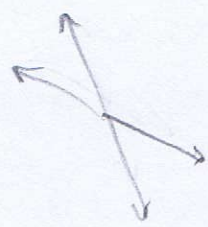
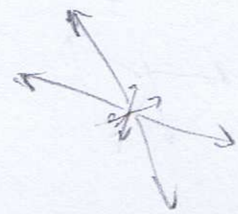
1a) Due to symmetries it is sufficient to consider the upper left region.

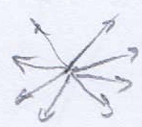
Q →



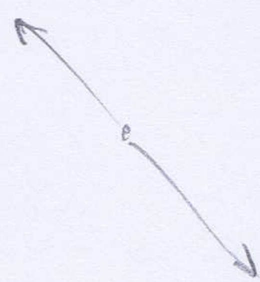
A →



B →



C →



A gives the most similar descriptor. Due to normalization the intensity difference doesn't matter

1b) Given a set of images of known buildings $I_1, \dots I_n$ and a query image $Q$, we extract SIFT features and do Ransac-based registration between $Q$ and every $I_j$. The $I_k$ that yielded most inliers is probably an image of the same building as $Q$.

2a) Let $x_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix}$, $\tilde{x}_i = \begin{pmatrix} \tilde{u}_i \\ \tilde{v}_i \end{pmatrix}$

We seek an affine transf.

$$\begin{pmatrix} \tilde{u}_i \\ \tilde{v}_i \end{pmatrix} \approx \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} u_i \\ v_i \end{pmatrix} + \begin{pmatrix} t_u \\ t_v \end{pmatrix} \quad (*)$$

Six unknowns, 2 equations per correspondence $\Rightarrow$ minimal case: 3 corrs

Rearrange $(*)$ for 3 corrs yields

$$\underbrace{\begin{pmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_2 & v_2 & 1 \\ u_3 & v_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_3 & v_3 & 1 \end{pmatrix}}_{M} \underbrace{\begin{pmatrix} a \\ b \\ t_u \\ c \\ d \\ t_v \end{pmatrix}}_{\theta} = \underbrace{\begin{pmatrix} \tilde{u}_1 \\ \tilde{v}_1 \\ \tilde{u}_2 \\ \tilde{v}_2 \\ \tilde{u}_3 \\ \tilde{v}_3 \end{pmatrix}}_{b}$$

Solve in Matlab: $\theta = M \backslash b$

b) Least squares is sensitive to <u>outliers</u> so we need to remove them first.

c) $I_W(2,3) = I_S(2+0, 3-1) = I_S(2,2) = 11$

3a) $\lambda \begin{pmatrix} \hat{u} \\ \hat{v} \\ 1 \end{pmatrix} = P\mathbb{X} = \begin{pmatrix} 80 \\ 320 \\ 2 \end{pmatrix} \Rightarrow$

$\lambda = 2$ (positive depth).

$\hat{u} = \frac{80}{2} = 40$, $\hat{v} = \frac{320}{2} = 160$

error $= |(40-37, 160-156)| =$

$\sqrt{3^2+4^2} = 5$

---

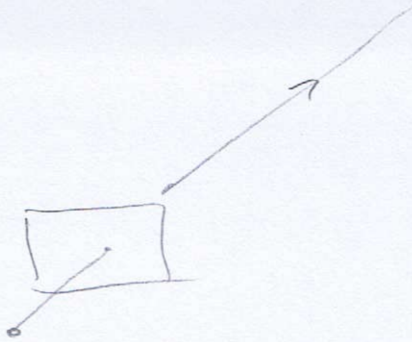
Negative depth $\Leftrightarrow$ behind camera.

b) Let $P = \begin{pmatrix} \leftarrow a^T \rightarrow \\ \leftarrow b^T \rightarrow \\ \leftarrow c^T \rightarrow \end{pmatrix}$

$r_u = \frac{a^T \mathbb{X}}{c^T \mathbb{X}} - u$

$r_v = \frac{b^T \mathbb{X}}{c^T \mathbb{X}} - v$

3c)



With only 1 inlier the distance to the camera is unknown. Useless.

d) Two measurements yields an over-determined system (six equations, five unknowns) so the point is probably correct.

e) Even better.

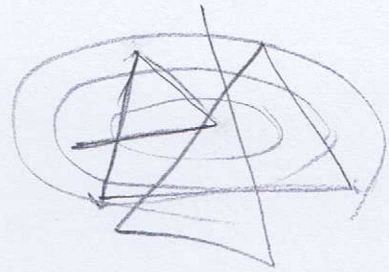
4a) We update the parameters as

$$\theta^{(k+1)} = \theta^{(k)} - \mu \nabla L_i .$$

This is much faster than using the real gradient $\nabla L = \sum_{i=1}^{n} \nabla L_i$.

(Especially for convolutional nets where we need to do backpropagation to compute $\nabla L_i$ )

b) Too large $\mu \Rightarrow$ unstable



Too small $\mu \Rightarrow$ slow.

"Solution": Start with a large $\mu$. Decrease when we stop improving on the validation set.

4c) Training set: Used to learn parameters with stoch. gradient descent.

Validation set: Used to compare different classifiers (or tune hyperparameters etc.)

Test set: Having decided on a classifier we test it once on a test set to assess how good it is.

5a) Probability of picking an outlier-free set : $\alpha = (1-0.875)^3$

Prob. of <u>not</u> picking an outlier-free set $K$ times in a row! $= (1-\alpha)^K$

We want this to be 1% :

$$(1-\alpha)^K = 0.01$$

$$K = \frac{\ln(0.01)}{\ln(1-\alpha)}$$

b) Rule of thumb :

$$K = \frac{100}{P(\text{outlier-free set})}$$

low-resolution : $K = \dfrac{100}{(1/8)^3}$

high resolution $K = \dfrac{100}{(1/4)^3}$

$\Rightarrow$ 1/8 of the iterations.

However, 4 times as many residuals to compute in each iteration

At least for large n : 2 times <u>faster</u>.

6) $R(t) = \exp\left([a]_\times \cdot 5t\right)$

$$= \exp\left(\underbrace{\begin{pmatrix} 0 & 0 & 4 \\ 0 & 0 & -3 \\ -4 & 3 & 0 \end{pmatrix}}_{S} t\right)$$

$$R'(t) = S \cdot \exp(St)$$

$\left[\text{Projection of } \underline{X}(t)\right] : \begin{pmatrix} \hat{u}(t) \\ \hat{v}(t) \end{pmatrix}$

with $P = \begin{pmatrix} \leftarrow a^T \rightarrow 0 \\ \leftarrow b^T \rightarrow 0 \\ \leftarrow c^T \rightarrow 9 \end{pmatrix}$, we get

$$\hat{u} = \frac{a^T \underline{X}(t) + 0}{c^T \underline{X}(t) + 9} = \frac{a^T R(t) \underline{X}_0}{c^T R(t) \underline{X}_0 + 9}$$

Derivative (as in lab 3)

$$\hat{u}'(t) = \frac{a^T R'(t) \underline{X}_0}{c^T R(t) \underline{X}_0 + 9} - \frac{\left(a^T R(t) \underline{X}_0\right)\left(c^T R'(t) \underline{X}_0\right)}{\left(c^T R(t) \underline{X}_0 + 9\right)^2}$$

$$\hat{u}'(0) = \frac{a^T S \underline{X}_0}{(c^T \underline{X}_0 + 9)} - \frac{(a^T \underline{X}_0)(c^T S \underline{X}_0)}{(c^T \underline{X}_0 + 9)^2} = 56$$

In the same way

$$\hat{v}'(0) = \frac{b^T S \underline{X}_0}{c^T \underline{X}_0 + 9} - \frac{(b^T \underline{X}_0)(c^T S \underline{X}_0)}{(c^T \underline{X}_0 + 9)^2} = -4$$