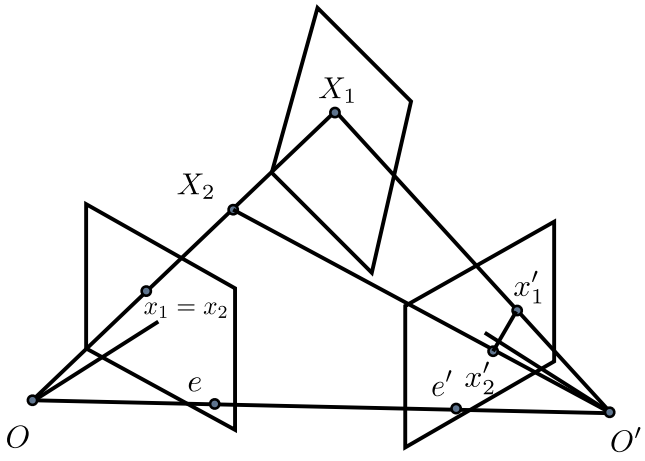
C4B Computer Vision
Question Sheet 2 answers

Andrea Vedaldi
vedaldi@robots.ox.ac.uk

Question 1- Structure from Motion

- Consider two cameras O_1, O_2 looking at a plane
  - Let X_1 be a point *on the plane* and x_1 , x_1' its projections on the cameras O, O'
  - (The homography H is the matrix such that $x_1' \propto H\, x_1$)
- Let X_2 and a 3-D point that overlaps with X_1 as seen fro from O (i.e. $x_1 = x_2$)
  - X_2 is in on the triangle of vertices X_1, O, O'. This triangle:
    - contains the epipolar line passing through x_1';
    - contains both x_1' and x_2', hence the segment x_2' - x_1'.
  - Hence the epipolar line contains the segment x_2' - x_1' .



Question 1 - A note on homography

- An homography H is a matrix that relates the projections of points belonging to a given 3-D plane on two different cameras
  - Let $x_1 = P X_1$ and $x_1' = P' X_1$
  - Assume (without loss of generality) that the world reference frame is such that the plane has equation z = 0, i.e.

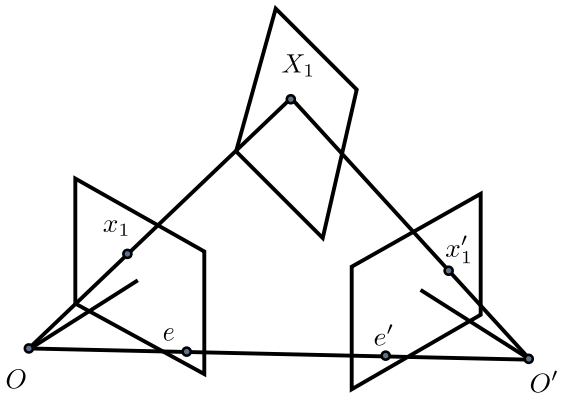
$$X_1 = \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

  - Then multiplying P by X_1 effectively removes one column from P, yielding an invertible matrix Q:

$$x_1 = PX_1 = Q \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad x_1' = P'X_1 = Q' \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  - Hence we can write

$$x_1' = Q'^{-1} Q x_1 = H x_1$$



Question 2 - RANSAC

- 3/4 = probability that an element is an inlier

- we select at random a subset of *n* elements (call this ***trial***)
  - P0 = probability that a trial contains only inliers = (3/4)^n

- we compute the probability of hitting a trial of only inliers after trying *m* times
  - Pm = probability that at least one trial out of *m* has only inliers
  - Pm = 1 - probability that all *m* trials do not have only inliers
  - Pm = 1 - (1 - P0)^m

- Hence we can compute how large m should be so that Pm >= .99
  - *m* >= log(1 - .99) / log(1 - P0)

- Examples:
  - for *n* = 8 elements in a trial : P0 = $(3/4)^8$, *m* >= 44
  - for *n* = 7 elements in a trial : P0 = $(3/5)^7$, *m* >= 33

Question 3 - Feature points

- Features are located by searching for local extrema (in location and scale) of the Difference of Gaussian (DoG) scale space. The DoG function approximates the response of a scale-normalised Laplacian operator, which can be thought as a match filtrer for blob-like structure.

- Local maxima are invariant to a rescaling of the DoG function, achieving illumination invariance. Invariance (in fact covariance) to translation and scale is obtained by associating feature points to structures (blobs) that are searched at all locations and scales. Covariance to rotation is obtained by associating the feature point rotation to the dominant image orientation in the feature point region.

- The second derivative of the Gaussian kernel is proportional to its derivative with respect to scale. For instance, in the 1-D case:

$$
\begin{aligned}
g_\sigma(x) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{x^2}{\sigma^2}} \\
\frac{\partial g}{\partial x}(x) &= -g_\sigma(x)\frac{x}{\sigma^2} \\
\frac{\partial^2 g}{\partial x^2}(x) &= g_\sigma(x)\frac{x^2-\sigma^2}{\sigma^4} \\
\frac{\partial g}{\partial \sigma}(x) &= g_\sigma(x)\frac{x^2-\sigma^2}{\sigma^3} = \sigma\frac{\partial^2 g}{\partial x^2}(x)
\end{aligned}
$$

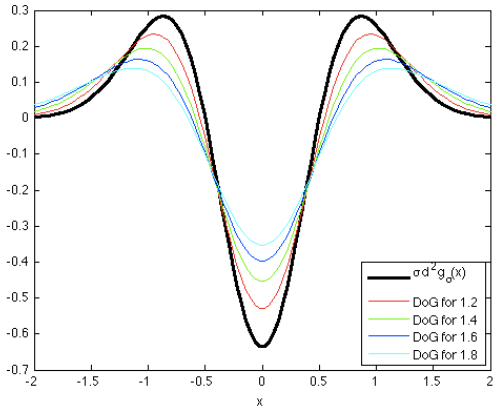
Question 3 - Plots

```
function question4
  x = linspace(-2,2,100) ;
  sigma = 0.5 ;

  figure(1) ; clf ;
  plot(x, ddg(x, sigma) * sigma, 'k','linewidth',2) ;
  hold on ;
  leg = {'\sigma d^2g_\sigma(x)'} ;
  cls = {'r','g','b','c'} ;
  for factor = [1.2, 1.4, 1.6, 1.8]
    sigmap = sigma * factor ;
    dsigma = sigmap - sigma ;
    plot(x, (g(x,sigmap) - g(x,sigma)) / dsigma, cls{1}) ;
    cls(1) = [] ;
    leg{end+1} = sprintf('DoG for %.1f', factor) ;
  end
  xlabel('x') ;
  legend(leg{:},'location','southeast') ;
end

function y = g(x, sigma)
  y = 1/(2*pi*sigma) * exp(.5 * -x.^2/sigma^2) ;
end

function y = ddg(x, sigma)
  y = g(x, sigma) .* (x.^2 - sigma^2) / sigma^4 ;
end
```



Question 4 - Projected visual motion

- Consider a moving 3-D point $X(t) = R(t)X_0 + T(t)$
  - its velocity writes $\dot{X}(t) = \Omega X_0 + V$
  - **Rem.** Here for convenience we use a matrix notation for the vector product:

$$[\omega]_\times = \Omega = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \quad [\omega]_\times a = \omega \times a.$$

  The hat here is the standard notation for *hat* operator, and it does not denote a unit vector. Using this is not necessary, but definitely convenient.
- The projection *x*(*t*) of *X*(*t*) and its derivative writes

$$x(t) = f\frac{X(t)}{\Pi_3 X(t)} \qquad \begin{aligned} \dot{x}(t) &= f\frac{(\Omega X_0 + V)(\Pi_3 X(t)) - X(t)(\Pi_3(\Omega X_0 + V))}{(\Pi_3 X(t))^2} \\ &= f\frac{\Omega X_0 + V}{\Pi_3 X(t)} - x(t)\frac{\Pi_3(\Omega X_0 + V)}{\Pi_3 X(t)} \end{aligned}$$

where $\Pi_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$ extracts the third component (depth) form a vector.
- Assuming R(0) = I and T(0) = 0, the derivative (velocity) at *t* = 0 is

$$\dot{x}(0) = (fI - x(0)\Pi_3)\left(\frac{V}{\Pi_3 X_0} + \Omega x(0)\right)$$

velocity / depth ambiguity

Question 4 - Focus of expansion

- We simplify the previous formula for the case $R(0) = I, \quad T(0) = 0, \quad \Omega = 0$ and keeping only the first two components of the velocity (the third is zero anyways...)

$$v = \Pi_{12}\dot{x}(0) = \frac{1}{\Pi_3 X_0}(fV_{12} - xV_3) \qquad \Pi_{12} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

  - *v* is the vector of the first two components of $\dot{x}(0)$
  - x denotes the first two components of x(0) (i.e., the pixel coordinates)
- We want to compute the focus of expansion given two pixels x, x' and their velocities v,v'
  - The focus of expansion is a third pixel x'' that has velocity v'' = 0
  - We write down and solve a system of linear equations capturing such information

$$\begin{aligned} \alpha v &= fV_{12} - xV_3 \\ \beta v' &= fV_{12} - x'V_3 \\ 0 &= fV_{12} - x''V_3 \end{aligned} \qquad \begin{aligned} \alpha v &= (x'' - x)V_3 \\ \beta v' &= (x'' - x')V_3 \end{aligned} \qquad \begin{aligned} \bar{\alpha} v + x &= x'' \\ \bar{\beta} v' + x' &= x'' \end{aligned}$$

- Hence x'' is the intersection of two lines along the velocities v, v'.
  - Intuitively, the intersection is unstable when the two lines are quasi-parallel.
  - Formally, this can be seen by taking the determinant of the linear system:

$$\begin{bmatrix} I & -v & 0 \\ I & 0 & -v' \end{bmatrix} \begin{bmatrix} x'' \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} x \\ x' \end{bmatrix} \qquad \det \begin{bmatrix} I & -v & 0 \\ I & 0 & -v' \end{bmatrix} = v \wedge v'$$

Question 4 - Rotation

- We use the previous formula for the case $R(0) = I, \quad T(0) = 0, \quad \Omega \neq 0$
  - **Rem.** The angular velocity is not zero.

$$\dot{x}(0) = (fI - x(0)\Pi_3)\left(\frac{V}{\Pi_3 X_0} + \Omega x(0)\right)$$

- We keep only the first two components (the third is equal to zero anyways ...)
  - *v* is the measured velocity of the pixel *x*
  - we compensate the velocity for the angular component (which we know)

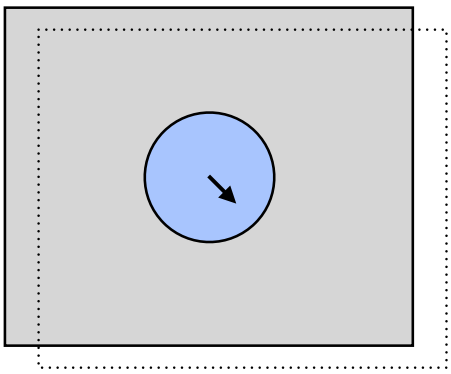
$$\tilde{v} = v - (f\Pi_{12} - x\Pi_3)\Omega x(0) = f\frac{V_{12}}{\Pi_3 X_0} - x\frac{V_3}{\Pi_3 X_0}$$

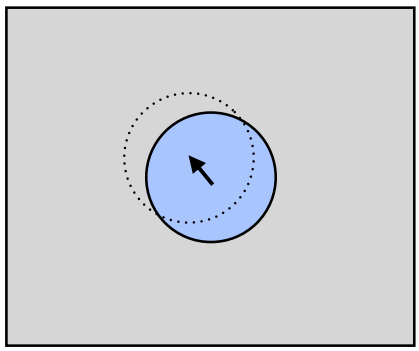
- notice that, predictably, V_{12} and V_3 are divided by the depth
  - the scale cannot be determined (depth/velocity ambiguity)
- given two pixels x, x' and the corresponding compensated velocities, form the system

$$\begin{bmatrix}\tilde{v}\\ \tilde{v}'\end{bmatrix} = \begin{bmatrix}fI & -x\\ fI & -x'\end{bmatrix}\left(\frac{1}{\Pi_3 X_0}\begin{bmatrix}V_{12}\\ V_3\end{bmatrix}\right)$$

- this is a 4x3 (overcomplete) linear system that can be solved for the scaled velocity

Question 5 - LK tracker





- Standard LK tracker
  - try to find a small translation of the image so that the image window matches the template better

$$\min_{\delta t_x, \delta t_y} \sum_{xy} \left(I(x + t_x + \delta t_x, y + t_y + \delta t_y) - T(x, y)\right)^2$$

- Inverse KL tracker
  - try to find a small translation of the template so that it matches better the image window

$$\min_{\delta t_x, \delta t_y} \sum_{xy} \left(I(x + t_x, y + t_y) - T(x - \delta t_x, y - \delta t_y)\right)^2$$

Question 5 - LK tracker

- The goal is to solve the minimisation problem

$$\min_{\delta t_x, \delta t_y} \sum_{xy} \left(I(x+t_x, y+t_y) - T(x-\delta t_x, y-\delta t_y)\right)^2$$

- Expanding the term inside the square to the first order:

$$\begin{aligned} &\approx \sum_{xy} \left(I(x+t_x, y+t_y) - T(x,y) - \nabla T(x,y) \begin{bmatrix}\delta t_x \\ \delta t_y\end{bmatrix}\right)^2 \\ &= \sum_{xy} (I(x+t_x, y+t_y) - T(x,y))^2 - 2\left(\sum_{xy} (I(x+t_x, y+t_y) - T(x,y)) \nabla T(x,y)\right) \begin{bmatrix}\delta t_x \\ \delta t_y\end{bmatrix} \\ &\quad + \begin{bmatrix}\delta t_x & \delta t_y\end{bmatrix} \left(\sum_{xy} \nabla T(x,y)^\top \nabla T(x,y)\right) \begin{bmatrix}\delta t_x \\ \delta t_y\end{bmatrix} \end{aligned}$$

- This is a quadratic function in dt_x, dt_y. We minimise it by setting the partial derivatives to zero, obtaining

$$\begin{bmatrix}\delta t_x \\ \delta t_y\end{bmatrix} = \left(\sum_{xy} \nabla T(x,y)^\top \nabla T(x,y)\right)^{-1} \left(\sum_{xy} (I(x+t_x, y+t_y) - T(x,y)) \nabla T(x,y)^\top\right)$$

- Compare with the standard LK tracker

$$\begin{bmatrix}\delta t_x \\ \delta t_y\end{bmatrix} = \left(\sum_{xy} \nabla I(x+t_x, y+t_y)^\top \nabla I(x+t_x, y+t_y)\right)^{-1} \left(\sum_{xy} (I(x+t_x, y+t_y) - T(x,y)) \nabla I(x,y)^\top\right)$$

  - advantage 1: the matrix to be inverted depends only on T and can be compute once
  - advantage 2: we compute the gradient of T once rather than the gradient of I

Question 6 - Rapid Tracker

- Obtaining the RAPID tracker equation is analogous to the derivation of the projected motion from Question 4. In fact, they are the same up to a renaming of the symbols.
  - recall from Question 4

$$X(t) = R(t)X_0 + T(t) \qquad \dot{X}(t) = \Omega X_0 + V \qquad [\omega]_\times = \Omega = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \quad [\omega]_\times a = \omega \times a.$$

$$x(t) = f\frac{\Pi_{12} X(t)}{\Pi_3 X(t)} \qquad \dot{x}(0) = f\frac{\Pi_{12}\Omega X_0 + V_{12}}{\Pi_3 X(0)} - x(0)\frac{\Pi_3 \Omega X_0 + V_3}{\Pi_3 X(0)}$$

  - thus given X(0) and x(0) we can approximate X(dt) and x(dt)

$$X(\delta t) = X(0) + \delta t V + \delta t \Omega X_0 \qquad x(\delta t) = x(0) + \dot{x}(0)\delta t$$

  - This is the same as the RAPID tracker, provided that X(0) = R(0) X_0 + T(0) = **T** + **P**
    To this end:
    - set f = 1, X_0 = **P**, R(0) = I, and T(0) = **T**
    - rename V as v,
      x(0) = **x**, X(0) = **X**,
      x(dt) = **x**', and X(dt) = **X**'

$$\mathbf{X}' = \mathbf{T} + \delta t \mathbf{v} + \mathbf{P} + \delta t \boldsymbol{\omega} \times \mathbf{P}$$
$$X(\delta t) = X(0) + \delta t V + \delta t \Omega X_0$$

- With this equivalence in mind, we obtain the answer by expanding the formula for the derivative of x(t)

$$x(\delta t) \approx \begin{bmatrix}x_1 \\ x_2\end{bmatrix} + \frac{1}{X_{03} + T_3} \begin{bmatrix} V_1 + (\omega_2 X_{03} - \omega_3 X_{02}) - x_1(\omega_1 X_2 - \omega_2 X_1) \\ V_2 + (\omega_3 X_{01} - \omega_1 X_{01}) - x_2(\omega_1 X_2 - \omega_2 X_1) \end{bmatrix} \delta t$$

Question 6 - continued

- We get the final result by doing the substitutions

$$x(\delta t) \approx \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \frac{1}{X_{03} + T_3} \begin{bmatrix} V_1 + (\omega_2 X_{03} - \omega_3 X_{02}) - x_1(\omega_1 X_2 - \omega_2 X_1) \\ V_2 + (\omega_3 X_{01} - \omega_1 X_{01}) - x_2(\omega_1 X_2 - \omega_2 X_1) \end{bmatrix} \delta t$$

$$\mathbf{x}' \approx \begin{bmatrix} x + \delta t\,(v_x + \omega_y P_z - \omega_z P_y - x(v_z + \omega_x P_y - \omega_y P_x))\,/\,(T_z + P_z) \\ y + \delta t\,(v_y + \omega_z P_x - \omega_x P_z - y(v_z + \omega_x P_y - \omega_y P_x))\,/\,(T_z + P_z) \end{bmatrix}$$

- The small motion assumption is required so that control points are associated to the correct edges.

Question 7 - AdaBoost

- Remarks:
  - Weak classifiers are specified as their values on the training data. This is sufficient for training, but an explicit functional form is required for testing.
- Sample output:

```
errors =

   0.3333   0.3333   0.2222   0.3333

errors =

   0.2143   0.2143   0.5000   0.2143

errors =

   0.5000   0.1364   0.3182   0.2576

x =

    1    1    1    1   -1   -1   -1   -1   -1

H =

    1    1    1    1   -1   -1   -1   -1   -1
```

```
function boost

% training data
x = [1 1 1 1 -1 -1 -1 -1 -1] ;

% weak classifiers response on the training data
h = zeros(4,9) ;
h(1,:) = [1 1 -1 1 -1 -1 -1 1 1] ;
h(2,:) = [1 -1 1 1 -1 1 1 -1 -1] ;
h(3,:) = [1 1 1 -1 1 -1 -1 -1 -1] ;
h(4,:) = [-1 1 -1 1 -1 -1 1 -1 -1] ;

w = ones(1,9) ;

T = 3 ;
alpha = zeros(1,T) ;
weakClassifiers = zeros(1,T) ;
errors = zeros(1, size(h,1)) ;

for t=1:T
  w = w / sum(w) ;
  for j=1:size(h,1)
    errors(j) = sum(w .* (1 - h(j,:) .* x) / 2) ;
  end
  [bestError, bestWC] = min(errors) ;
  weakClassifiers(t) = bestWC ;
  alpha(t) = 0.5 * log((1 - bestError) / bestError) ;
  w = w .* exp(- alpha(t) * x .* h(bestWC, :)) ;
  errors
end

x
H = sign(alpha * h(weakClassifiers, :))
```

Question 7 - Monotonicity of the weights

- We plot alpha as a function of epsilon

$$\alpha(\epsilon) = \frac{1}{2} \log \frac{1-\epsilon}{\epsilon}$$