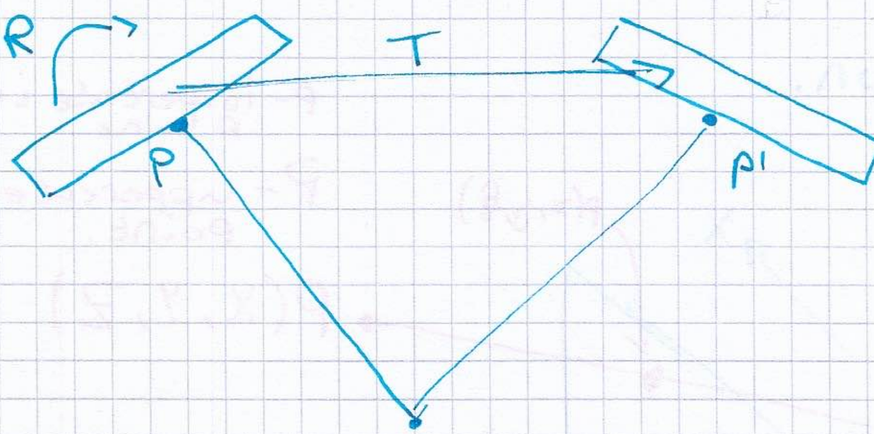
$$x = \frac{fX}{Z}$$

$$\cancel{y = \frac{fZ}{}}$$

$$y = \frac{fY}{Z}.$$

3D-Rigid motion:

- Tracking a moving point with a fixed camera
- or · tracking a fixed point with a moving camera.



$P' = RP + T$

We can split a rotation up into its 3 corresponding components.

$R = R_x R_y R_z$

Applying these to the point individually gives us.

$$R_y P = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x\cos\theta_y + z\sin\theta_y \\ y \\ z\cos\theta_y - x\sin\theta_y \end{bmatrix}$$

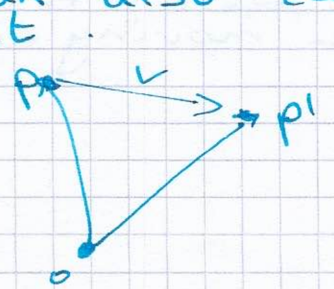
We can also say for small θ_y that,

$$R_y = \begin{bmatrix} 1 & 0 & \theta_y \\ 0 & 1 & 0 \\ -\theta_y & 0 & 1 \end{bmatrix}$$ because of small angle approximation.

This gives us,

$$R = \begin{bmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{bmatrix}$$ for small angles.

We can also calculate the velocity of the point.



$$V = \lim_{\delta t \to 0} \{ P' - P = (R - I)P + T \}$$

So $$V = \begin{bmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}.$$

$$V_x = \theta_y Z - \theta_z Y + T_x$$
$$V_y = \theta_z X - \theta_x Z + T_y$$
$$V_z = \theta_x Y - \theta_y X + T_z$$

$(\theta_x, \theta_y, \theta_z)$: Angular velocity

(T_x, T_y, T_z) = Rectilinear velocity.

To map this to the image space,

$$v_x = \frac{dx}{dt} = \frac{d\left(\frac{fX}{Z}\right)}{dt} \qquad V_x = \frac{dX}{dt},\ V_y = \frac{dY}{dt},\ V_z = \frac{dZ}{dt}$$

using the quotient rule we get

$$v_x = f\,\frac{V_x Z - X V_z}{Z^2}$$

we can now substitute for V_x, V_y, V_z. giving us.

$$v_x = f\,\frac{(\theta_y Z - \theta Y + T_x)Z - X(\theta_x Y - \theta_y X + T_z)}{Z}$$

$$v_x = \frac{(fT_x - xT_z)}{Z} + f\theta_y - \theta_z y - \frac{(\theta_x xy - \theta_y x^2)}{f}$$

$$v_y = \underbrace{\frac{(fT_y - yT_z)}{Z}}_{\text{translational}} \underbrace{- f\theta_x + \theta_z x + \frac{(\theta_y xy - \theta_x y^2)}{f}}_{\text{rotational}}$$

we can split these equations into two components translational and rotational,

we can see that translational is dependent on the depth Z, but the ~~rot~~ rotational is not.

So we can say,

dependent on depth

not dependent on depth.

$$V = \underbrace{tran(x, y, T, Z)}_{\text{translational}} + \underbrace{rot(x, y, \theta)}_{\text{rotational}}.$$

Motion estimation:

This is the estimation of the 2D motion field from frames in an image sequence.
This uses the ~~variation~~ Spatial and temporal variation in pixel values

This does not model real motion, just apparent motion or optical flow, It is just a relationship between variation in pixel values.

We have an image sequence $I(x,y,t)$. and we want to find the variation of pixel values between frames to find the motion.

We can assume $I(x,y,t)$ is constant along the trajectory.

i.e. $\frac{d}{dt}(I(x,y,t)) = 0.$

we know x and y are functions of t. hence we can use the chain rule to give a total derivative.

$$\frac{\partial I}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial I}{\partial y} \cdot \frac{dy}{dt} + \frac{\partial I}{\partial t} = 0.$$

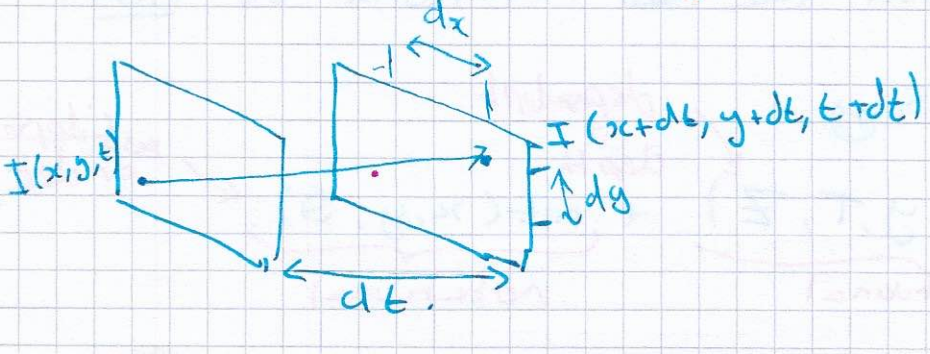
motion (V_x, V_y).

Gradients (I_x, I_y, I_t).

So $I_x V_x + I_y V_y + I_t = 0.$

We could model motion in the following way, The pixel value (x,y,t) will move by dx, dy and dt.

$I(x+dt, y+dt, t+dt) = I(x,y,t).$



we could use linear approximation.

$$I(x+dx, y+dy, t+dt) \approx I(x,y,t) + \underbrace{\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt}_{=0}$$

we can then divide this approximation by dt

$$\frac{\partial I}{\partial x}\cdot\frac{dx}{dt} + \frac{\partial I}{\partial y}\cdot\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0.$$

Again this gives us

Gradients $\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial t}\right) = (I_x, I_y, I_t)$

Motion $\left(\frac{dx}{dt}, \frac{dy}{dt}\right) = (V_x, V_y)$.

with the Optical Flow Equation,

$$\boxed{I_x V_x + I_y V_y + I_t = 0}$$

At one pixel the OFE is underconstrained meaning it can only estimate normal flow. So we need to add extra constraints.

For example, assume a parametric form of motion field in regions, or have a constant velocity.

we could assume it is linear and x and y
eg $V_x = ax + by + c$.

Lets look at the <u>constant velocity model:</u>

- For a region, find the velocity $v = (V_x, U_y)$ which minimises

$$\varepsilon(V_x, V_y) = \sum_{region} (I_x V_x + I_y V_y + I_t)^2$$

To solve this we can take derivatives wrt V_x and V_y. Set them to zero and solve for V_x and V_y.

Lucas & Kanade Algorithm.

Solve for $V = (V_x, V_y)$ given that:

$$V_x \sum I_x^2 + V_y \sum I_x I_y = - \sum I_t I_x$$

$$V_x \sum I_x I_y + V_y \sum I_y^2 = - \sum I_t I_y$$

we can define

$$A = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad b = -\sum \begin{bmatrix} I_t I_x \\ I_t I_y \end{bmatrix}$$

we can calculate V,

$$V = \begin{bmatrix} V_x \\ V_y \end{bmatrix} = A^{-1} b.$$

Full derivation in worksheet 2 Qs

Motion Segmentation.

Motion is a good way of identifying different objects and inferring depth.

We can use motion to segment frames into regions for example,

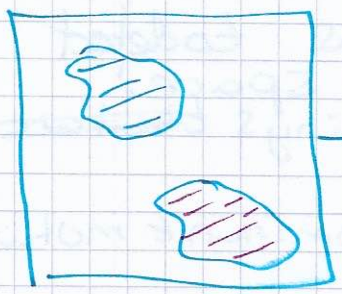
- To extract foreground from background.
- To isolate individual objects.

eg If a person is moving against a white wall we can isolate moving pixels giving us foreground and background.

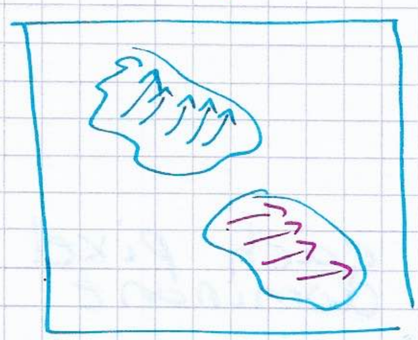
But this is not straight forward!

1. Motion on its own may be abiguous for segmentation, we also want colour, shape, texture
2. What do we consider a homogeneous motion region. single motion, ~~ke~~ parametric variation?
3. Generalised Aperture Problem. Large regions are needed for good motion estimates, but they are more likely to contain complex motions. Bigger the area the more stuff likely going on.

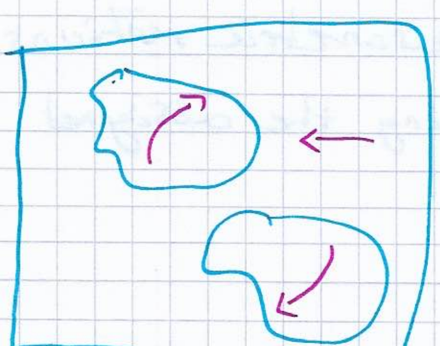
Overview

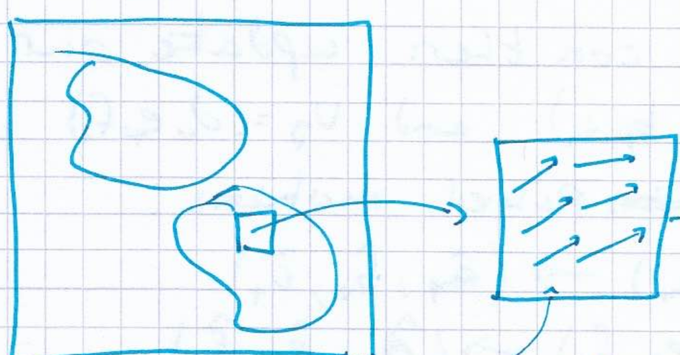


→ Motion Estimation
Eg. L&K.
→ Block motion fitting
↓
Detection of dominant motions
← Motion Segmentation
← Final parametric motion fitting
↓



Block Motion Fitting





→ 3D Motion.

$V_x \approx ax + by + c$

$V_y \approx dx + ey + f$

we can assume linear motion

Detect Dominant Motions

- We can use k-means to detect clusters in the parameter space!
  Ie. group, similar moving things ~~that we~~ together.

  These clusters are the dominant motions.

$$V_x = ax + by + c$$

$$V_y = dx + ey + f$$

run kmeans over (a, b, c)

and (d, e, f).

Each cluster gives us a (a_y, b_y, c_y)

Motion Segmentation

We can then assign each pixel motion to the closest dominant motion.

Parametric Motion Regions

We can then update our parametric motions $V_x = (a_y, b_y, c_y)$ and $V_y = (d_y, e_y, f_y)$ using the assigned ~~motion~~ pixel motions.

$(a_y, b_y, c_y) \rightarrow (\hat{a}_y, \hat{b}_y, \hat{c}_y)$

$(d_y, e_y, f_y) \rightarrow (\hat{d}_y, \hat{e}_y, \hat{f}_y)$.

We now have our motions for each region. the last thing to do is link these regions between the frames.