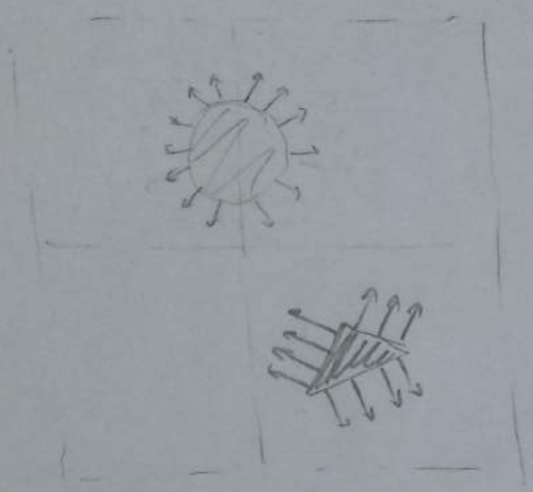
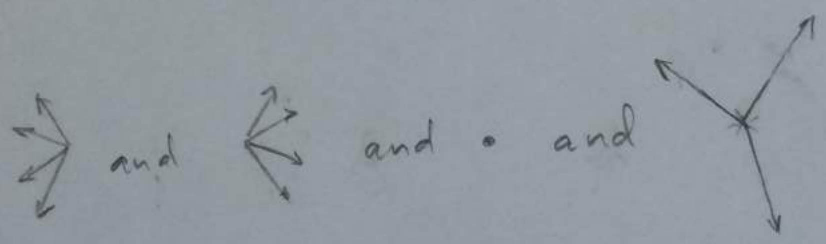
1a) Image gradients will look roughly like this:



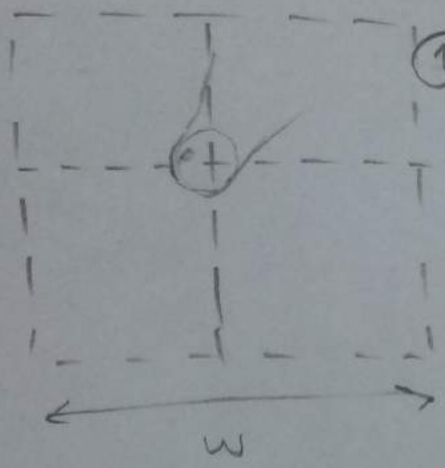
yielding four histograms:



These four 8-bin-histograms are stacked into a 32-vector.

Then the 32-vector is normalized.

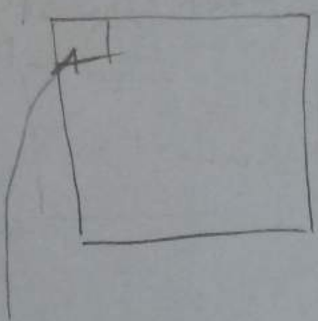
1b) First of all the blob detector also detects the size of the blob, σ.
In SIFT, we place a number of regions around the blob:



① To get scale invariance, the size of this grid is proportional to σ.

② Before computing gradients we filter the image with a Gaussian filter (depending on σ).

2a) Normally, the warped image has the same size as the target image, i.e. 4×4.



$I_w(1,1) = ?$

$$A\begin{pmatrix}1\\1\end{pmatrix} + t = \begin{pmatrix}2\\1\end{pmatrix} + \begin{pmatrix}-1\\1\end{pmatrix} = \begin{pmatrix}1\\2\end{pmatrix}$$

$I_w(1,1) = I_s(1,2) = 6$

$I_w(1,2) = ?$

$$A\begin{pmatrix}1\\2\end{pmatrix} + t = \begin{pmatrix}2\\2\end{pmatrix} + \begin{pmatrix}-1\\1\end{pmatrix} = \begin{pmatrix}1\\3\end{pmatrix}$$

$I_w(1,2) = I_s(1,3) = 29$

$I_w(2,1) = ?$

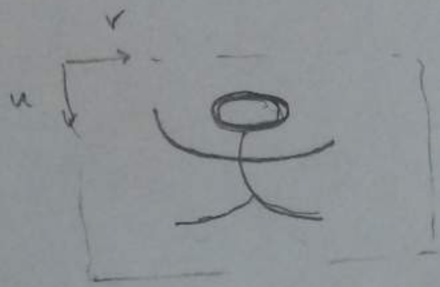
$$A\begin{pmatrix}2\\1\end{pmatrix} + t = \begin{pmatrix}4\\1\end{pmatrix} + \begin{pmatrix}-1\\1\end{pmatrix} = \begin{pmatrix}3\\2\end{pmatrix}$$

$I_w(2,1) = I_s(3,2) = 15$

Now we see the pattern:

$$I_w = \begin{pmatrix} 6 & 29 & 11 & 10 \\ 15 & 9 & 4 & 24 \\ 28 & 21 & 27 & 20 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

2b) The stick figure will be compressed to half its height.



2c) A similarity transformation, i.e. scaling, rotation and translation. An affine transformation requires three points in Ransac ⇒ Much more iterations. ⇒ longer time.

3a) We have 3 unknowns in $X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}$.

Camera equation:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = P \begin{pmatrix} X \\ 1 \end{pmatrix} = \begin{pmatrix} \leftarrow a \rightarrow \\ \leftarrow b \rightarrow \\ \leftarrow c \rightarrow \end{pmatrix} \begin{pmatrix} X \\ 1 \end{pmatrix} \quad (*)$$

Each camera/view $\Rightarrow$ 3 equations and 1 extra unknown (λ) so we need two cameras. We use $\sim$ for variables in the second camera. Rewrite $(*)$ as

$$\lambda u = a_1 X_1 + a_2 X_2 + a_3 X_3 + a_4$$
$$\lambda v = b_1 X_1 + b_2 X_2 + b_3 X_3 + b_4$$
$$\lambda = c_1 X_1 + c_2 X_2 + c_3 X_3 + c_4$$

$\Leftrightarrow$

$$\begin{pmatrix} a_1 & a_2 & a_3 & -u \\ b_1 & b_2 & b_3 & -v \\ c_1 & c_2 & c_3 & -1 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ \lambda \end{pmatrix} = \begin{pmatrix} -a_4 \\ -b_4 \\ -c_4 \end{pmatrix}$$

(not solvable since the matrix is 3×4)

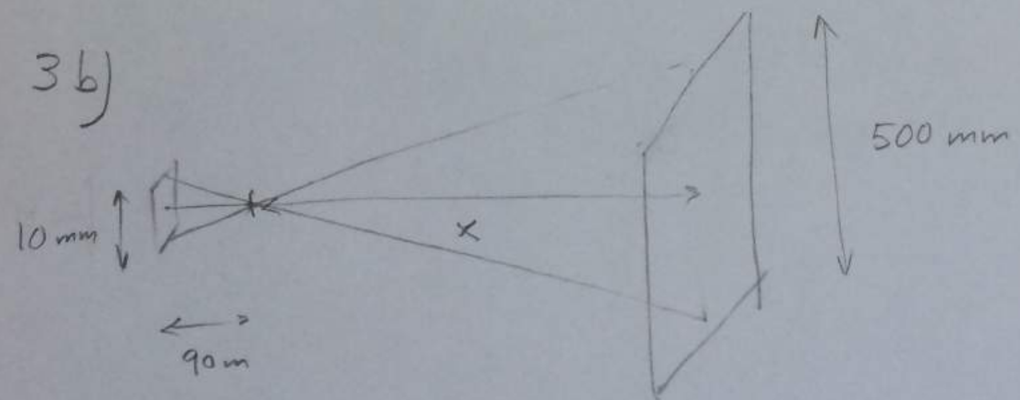
Adding the equations for the other camera yields:

3a) (cont)

$$\underbrace{\begin{pmatrix} a_1 & a_2 & a_3 & -u & 0 \\ b_1 & b_2 & b_3 & -v & 0 \\ c_1 & c_2 & c_3 & -1 & 0 \\ \tilde{a}_1 & \tilde{a}_2 & \tilde{a}_3 & 0 & -\tilde{u} \\ \cancel{\tilde{b}_1} & \cancel{\tilde{b}_2} & \cancel{\tilde{b}_3} & \cancel{0} & \cancel{-\tilde{v}} \\ \tilde{c}_1 & \tilde{c}_2 & \tilde{c}_3 & 0 & -1 \end{pmatrix}}_{M} \underbrace{\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ \lambda \\ \tilde{\lambda} \end{pmatrix}}_{\theta} = \underbrace{\begin{pmatrix} -a_4 \\ -b_4 \\ -c_4 \\ -\tilde{a}_4 \\ \cancel{-\tilde{b}_4} \\ -\tilde{c}_4 \end{pmatrix}}_{b}$$

To avoid an overdetermined system we throw away one equation and solve in Matlab using $\theta = M \backslash b$.
(works without throwing as well.)

3b)



$$\frac{90}{10} = \frac{x}{500} = 4500$$

At 4.5 m from the camera centre.

4a) The negative log-likelihood is

$$\sum_{\substack{\text{positive}\\\text{examples}}} -\log(p_i) + \sum_{\substack{\text{negative}\\\text{examples}}} -\log(1-p_i)$$

$$= -\log(0.9) - \log(0.9) - \log(0.8) - \log(0.8)$$
$$- \log(1-0.1) - \log(1-0.1) - \log(1-0.3) - \log(1-0.2)$$

$$= -4\log(0.9) - 3\log(0.8) - \log(0.7)$$

4b) More filters always tend to give a lower loss, but this may be due to overfitting. Instead we use a separate validation set to choose the best classifier.

4c) In data augmentation we generate "new" training data from existing data, e.g. by rotating or scaling existing examples. For cells every possible rotation yields a new example, We could also use small translations and scalings. For digits we should not use large rotations.

5a) Using the chain rule:

$$\frac{\partial L_i}{\partial w_1} = \sum_{k=1}^{3} \frac{\partial L_i}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial w_1}$$

$$\frac{\partial \hat{y}_k}{\partial w_1} = \hat{w}_k \cdot \frac{\partial \hat{x}}{\partial w_1} = \hat{w}_k \cdot \frac{d\hat{x}}{dz} \cdot \frac{\partial z}{\partial y_1} x_1$$

and here $\frac{\partial z}{\partial y_1} = 1$ and

$$\frac{d\hat{x}}{dz} = \zeta'(z) = \frac{e^z}{1+e^z} - \frac{(e^z)^2}{(1+e^z)^2} =$$

$$= \frac{(1+e^z)e^z - (e^z)^2}{(1+e^z)^2} = \frac{e^z}{(1+e^z)^2} = \hat{x}(1-\hat{x})$$

Formula:

$$\frac{\partial L_i}{\partial w_1} = \sum_{k=1}^{3} \left(\frac{\partial L_i}{\partial \hat{y}_k} \cdot \hat{w}_k \, \hat{x}(1-\hat{x}) \, x_1 \right)$$

$$= \hat{x}(1-\hat{x}) x_1 \sum_{k=1}^{3} \left(\hat{w}_k \frac{\partial L_i}{\partial \hat{y}_k} \right)$$

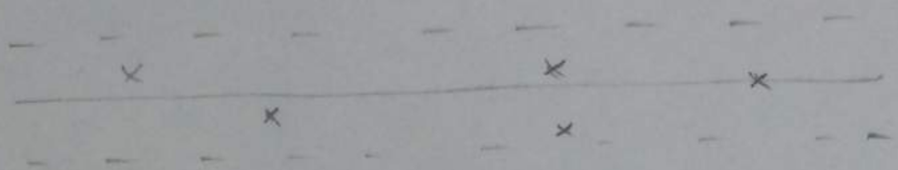
5b) Inserting numerical values

$$\frac{\partial L_i}{\partial w_2} = 0.5(1-0.5)\cdot 2\left(2\cdot 1 + 4\cdot 1 + 1\cdot 2\right)$$

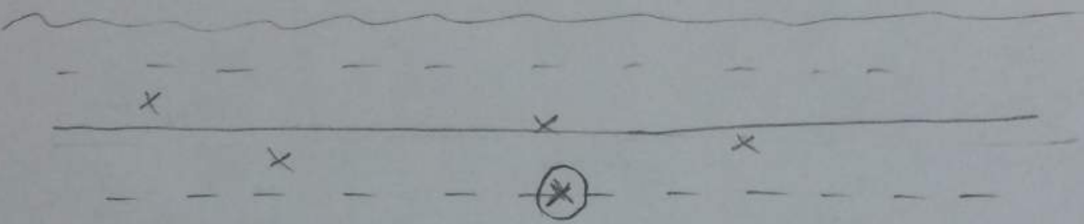
$$= \frac{1}{2}(8) = 4.$$

5c) $w_1 = 1 - 0.01 \cdot 4 = 0.96$

6 a) Consider an optimal line.



If no point has $|r_i(\theta)| = \tau$, we can change increase b without changing the number of outliers. Increase b until at least one $|r_i(\theta)| = \tau$.



If only one point has $|r_i(\theta)| = \tau$, we can rotate the line about this point until another $|r_i(\theta)| = \tau$. This does not change the number of outliers.

$\Rightarrow$ We have found an optimal line with at least two $|r_i(\theta)| = \tau$.

6b) Go through every pair of points i and j. Find the lines such that $|r_i(\theta)| = \tau$ and $|r_j(\theta)| = \tau$ and compute the loss for each such line.

The line with the lowest loss is optimal.