CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

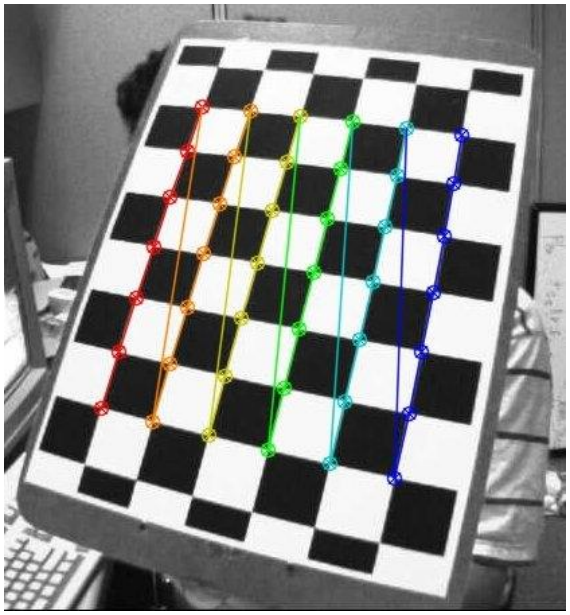
Machine Vision

Lecture 11: Auto-calibration, SLAM & wrap-up

Auto-calibration

- In the previous lecture we have seen how metric scene calibration is a 2-step process, with camera calibration always the first step
- This approach requires control of the environment, i.e. we need to be able to take pictures of a calibration object ourselves

- Question: Can we obtain a metric scene reconstruction without this pre-calibration step?



Auto-calibration

- Without calibration matrix $\boldsymbol{K}$ the best we can do from pure point correspondences is determine a projective reconstruction $\{\boldsymbol{P_i}, \boldsymbol{X_j}\}$, which is only determined up to a common 3d homography $\boldsymbol{H}$
- Typically we fix this homography by setting $\boldsymbol{P_1} = [\boldsymbol{I} \quad \boldsymbol{0}]$
- What this means is that all reconstructions $\{\boldsymbol{P_i H}, \boldsymbol{H^{-1} X_j}\}$ are perfectly feasible solutions as well
- **Auto-calibration** therefore seeks to determine a homography $\boldsymbol{H}$ so that the result is metric, i.e. only determined up to a 3d similarity (scale, translation, rotation)

Auto-calibration

- In the calibrated case we would set the first camera to $\boldsymbol{P_1' = K[I \quad 0]}$
- Therefore, the homography we seek is $\boldsymbol{P_1 H = P_1'}$
- Hence

$$\boldsymbol{H = \begin{bmatrix} K & 0 \\ v^T & 1 \end{bmatrix}}$$

- If we apply this homography to the plane at infinity we get the **plane at infinity** in the projective reconstruction

$$\boldsymbol{\pi_\infty = H^{-T} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{bmatrix} -K^{-T} v \\ 1 \end{bmatrix}}$$

- Therefore a projective reconstruction can be transformed into a metric reconstruction if we know the calibration matrix of the first camera $\boldsymbol{K}$ and the plane at infinity $\boldsymbol{\pi_\infty = [p^T \quad 1]^T}$

Auto-calibration

- The other projective cameras $\boldsymbol{P_i = [A_i \quad a_i]}$ also transform into a metric coordinate frame $\boldsymbol{P'_i = K_i R_i^T [I \quad -t_i]}$ according to $\boldsymbol{P_i H = P'_i}$
- Therefore

$$\boldsymbol{[A_i \quad a_i] \begin{bmatrix} K & 0 \\ -p^T K & 1 \end{bmatrix} = K_i R_i^T [I \quad -t_i]}$$

- Or

$$\boldsymbol{K_i R_i^T = (A_i - a_i p^T) K}$$

- Because $\boldsymbol{R_i^T R_i = I}$ we can square this equation to eliminate the rotation and get the **auto-calibration equations**

$$\boldsymbol{K_i K_i^T = (A_i - a_i p^T) K K^T (A_i - a_i p^T)}$$

Auto-calibration

- The product of calibration matrices $\boldsymbol{\omega}^* = \boldsymbol{K}\boldsymbol{K}^{\boldsymbol{T}}$ is called the dual image of the absolute conic (DIAC)

- If we assume that all the calibration matrices are identical, i.e. $\boldsymbol{K_i} = \boldsymbol{K}$, the auto-calibration equations (at least in theory) provide constraints on the plane at infinity $\boldsymbol{p}$ and the common DIAC

$$\boldsymbol{\omega}^* = \left(\boldsymbol{A_i} - \boldsymbol{a_i}\boldsymbol{p}^{\boldsymbol{T}}\right)\boldsymbol{\omega}^*\left(\boldsymbol{A_i} - \boldsymbol{a_i}\boldsymbol{p}^{\boldsymbol{T}}\right)$$

Constraints on the DIAC

- Written in terms of the parameters of the calibration matrix the DIAC is

$$\boldsymbol{\omega}^* = \begin{pmatrix} c & s & x_0 \\ & \alpha c & y_0 \\ & & 1 \end{pmatrix} \begin{pmatrix} c & s & x_0 \\ & \alpha c & y_0 \\ & & 1 \end{pmatrix}^T = \begin{pmatrix} c^2 + s^2 + x_0^2 & s\alpha c + x_0 y_0 & x_0 \\ \cdot & \alpha^2 c^2 + y_0^2 & y_0 \\ \cdot & \cdot & 1 \end{pmatrix}$$

- Further constraints can be imposed, for instance
  - Digital cameras have zero skew, i.e. $s = 0$, which means that
$$\omega_{12}^* \omega_{33}^* = \omega_{13}^* \omega_{23}^*$$
  - The principal point is typically in the middle of the image, which we can define as the origin, i.e. $x_0 = y_0 = 0$. This means
$$\omega_{13}^* = \omega_{23}^* = 0$$

The modulus constraint

- We already saw that

$$\mu \boldsymbol{K R_i^T K^{-1}} = \boldsymbol{A_i} - \boldsymbol{a_i p^T}$$

- Because $\boldsymbol{K R_i^T K^{-1}}$ is a conjugate rotation, it's eigenvalues are $\{1, e^{i\theta}, e^{-i\theta}\}$ for some θ
- This means, that the eigenvalues of $\boldsymbol{A_i} - \boldsymbol{a_i p^T}$ are $\{\mu, \mu e^{i\theta}, \mu e^{-i\theta}\}$
- These eigenvalues are the roots of the characteristic polynomial

$$\det(\lambda \boldsymbol{I} - \boldsymbol{A_i} + \boldsymbol{a p^T}) = (\lambda - \mu)\left(\lambda - \mu e^{i\theta}\right)\left(\lambda - \mu e^{-i\theta}\right)$$

- From which we can eliminate μ and θ to derive constraints on the plane at infinity $\boldsymbol{p}$

Summary

- None of these constraints work particularly well in the presence of noise
- In particular, finding the plane at infinity $\boldsymbol{\pi}_\infty$ from uncalibrated images turns out to be extremely difficult
- Therefore it is advisable to use calibrated cameras whenever possible in most machine vision applications

Simultaneous localisation & mapping

- In the previous lecture we also looked at the computation of camera locations and scene points from sequences of images
- Depending on the application this problem is called **simultaneous localisation and mapping** (SLAM)
- The parameters are camera locations and scene points, while the observations are image coordinates
- We saw that the optimal solution is a bundle adjustment minimising the reprojection error
- When processing live image sequences calculating a full bundle adjustment over the full past sequence is often not practical
- Instead we need an approach where we can add images one by one and estimate the new location and additional scene points incrementally

Kalman filter

- Let's assume the model function for the bundle adjustment is a reprojection error like the following

$$g_0[\boldsymbol{p}, \boldsymbol{l_0}] = f_0[\boldsymbol{p}] - \boldsymbol{l_0} = \boldsymbol{0}$$

- Then (with the Jacobian $\boldsymbol{A_0} = \frac{\partial f_0}{\partial \boldsymbol{p}}$) the Normal equation system for this problem is

$$\left(\boldsymbol{A_0^T C_{l_0 l_0}^{-1}}\, \boldsymbol{A_0}\right)\boldsymbol{\Delta p} = \boldsymbol{A_0^T C_{l_0 l_0}^{-1} \Delta l_0}$$

Kalman filter

- The solution is obtained by inverting the Normal equation matrix as

$$\boldsymbol{\Delta p} = \left(\boldsymbol{A_0^T C_{l_0 l_0}^{-1} A_0}\right)^{-1} \boldsymbol{A_0^T C_{l_0 l_0}^{-1} \Delta l_0}$$

- The covariance matrix of this estimate is the inverse of the Normal equation matrix

$$\boldsymbol{C_{pp}} = \left(\boldsymbol{A_0^T C_{l_0 l_0}^{-1} A_0}\right)^{-1}$$

- So we can also write the parameter update in terms of the covariance estimate as

$$\boldsymbol{\Delta p} = \boldsymbol{C_{pp} A_0^T C_{l_0 l_0}^{-1} \Delta l_0}$$

Kalman filter

- So what happens if we acquire new observations $\boldsymbol{l_1}$ (i.e. another image in the sequence)?

- If we assume the model equation for this observation to be

$$g_1[\boldsymbol{p}, \boldsymbol{l_1}] = f_1[\boldsymbol{p}] - \boldsymbol{l_1} = \boldsymbol{0}$$

- Then (with the Jacobian $\boldsymbol{A_1} = \frac{\partial f_1}{\partial \boldsymbol{p}}$) the new joint Normal equation system for this problem is simply

$$\left(\boldsymbol{A_0^T C_{l_0 l_0}^{-1}}\ \boldsymbol{A_0} + \boldsymbol{A_1^T C_{l_1 l_1}^{-1}}\ \boldsymbol{A_1}\right)\boldsymbol{\Delta p'} = \boldsymbol{A_0^T C_{l_0 l_0}^{-1} \Delta l_0} + \boldsymbol{A_1^T C_{l_1 l_1}^{-1} \Delta l_1}$$

Kalman filter

- The solution for this new system could be obtained again as

$$\boldsymbol{\Delta p' = \left(A_0^T C_{l_0 l_0}^{-1} A_0 + A_1^T C_{l_1 l_1}^{-1} A_1\right)^{-1} \left(A_0^T C_{l_0 l_0}^{-1} \Delta l_0 + A_1^T C_{l_1 l_1}^{-1} \Delta l_1\right)}$$

- However, this would require the full inversion of the Normal equation matrix again and again in every step
- Instead we can use the following very useful matrix identity, for inverses of sums with outer products:

$$\boldsymbol{\left(N - A^T C^{-1} A\right)^{-1} = N^{-1} + N^{-1} A^T \left(C - A N^{-1} A^T\right)^{-1} A N^{-1}}$$

- Note, that the $\boldsymbol{N}$ only appears as inverse on the right hand side and that we already computed the covariance matrix

$$\boldsymbol{C_{pp} = \left(A_0^T C_{l_0 l_0}^{-1} A_0\right)^{-1}}$$

Kalman filter

- If we apply this to the inversion of the Normal equation matrix we get the following inverse in terms of the previously calculated covariance matrix

$$\left(\boldsymbol{A_0^T C_{l_0 l_0}^{-1} A_0} + \boldsymbol{A_1^T C_{l_1 l_1}^{-1} A_1}\right)^{-1} = \boldsymbol{C_{pp} - C_{pp} A_1^T \left(C_{l_1 l_1} + A_1 C_{pp} A_1^T\right)^{-1} A_1 C_{pp}}$$

- The following is called the **Kalman gain matrix**

$$\boldsymbol{F = C_{pp} A_1^T \left(C_{l_1 l_1} + A_1 C_{pp} A_1^T\right)^{-1}}$$

- Which is used to abbreviate the covariance matrix update as follows

$$\boldsymbol{C'_{pp} = (I - F A_1) C_{pp}}$$

- Note, that the inversion required for the Kalman gain is much smaller as it only involves the new observations in the latest image of the sequence

Kalman filter

- Using the updated covariance matrix (which is the inverse of the Normal equation matrix) we can calculate the parameter updated that is required taking the new observations into account to be

$$\begin{aligned}\boldsymbol{\Delta p'} &= \boldsymbol{C'_{pp}}\left(\boldsymbol{A_0^T C_{l_0 l_0}^{-1} \Delta l_0} + \boldsymbol{A_1^T C_{l_1 l_1}^{-1} \Delta l_1}\right) \\ &= \boldsymbol{\Delta \mathbf{p}} + \boldsymbol{C'_{pp} A_1^T C_{l_1 l_1}^{-1} \Delta l_1}\end{aligned}$$

- Using

$$\boldsymbol{C_{pp}}' = (\boldsymbol{I} - \boldsymbol{F A_1})\boldsymbol{C_{pp}}$$

- By maintaining the current covariance matrix of the parameters we can incrementally add more observations to the bundle adjustment
- This approach is called **Kalman filtering**

Modifying parameters

- The Kalman update equations

$$\boldsymbol{F = C_{pp} A_1^T \left(C_{l_1 l_1} + A_1 C_{pp} A_1^T \right)^{-1}}$$
$$\boldsymbol{C_{pp}{}' = (I - F A_1) C_{pp}}$$
$$\boldsymbol{\Delta \mathbf{p}' = \Delta \mathbf{p} + C'_{pp} A_1^T C_{l_1 l_1}^{-1} \Delta l_1}$$

- take as input only the parameter vector from the previous equation $\boldsymbol{\Delta \mathbf{p}}$ and it's covariance matrix $\boldsymbol{C_{pp}}$ and update these according to the new observations $\boldsymbol{l_1}$
- The history of observations, i.e. all previous images, are not relevant anymore, all past information is encoded in $\boldsymbol{\Delta \mathbf{p}}$ and $\boldsymbol{C_{pp}}$ at this stage

Modifying parameters

- This observation allows to modify the parameters between updates, called the **predict step** of the Kalman filter

- To do this we simply apply any function to the parameter vector

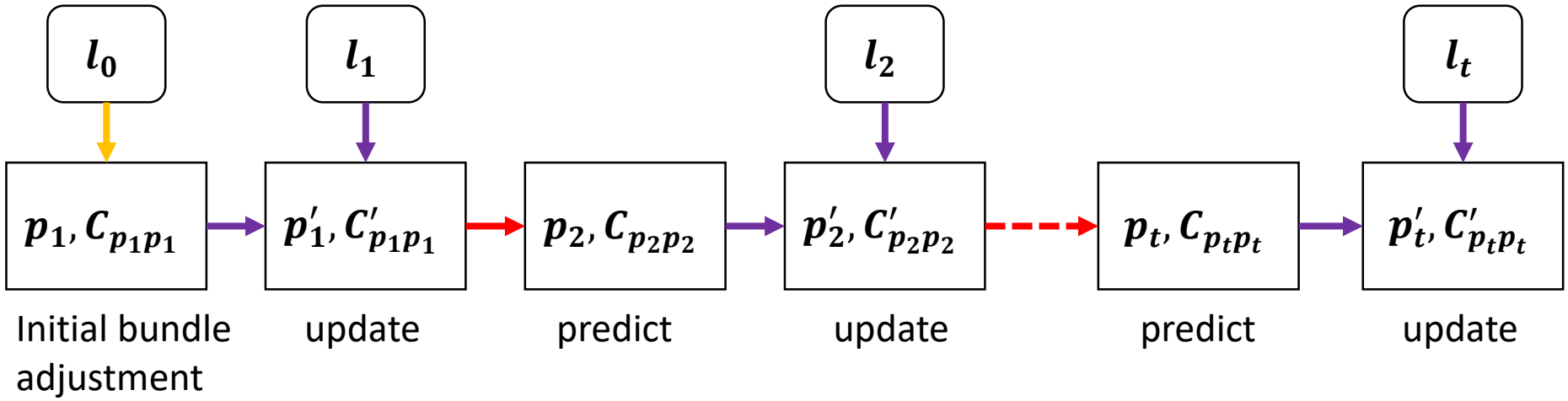
$$\boldsymbol{\Delta p_{t+1}} = h_t[\boldsymbol{\Delta p}]$$

- And modify the covariance matrix according to the error propagation rule (using the Jacobian $\boldsymbol{H} = \frac{\partial h_t}{\partial p}$)

$$\boldsymbol{C_{p_{t+1}p_{t+1}}} = \boldsymbol{H C_{p_t p_t} H^T}$$

Kalman filter algorithm

- By alternating the update and prediction step we can maintain a parameter vector (or state) over time, which is always updated with the most recent observations
- This approach lends itself very well to processing long image sequences, where every new frame creates a new set of observations



Adding and removing parameters

- The predict step can also be used to extend the parameter vector with add additional parameters (e.g. additional scene points and camera parameters)
- In this case we only append the initial values to the vector and supply an a-priori covariance matrix

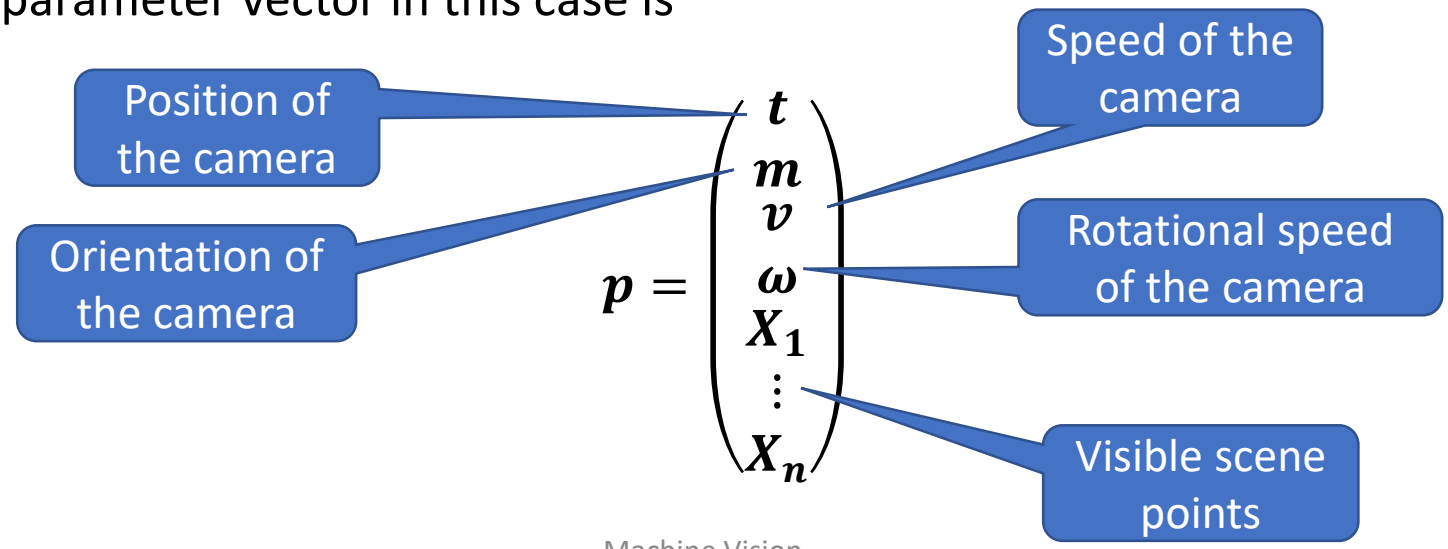
$$\boldsymbol{\Delta p_{t+1}} = \begin{pmatrix} \boldsymbol{\Delta p_{t+1}} \\ \boldsymbol{q} \end{pmatrix}$$

$$\boldsymbol{C_{p_{t+1}p_{t+1}}} = \begin{pmatrix} \boldsymbol{C_{p_t p_t}} & \\ & \boldsymbol{C_{qq}} \end{pmatrix}$$

- Similar we can also remove parameters that are no longer needed

Motion models & SLAM

- Instead of adding new camera parameters for every frame, we could modify the parameters of the current camera in the predict step and consider a single moving camera only
- This is possible, because in the update we only consider observations in the current frame, we therefore do not need to worry about how to calculate the reprojection into previous frames anymore
- We can even estimate the parameters of the motion itself in the process
- The parameter vector in this case is



Motion models & SLAM

- The prediction step is then

$$p_{t+1} = \begin{pmatrix} t + v \\ m + \omega \\ v \\ \omega \\ X_1 \\ \vdots \\ X_n \end{pmatrix}$$

Apply the motion to the camera

No a-priori acceleration

Keep the 3d points for subsequent frames

- Additional scene points can be added and obsolete scene points can be removed

Motion models & SLAM

- The Jacobian in this case is

$$H = \begin{pmatrix} I_3 & & I_3 & \\ & I_3 & & I_3 \\ & & I_3 & \\ & & & I_3 \\ & & & & I_n \end{pmatrix}$$

The off-diagonal elements are applying correlating the motion model with the position and rotation

- So that the covariance matrix transforms according to

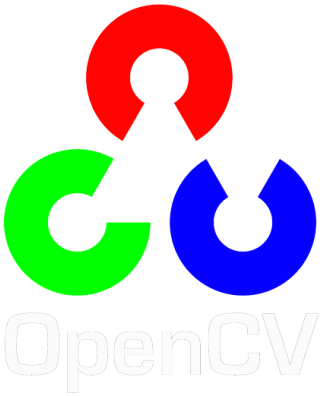
$$C_{p_{t+1}\, p_{t+1}} = H C_{p_t p_t} H^T$$

Wrap-up: module overview

- Introduction & Tools
- Image processing
- Content extraction
- Multi-view geometry
- Scene reconstruction

Tools

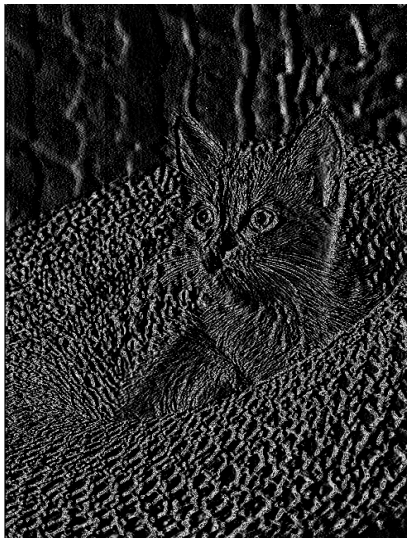
- OpenCV
  - Loading/saving images and videos
  - Capturing from a camera
  - Image processing functions
  - Calibration

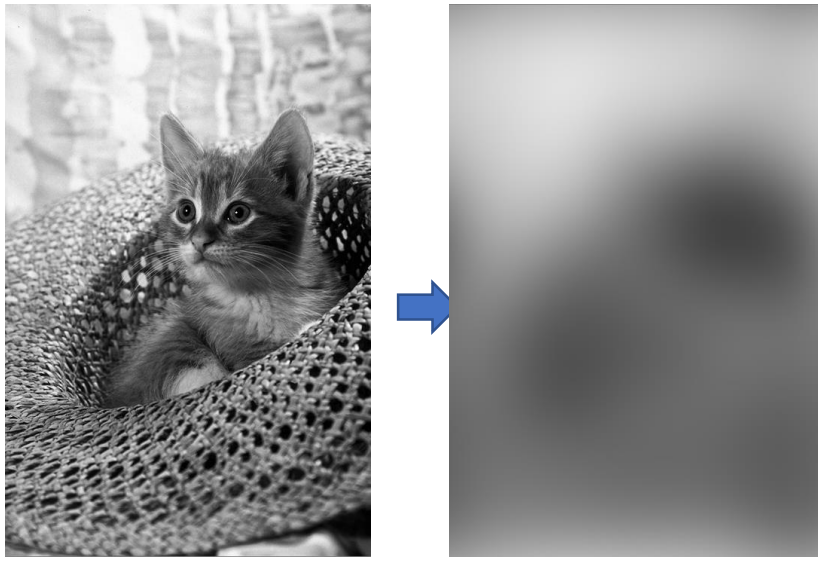


- NumPy
  - Image representation and basic processing
  - Linear algebra

Image processing

- Linear shift-invariant systems and convolution
- Gaussian smoothing and edge detection





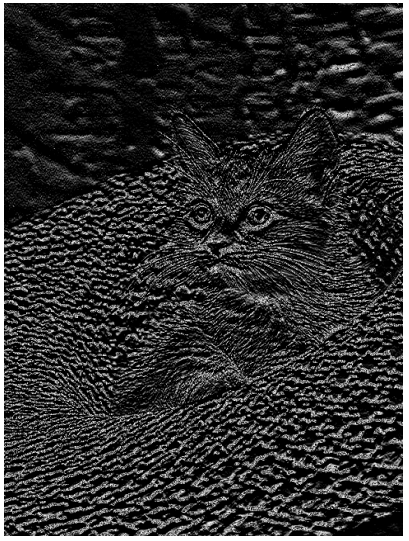


Image processing

- Fourier spectrum and frequency analysis

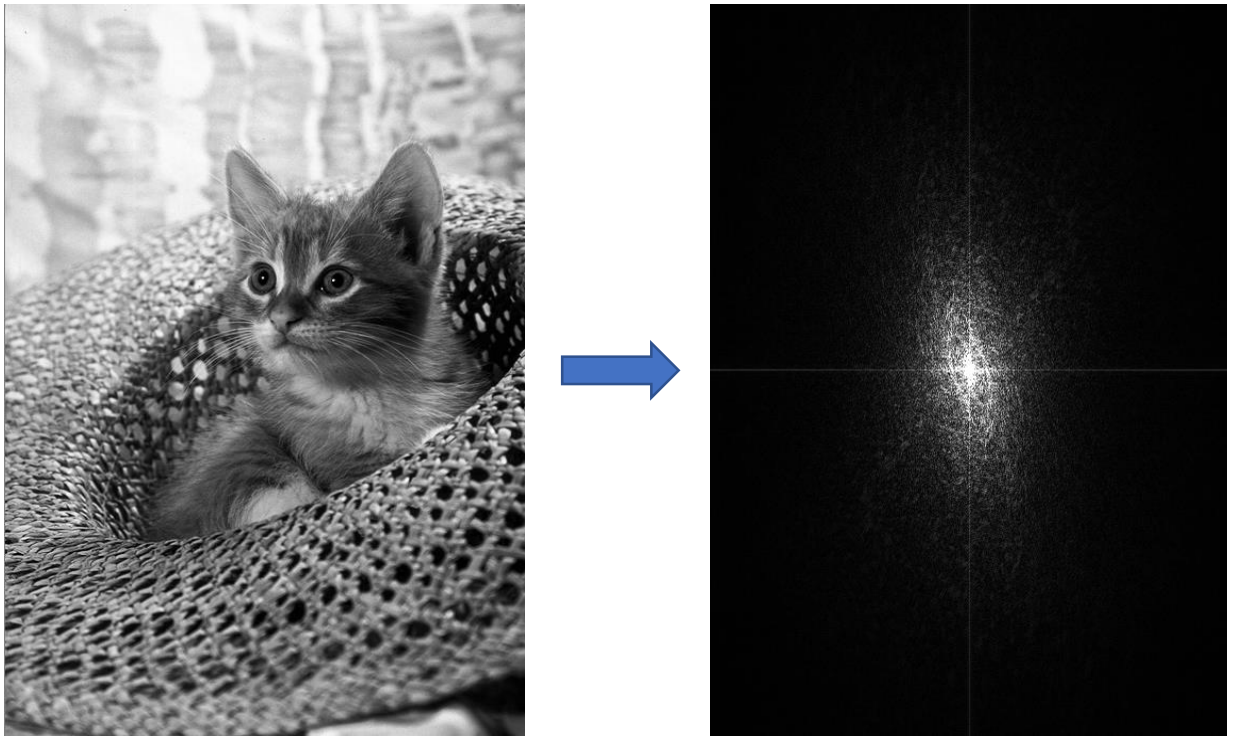
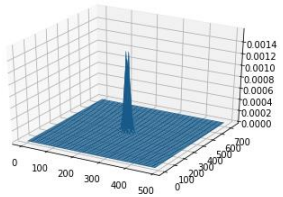


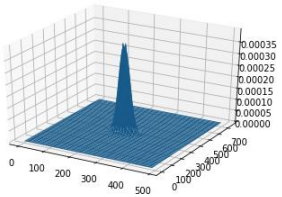
Image processing

- Processing images at different scales

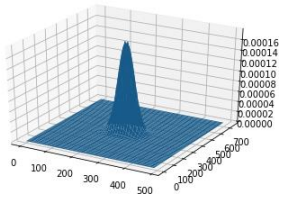
$\sigma = 10$

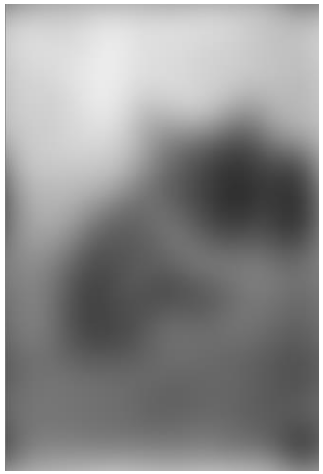


$\sigma = 20$

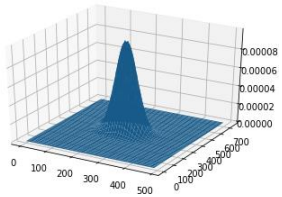


$\sigma = 30$





$\sigma = 40$



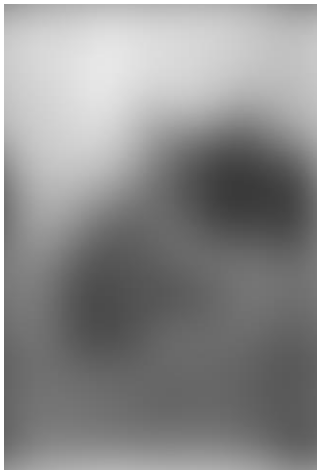


Image processing

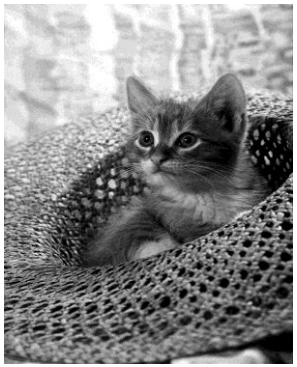
- Radiometric depth and discretisation

8-bit

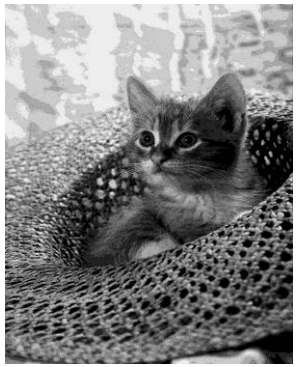
7-bit

6-bit

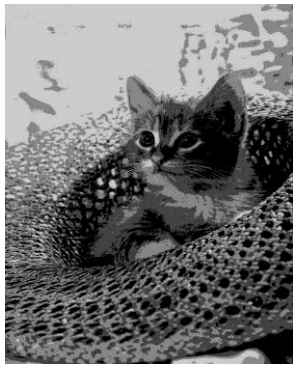
5-bit



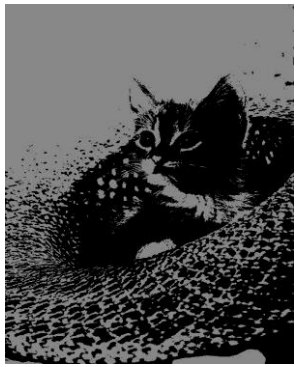
4-bit



3-bit



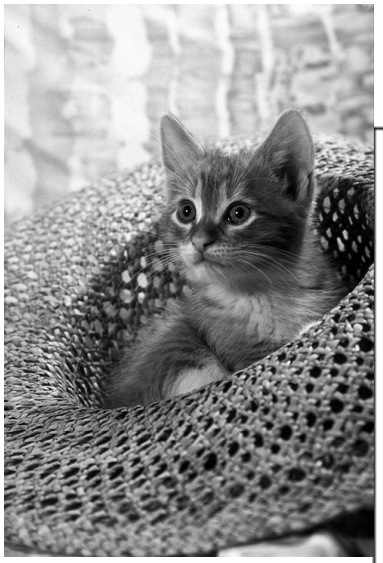
2-bit



1-bit

Image processing

- Histogram operations



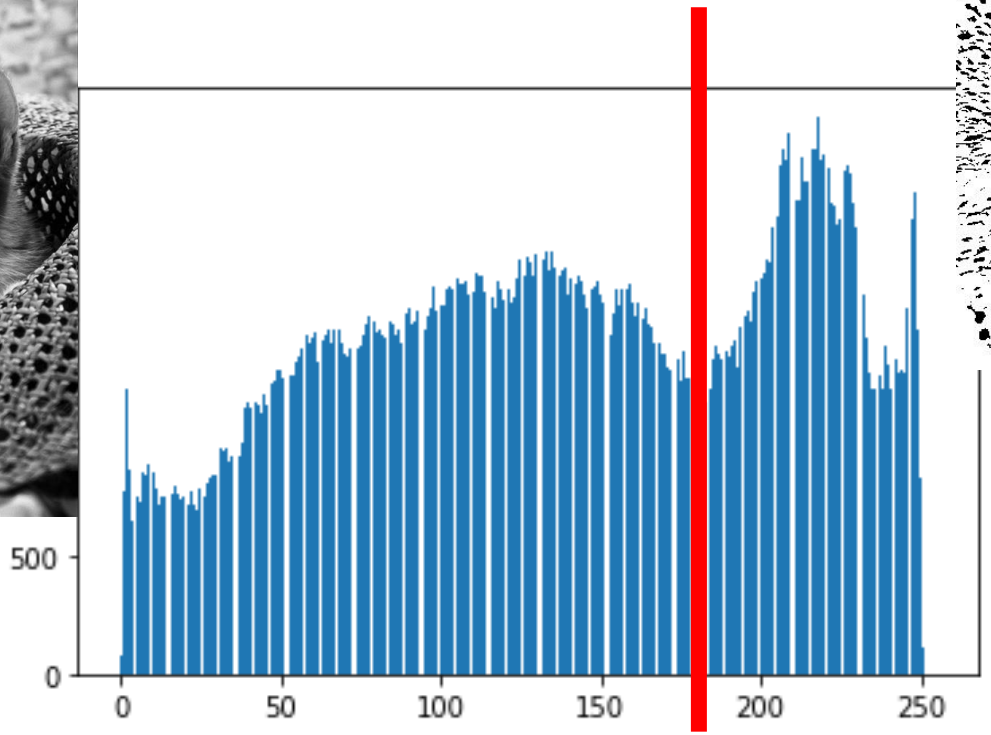
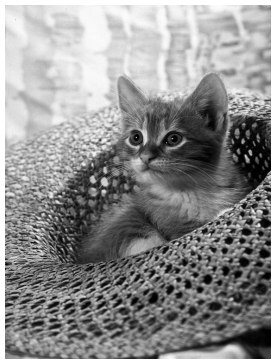


Image processing

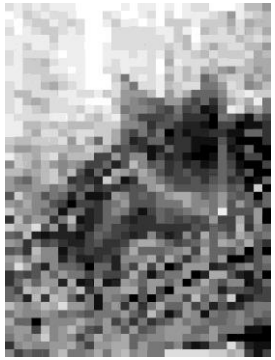
- Geometric resolution and sampling

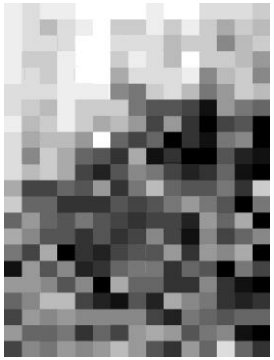

733x490

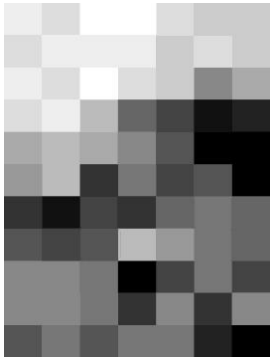

366x245


183x122


91x61


45x30


22x15


11x7

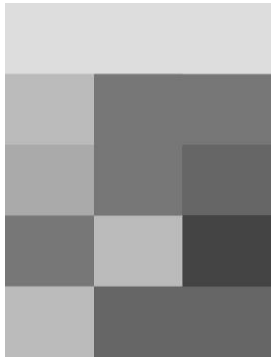
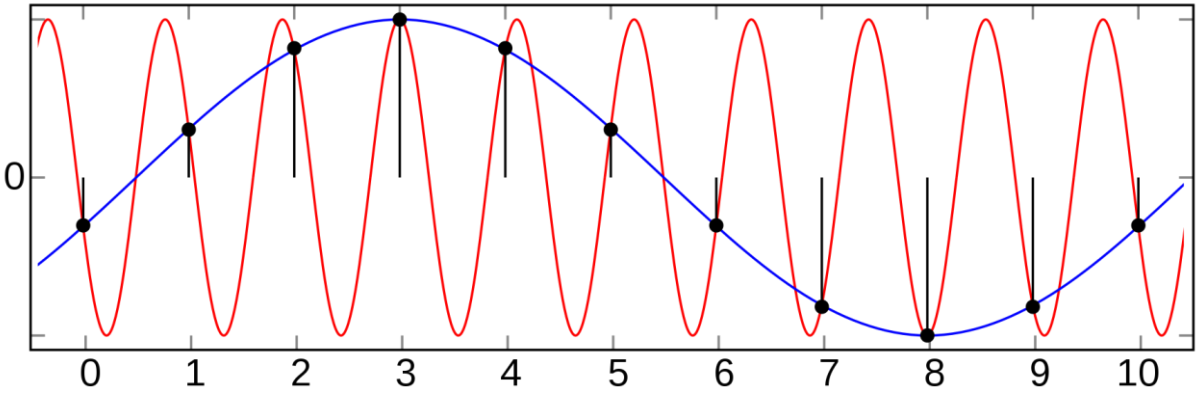
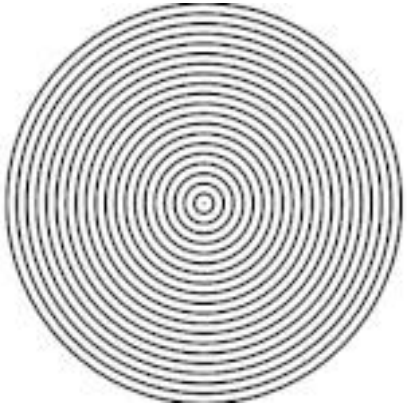

5x3

Image processing

- Nyquist frequency and aliasing





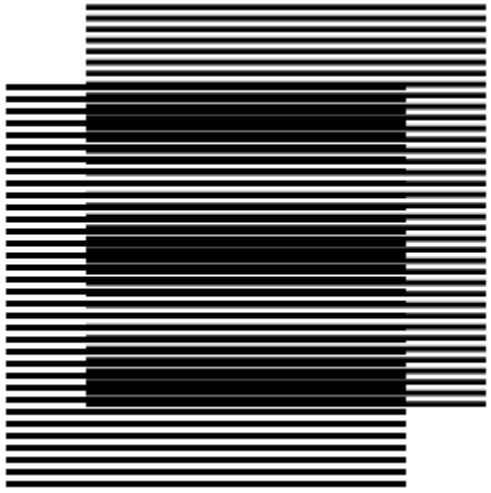
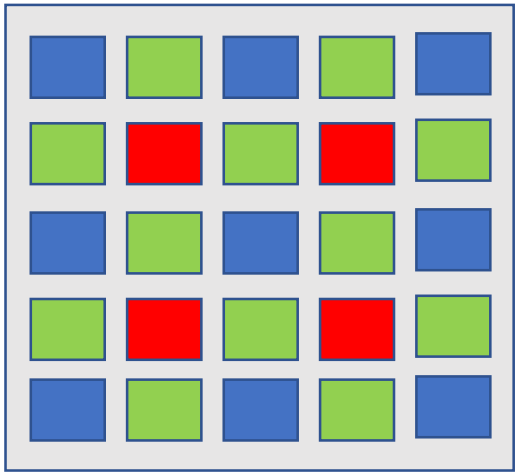


Image processing

- Colour image sensors and Bayer patterns



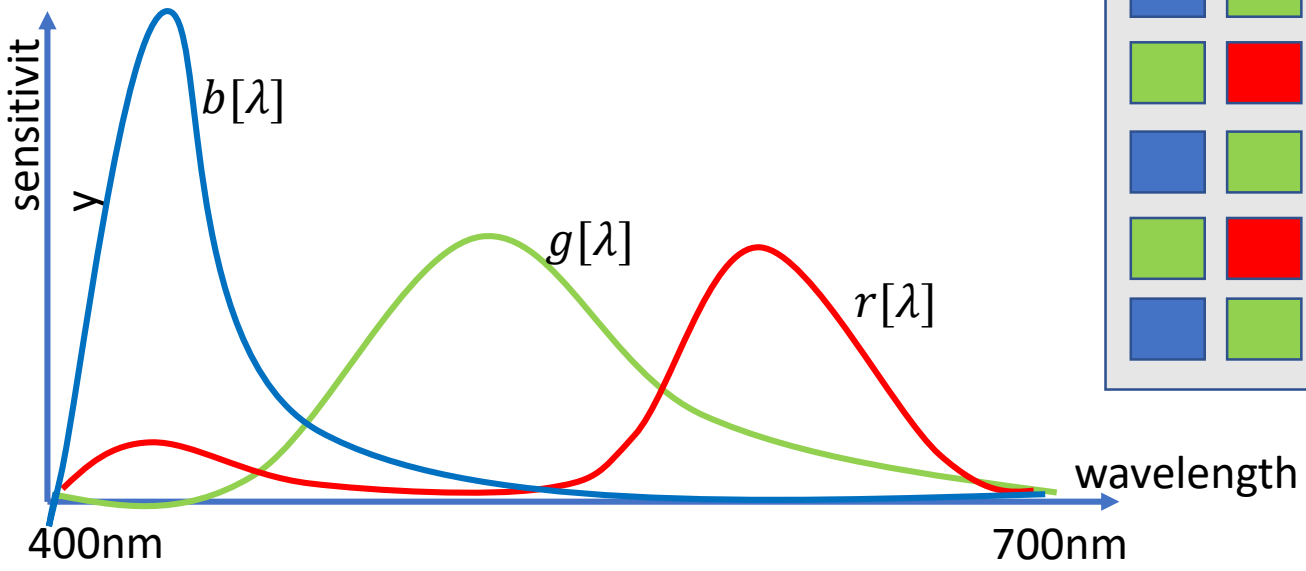
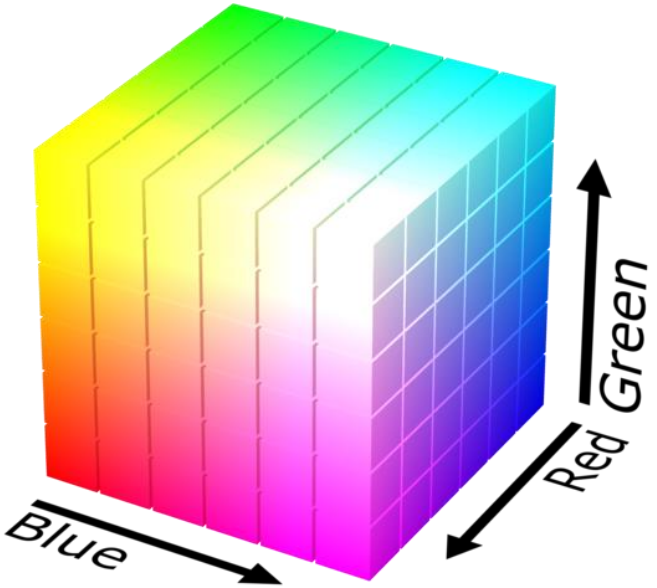
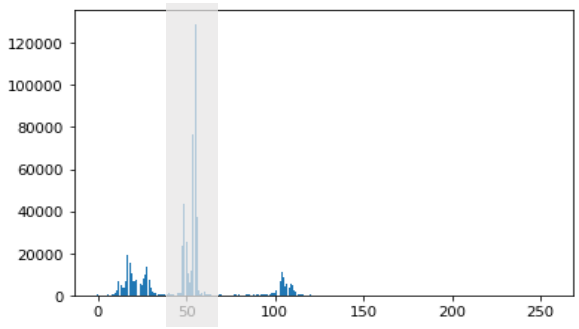


Image processing

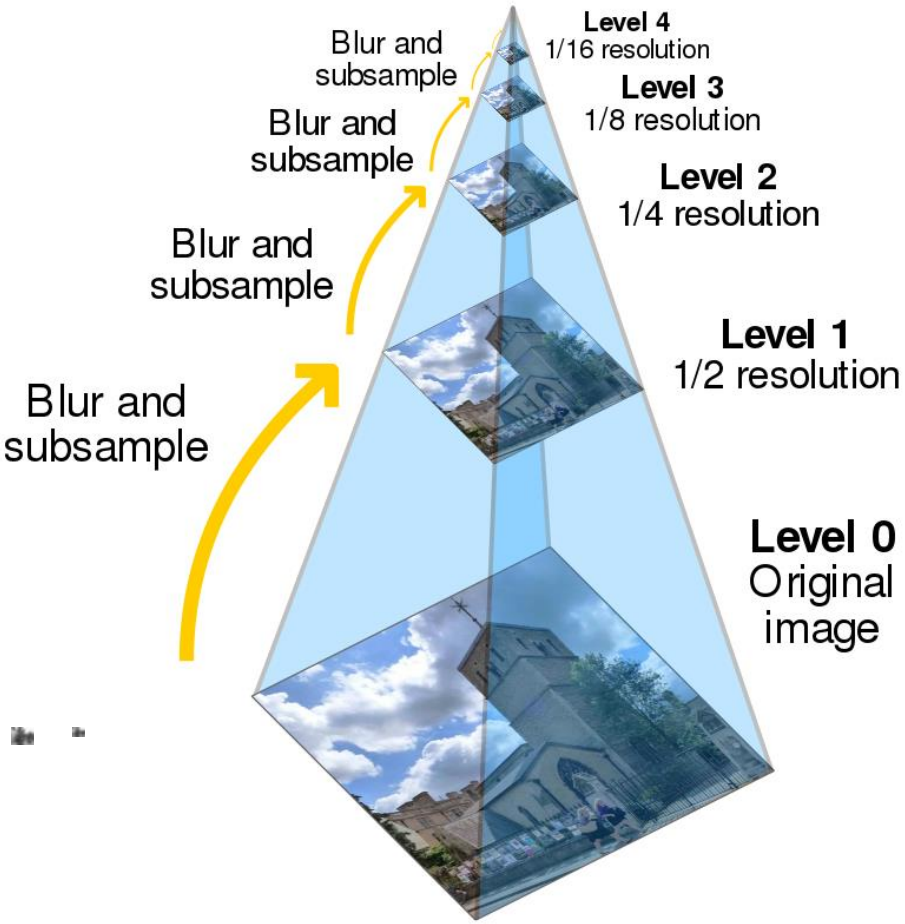
- Colour representation and chroma key segmentation





Content extraction

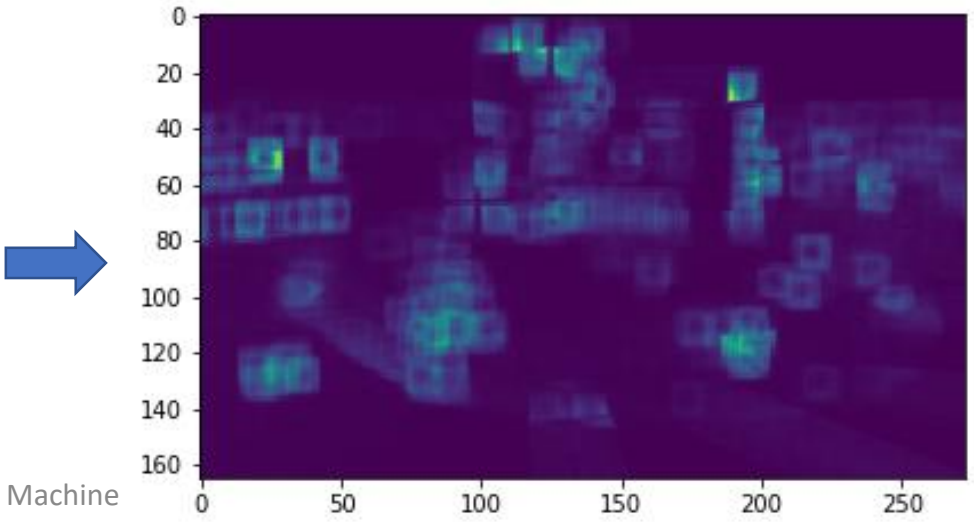
- Scale-space and image pyramids



Content extraction

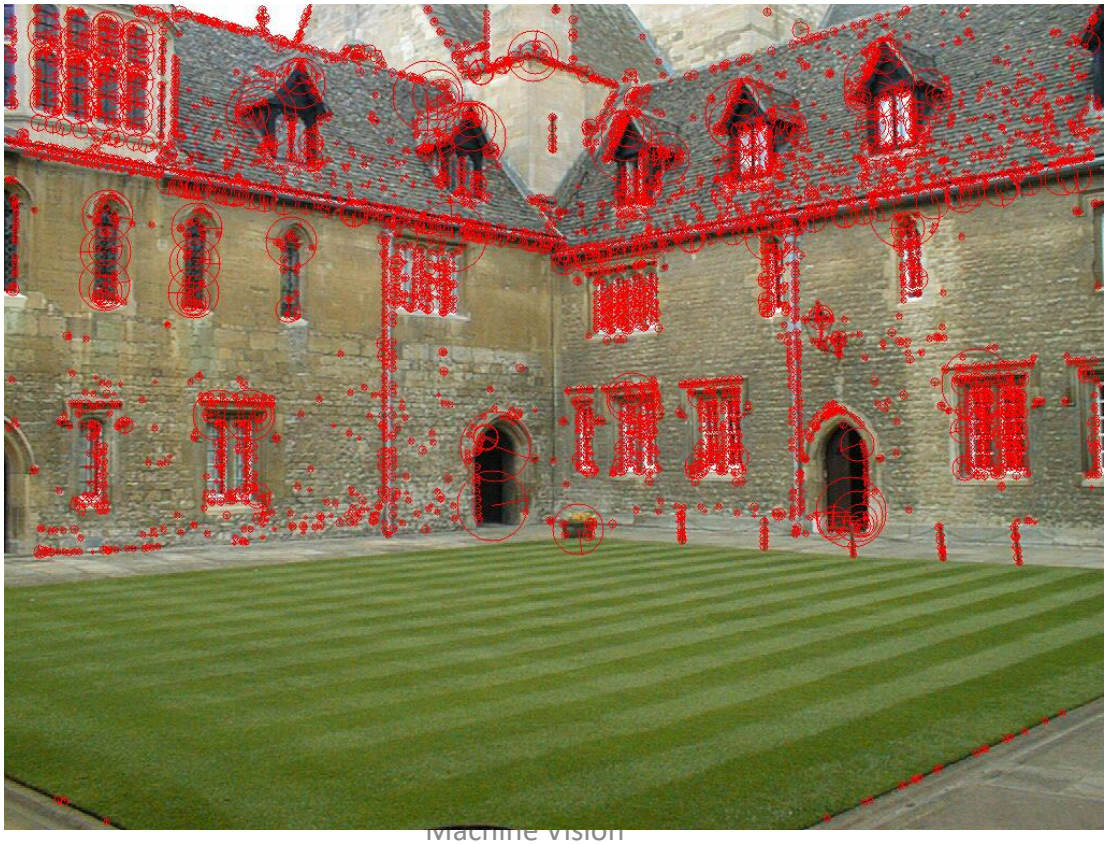
- Feature extraction and the structure tensor

$$\sum_i w_{\sigma_w}[|p - p_i|] \begin{pmatrix} g^2_{x;\sigma_d}[p_i] & g_{x;\sigma_d}[p_i] g_{y;\sigma_d}[p_i] \\ g_{x;\sigma_d}[p_i] g_{y;\sigma_d}[p_i] & g^2_{y;\sigma_d}[p_i] \end{pmatrix}$$



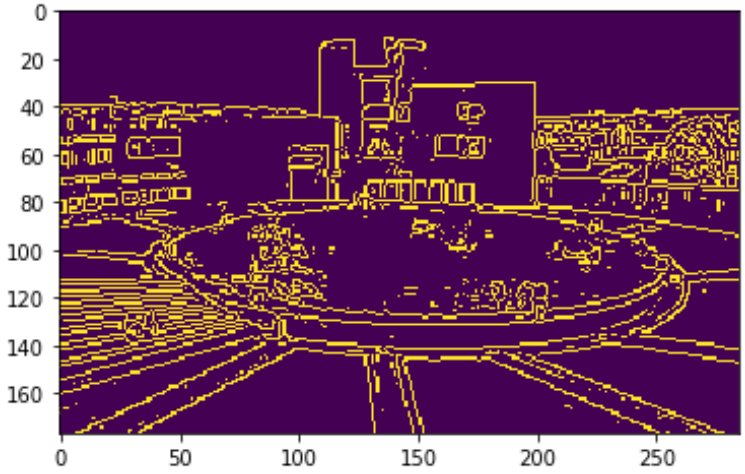
Content extraction

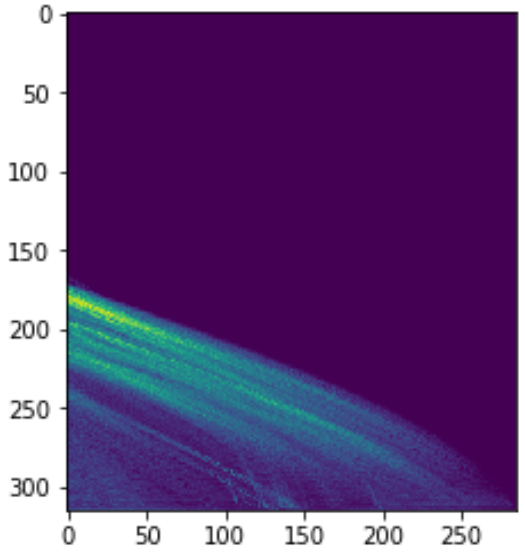
- Scale-invariant feature points

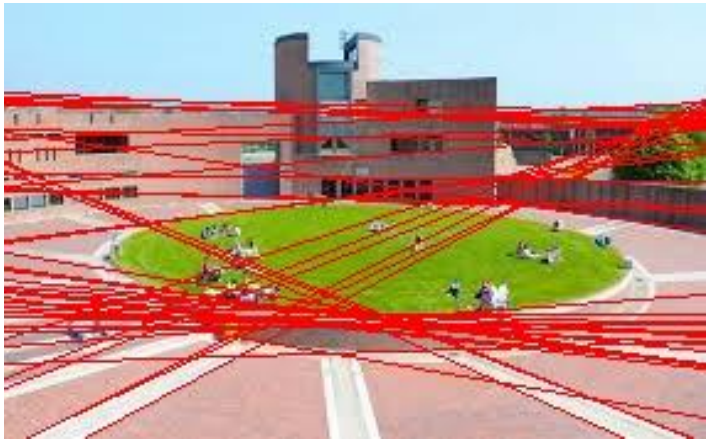


Content extraction

- Hough-transformation

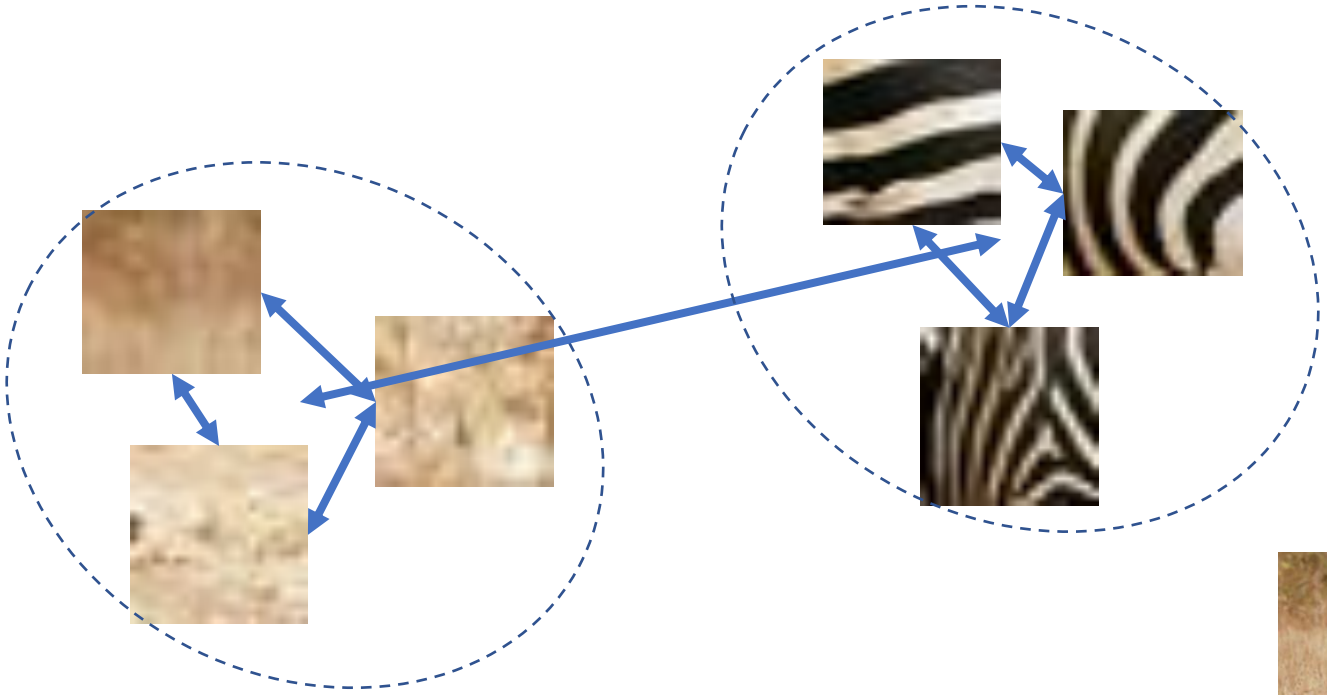






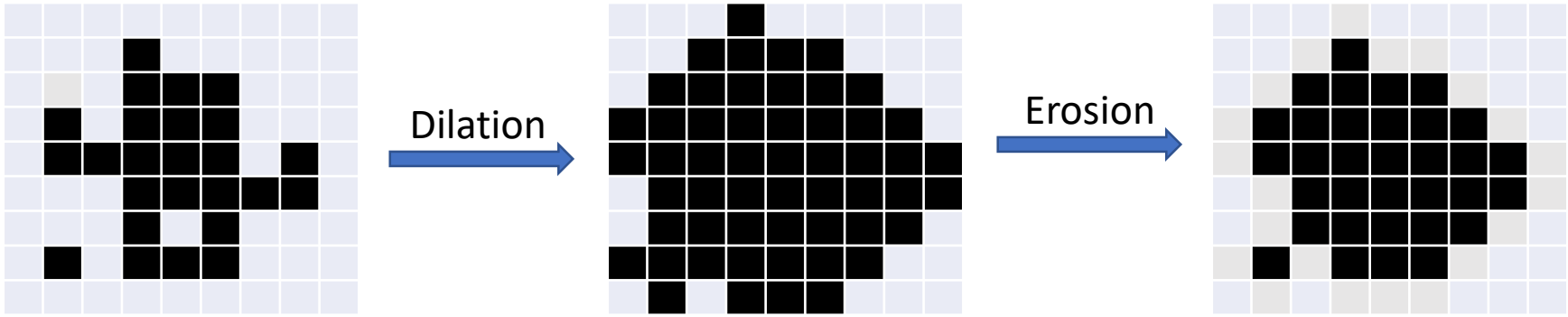
Content extraction

- Texture segmentation



Content extraction

- Binary image segmentation and morphological operations

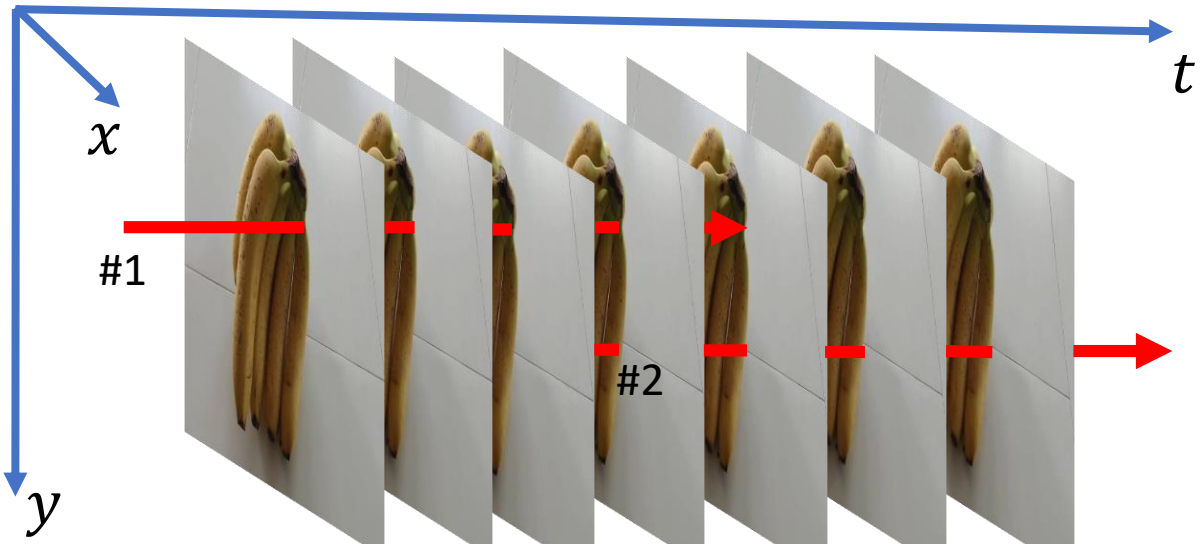


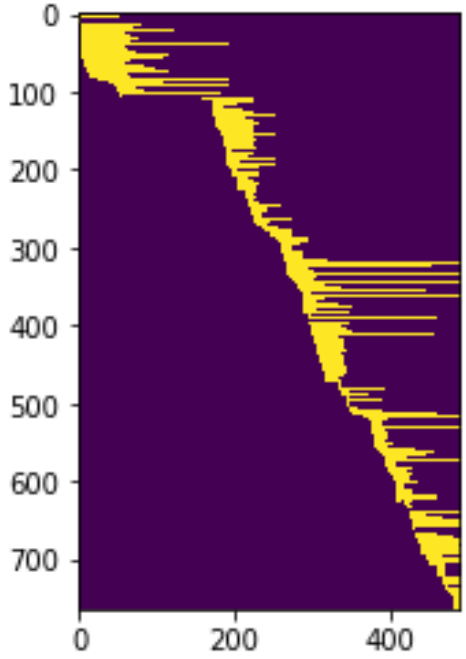
Content extraction

- Photometric stereo

Content extraction

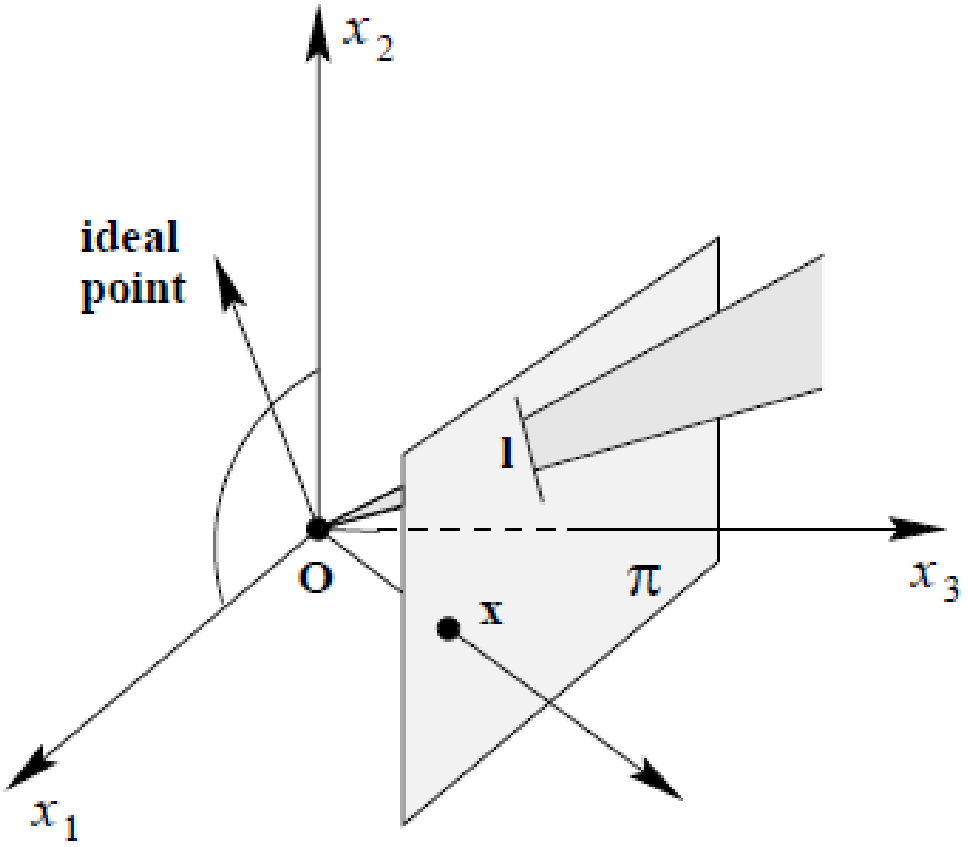
- Optical flow and feature tracking





Multi-view geometry

- Projective geometry and homogeneous coordinates



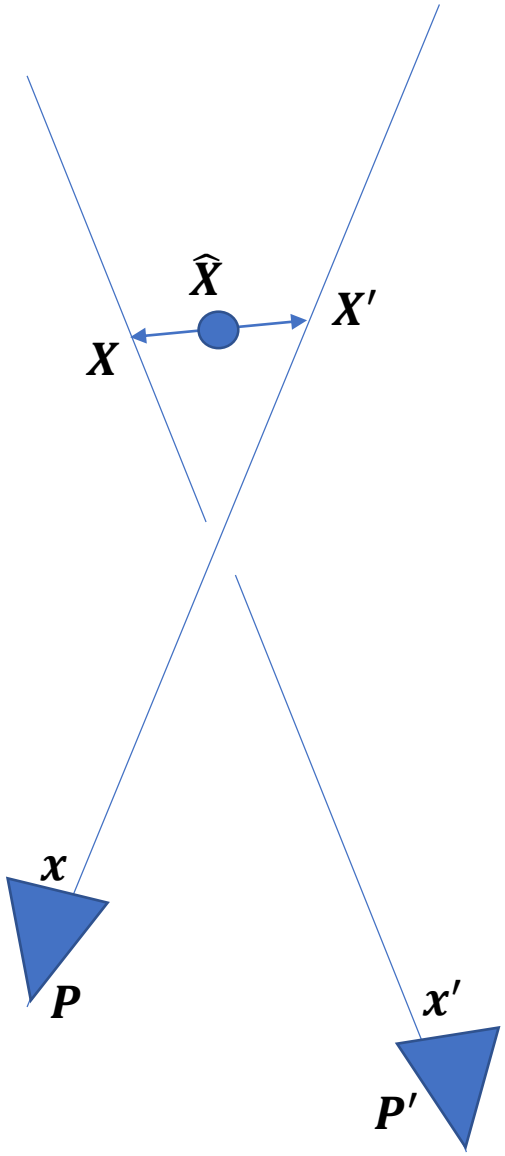
Multi-view geometry

- Homographies and image stiching

Multi-view geometry

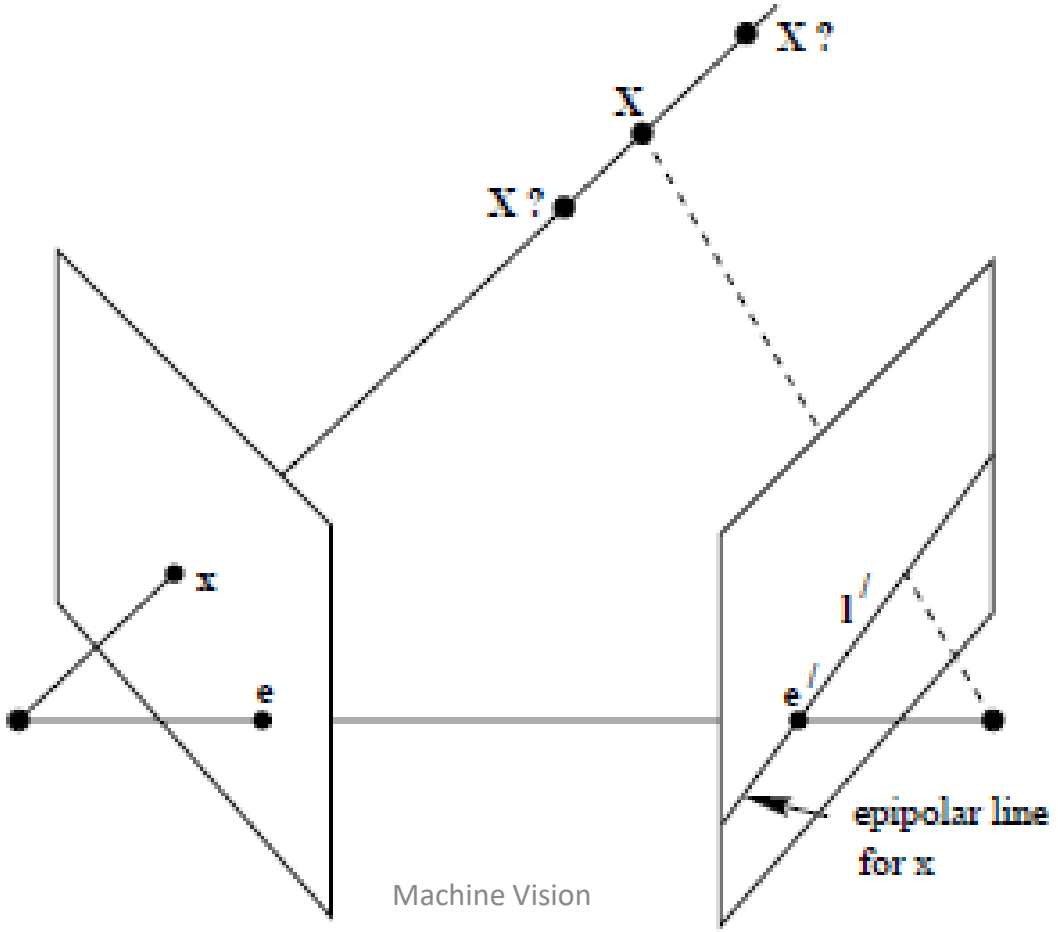
- The projective camera

$$\boldsymbol{x} = \boldsymbol{PX}$$
$$= \boldsymbol{KR}^T[\boldsymbol{I}_{3\times3} \quad -\boldsymbol{t}]\boldsymbol{X}$$



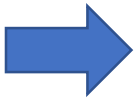
Multi-view geometry

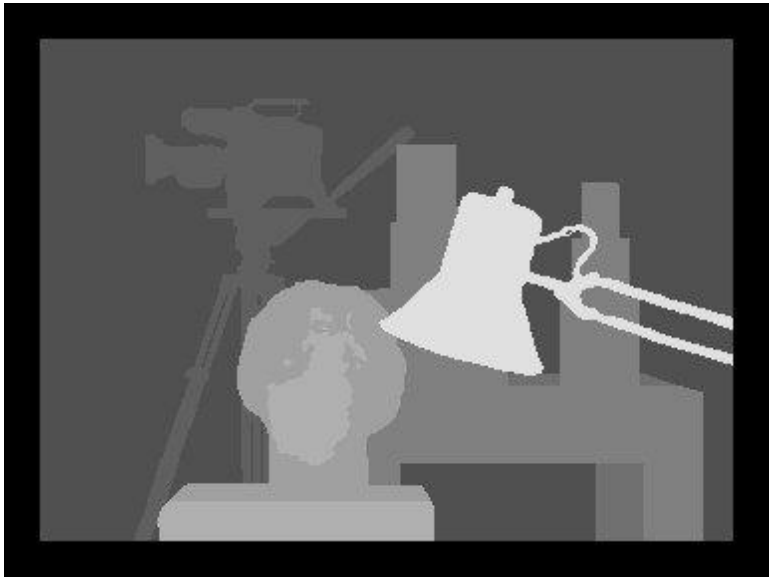
- Epipolar geometry



Multi-view geometry

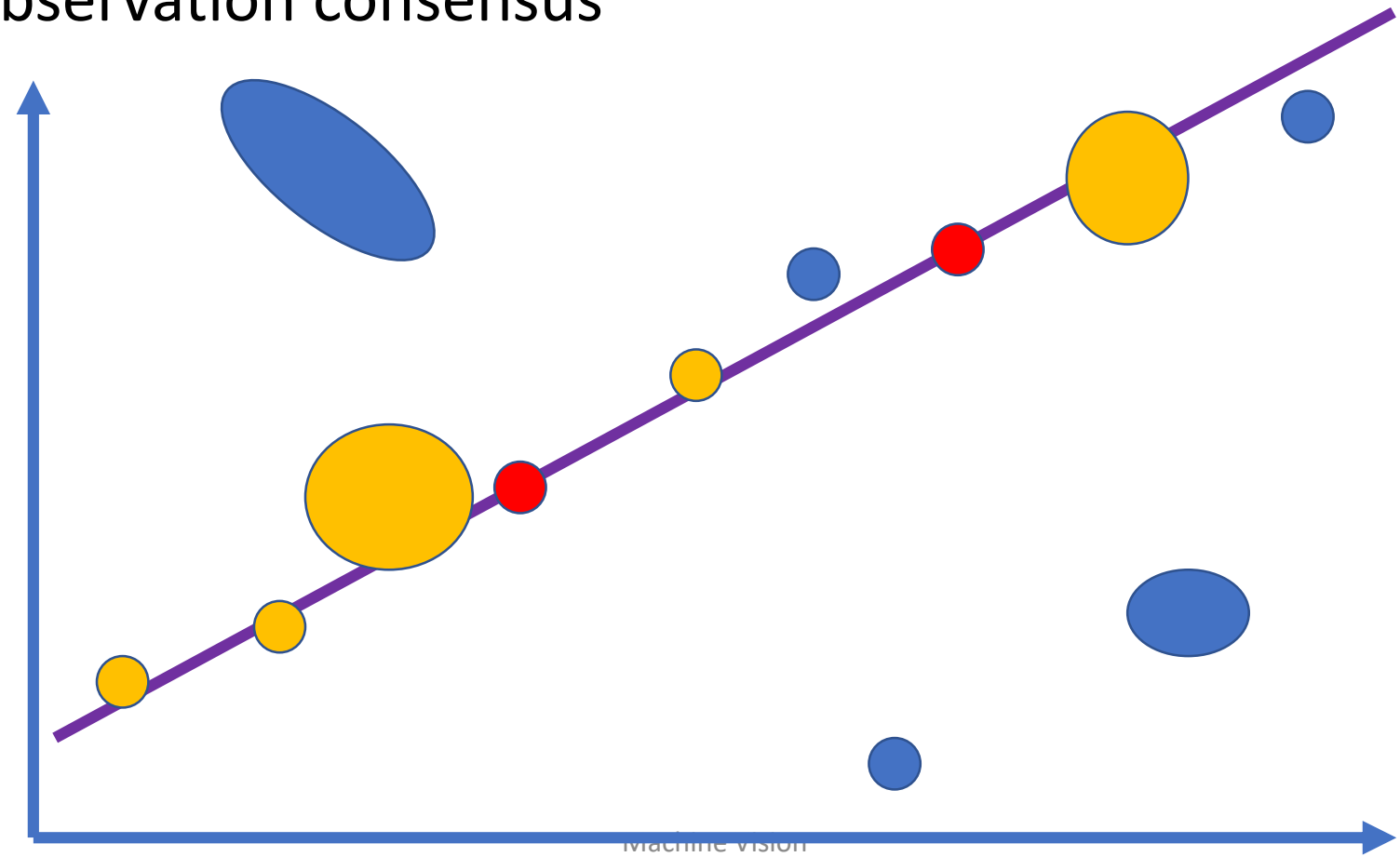
- Disparity and dense stereo





Multi-view geometry

- Robust estimation, direct minimal solutions and observation consensus

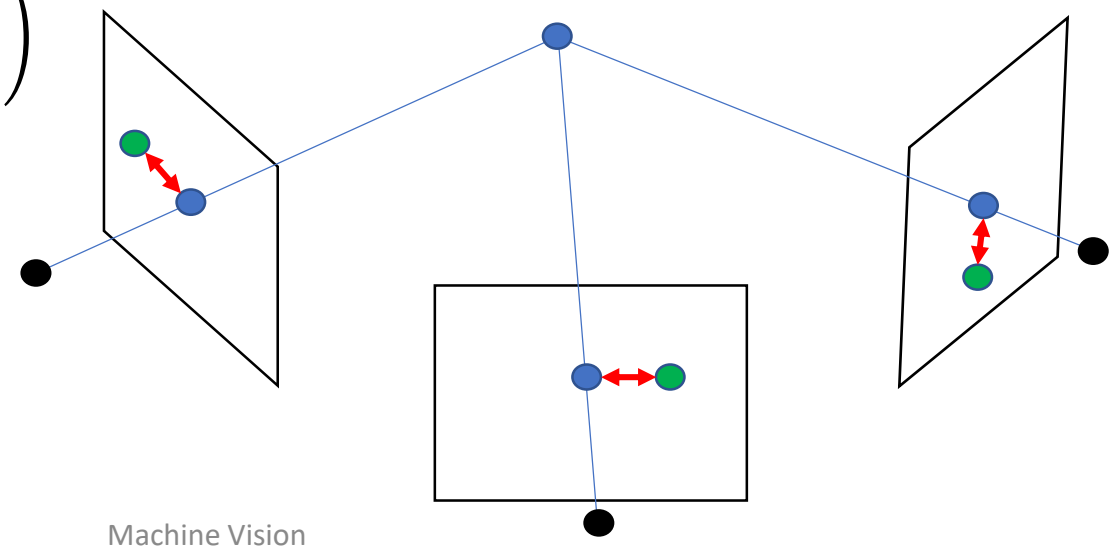


Multi-view geometry

- Reprojection error and bundle adjustment

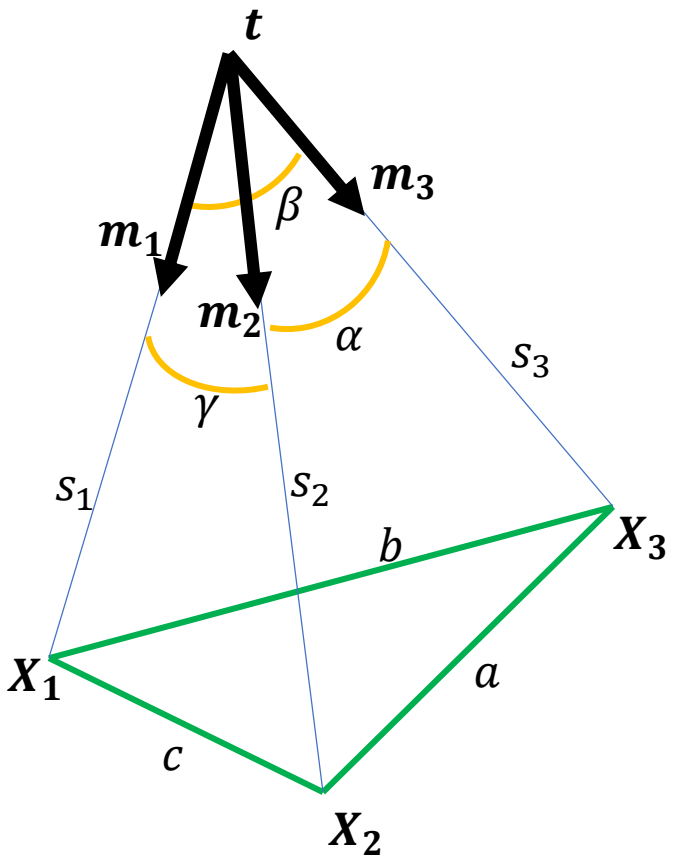
$$\begin{pmatrix} \boldsymbol{\Delta p} \\ \boldsymbol{\mu} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A^T\left(BCB^T\right)^{-1}A} & \boldsymbol{H^T} \\ \boldsymbol{H} & \boldsymbol{0} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{A^T\left(BCB^T\right)^{-1}(B(l_0 - l) - g_0)} \\ \boldsymbol{-h_0} \end{pmatrix}$$

$$\boldsymbol{C'_{pp}} = \begin{pmatrix} \boldsymbol{A^T\left(BCB^T\right)^{-1}A} & \boldsymbol{H^T} \\ \boldsymbol{H} & \boldsymbol{0} \end{pmatrix}^{-1}$$



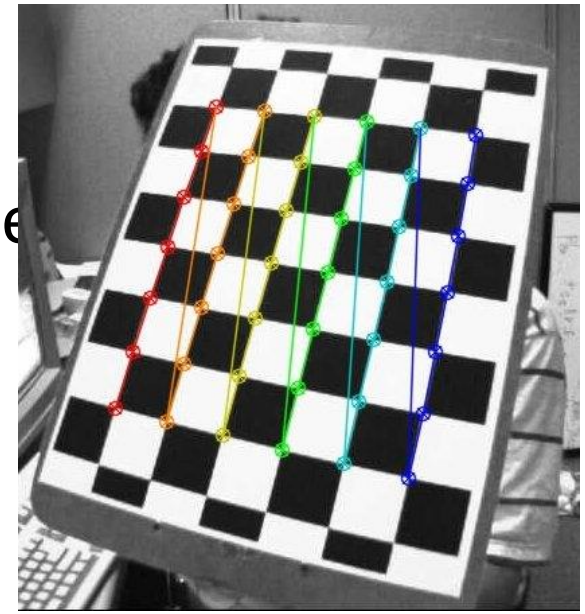
Multi-view geometry

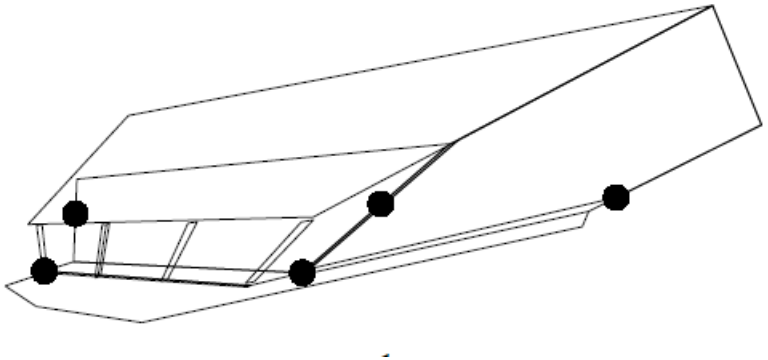
- Spatial resection

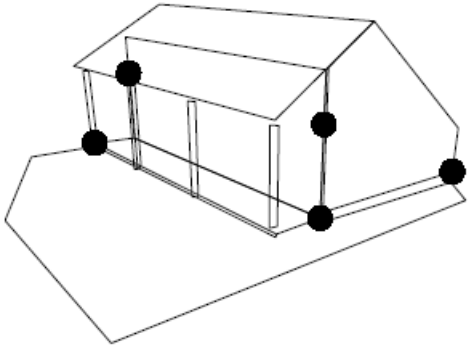


Multi-view geometry

- Camera calibration and metric scene reconstruction







Thank you for your attention!