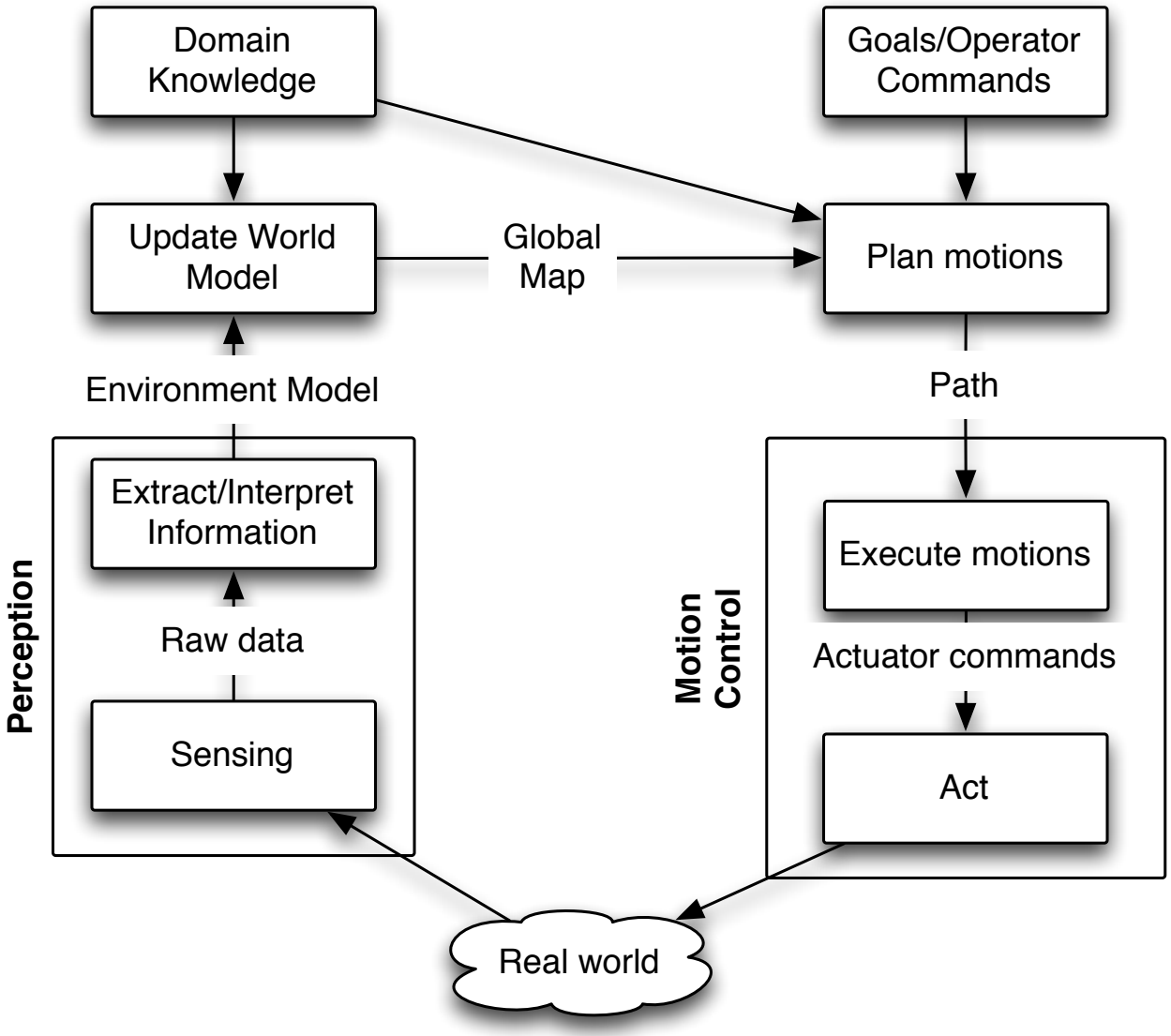
CS3027: Sensors and Perception

N. Oren
n.oren@abdn.ac.uk

University of Aberdeen

Where are we?


Domain Knowledge
Goals/Operator Commands
Update World Model
Global Map
Plan motions
Environment Model
Path
Perception
Extract/Interpret Information
Raw data
Sensing
Motion Control
Execute motions
Actuator commands
Act
Real world

Sensors

- Sensors are the key component for perceiving the environment.
- An understanding of their physical principles enables appropriate use
  - To select the correct sensor for a given application.
  - To properly model the sensor system — resolution, sensing speed, uncertainties.

Sensors

- http://www.youtube.com/watch?v=x4-E-IUKr3c
- http://www.youtube.com/watch?v=I8cRAQi-oqA&feature=relmfu
- http://www.youtube.com/watch?v=so9axknlftk

Case Studies

- Warehouse robot

Case Studies

- Warehouse robot — indoor structured environment
  - Line following sensor
  - Wheel encoders
  - Bump sensor?

Case Studies

- Warehouse robot — indoor structured environment
  - Line following sensor
  - Wheel encoders
  - Bump sensor?
- Robot arm

Case Studies

- Warehouse robot — indoor structured environment
  - Line following sensor
  - Wheel encoders
  - Bump sensor?
- Robot arm — indoor structured environment
  - Encoders/angle sensors
  - Task specific sensor (temperature probe etc)
  - Pressure sensor

Case Studies

- Warehouse robot — indoor structured environment
  - Line following sensor
  - Wheel encoders
  - Bump sensor?
- Robot arm — indoor structured environment
  - Encoders/angle sensors
  - Task specific sensor (temperature probe etc)
  - Pressure sensor
- Museum tour guide

Case Studies

- Warehouse robot — indoor structured environment
  - Line following sensor
  - Wheel encoders
  - Bump sensor?
- Robot arm — indoor structured environment
  - Encoders/angle sensors
  - Task specific sensor (temperature probe etc)
  - Pressure sensor
- Museum tour guide — indoor unstructured environment
  - Wheel encoders
  - Distance sensor ring
  - Pan-tilt camera?

Case Studies

- Robot car

Case Studies

- Robot car — outdoor unstructured environment
  - GPS
  - Accelerometers, wheel encoders
  - Laser rangefinder(s)
  - internal sensors (door status etc)
  - Cameras

Case Studies

- Robot car — outdoor unstructured environment
  - GPS
  - Accelerometers, wheel encoders
  - Laser rangefinder(s)
  - internal sensors (door status etc)
  - Cameras
- Autonomous aircraft

Case Studies

- Robot car — outdoor unstructured environment
  - GPS
  - Accelerometers, wheel encoders
  - Laser rangefinder(s)
  - internal sensors (door status etc)
  - Cameras
- Autonomous aircraft — outdoor unstructured environment
  - Inertial navigation system - gyros, accelerometers
  - Altimeter
  - GPS
  - Cameras

Case Studies

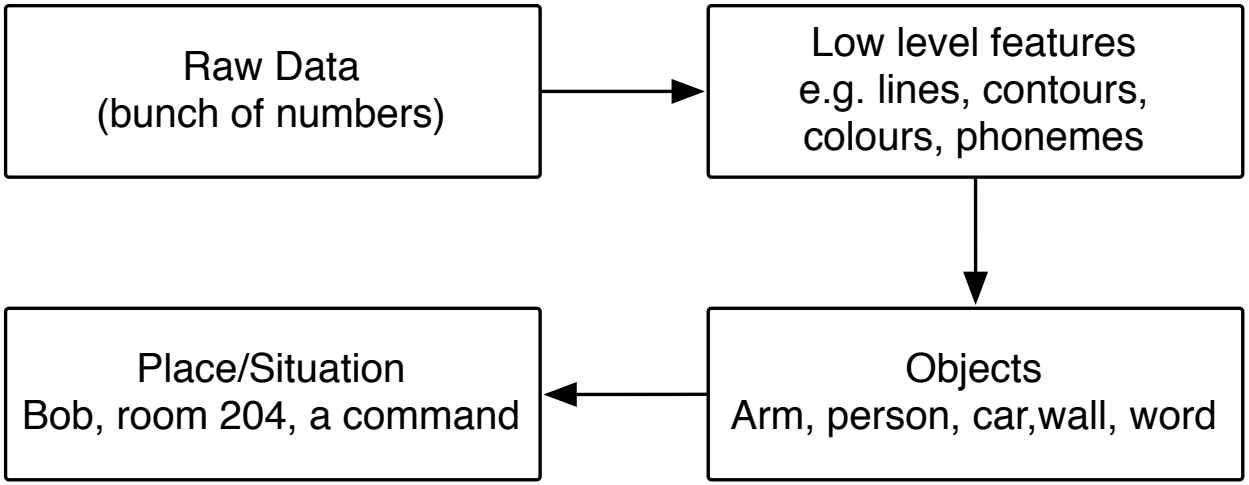
- Different domains have very differing requirements.
- Factors to consider
  - price
  - physical size
  - energy usage
  - data provided
- Camera — rich information, inexpensive, noise, no distance provided
- Laser rangefinder — high precision, expensive, little information

Sensor Data

Sensor Data

- Sensor information needs to be interpreted

Information Processing



Sensor Taxonomy

- What:
  - Proprioceptive sensors
    - Measure internal system (robot) values
    - Motor speed, wheel load, heading, battery status etc
  - Exteroceptive sensors
    - Environmental information
    - Distance to objects, light intensity, temperature, etc
- How:
  - Passive sensors
    - Measure energy coming from the environment
    - Camera, thermometer, microphones, encoders
  - Active sensors
    - Emit energy and measure the reaction.
    - Better performance than passive sensors, but affect environment
    - Sonar, laser

Sensor classification

General classification (typical use)	Sensor Sensor System	PC or EC	A or P
Tactile sensors (detection of physical contact or closeness; security switches)	Contact switches, bumpers	EC	P
	Optical barriers	EC	A
	Noncontact proximity sensors	EC	A
Wheel/motor sensors (wheel/motor speed and position)	Brush encoders	PC	P
	Potentiometers	PC	P
	Synchros, resolvers	PC	A
	Optical encoders	PC	A
	Magnetic encoders	PC	A
	Inductive encoders	PC	A
	Capacitive encoders	PC	A
Heading sensors (orientation of the robot in relation to a fixed reference frame)	Compass	EC	P
	Gyroscopes	PC	P
	Inclinometers	EC	A/P

A, active; P, passive; P/A, passive/active; PC, proprioceptive; EC, exteroceptive.

Sensor classification

General classification (typical use)	Sensor Sensor System	PC or EC	A or P
Ground-based beacons (localization in a fixed reference frame)	GPS	EC	A
	Active optical or RF beacons	EC	A
	Active ultrasonic beacons	EC	A
	Reflective beacons	EC	A
Active ranging (reflectivity, time-of-flight, and geometric triangulation)	Reflectivity sensors	EC	A
	Ultrasonic sensor	EC	A
	Laser rangefinder	EC	A
	Optical triangulation (1D)	EC	A
	Structured light (2D)	EC	A
Motion/speed sensors (speed relative to fixed or moving objects)	Doppler radar	EC	A
	Doppler sound	EC	A
Vision-based sensors (visual ranging, whole-image analysis, segmentation, object recognition)	CCD/CMOS camera(s)	EC	P
	Visual ranging packages		
	Object tracking packages		

Sensor performance

- Dynamic range
  - Informally: upper limit - lower limit of input values
  - Formally: ratio of maximum input value to minimum measurable input value
  - Measured in decibels: $10\log_{10}$(dynamic range)
  - Example: sensor measures current from 10mA to 20A

$$10\log(20/0.01) = 33dB$$

  - Decibels are intended to measure power - anything else needs to be converted (e.g. square of voltage is proportional to power, so calculation must be multiplied by 2).
- Resolution
  - minimum difference between two values.
  - usually measured at the lower limit of range
  - Example: 8 bit A/D converter of voltage from 0-5V. $5/255 = 20mV$

Sensor Performance

- Linearity
  - How the output signal changes as the input signal changes.
  - Linear response is often desired

$$x \rightarrow f(x) \qquad y \rightarrow f(y)$$

$$\alpha x + \beta y \rightarrow f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$$

- Bandwidth or frequency
  - Measures the speed at which a sensor can provide a stream of readings
  - Measurements per second = speed in hertz
  - Often affects the maximum speed a robot can move

Sensors in the real world

- Sensitivity
  - Degree to which an incremental change in the target input signal changes the output signal
  - In real environments, highly sensitive sensors are easily affected by environmental changes, e.g. illumination.
- Cross-sensitivity
  - sensitivity to environmental parameters other than the target parameters (e.g. temperature, magnetic field).
  - Example: compass is very sensitive to magnetic north. Also sensitive to ferrous materials — high cross-sensitivity makes it useless indoors.
  - Cross-sensitivity is undesirable, especially when it cannot be modelled.

Sensors in the real world

- Error
  - difference between the sensor's output measurement (m) and the true value (v) being measured (in some operating context).

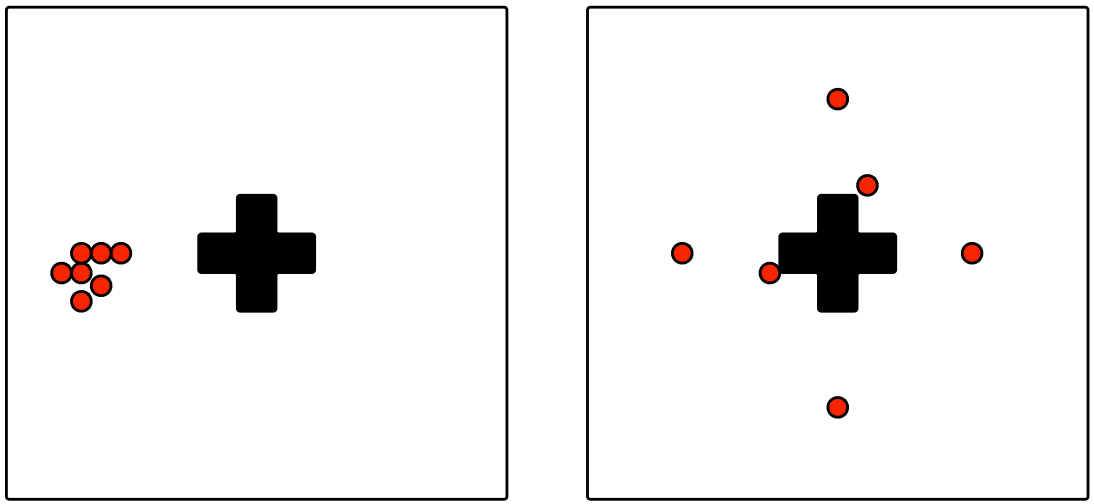
$$accuracy = 1 - \frac{|m - v|}{v}$$

  - Accuracy: degree of conformity between measurement and true value. Small error = high accuracy.
- Systematic errors
  - Caused by factors that can (in theory) be modelled.
  - Errors are predictable and deterministic
  - E.g. poor calibration of a rangefinder
- Random errors
  - Cannot be predicted
  - Are non-deterministic, but can be described in probabilistic terms
  - Noise in image

Precision

- Precision is often confused with accuracy.
- High precision means results are reproducible
- A sensor with high precision would return the same results for multiple readings of the same environment state.
- If a sensor's random error has mean ν and standard deviation σ

$$precision = \frac{range}{\sigma}$$



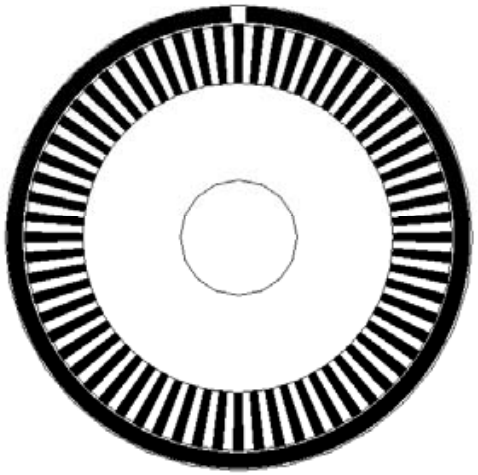
Sensors

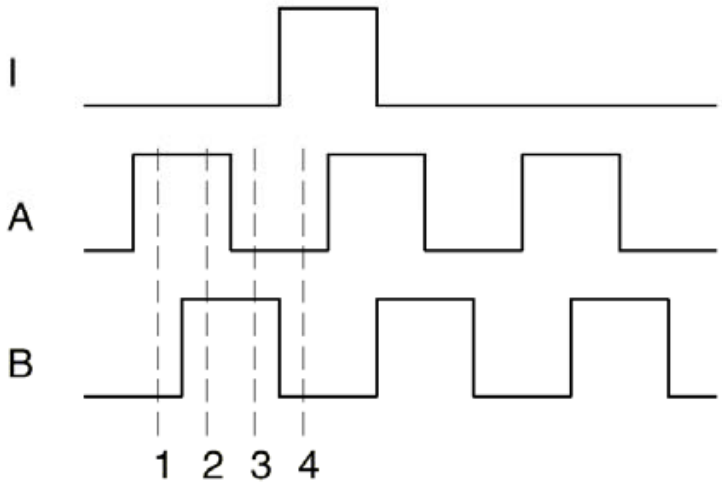
- Optical encoders
- Heading sensors
- Accelerometer
- IMU
- GPS
- Range sensors
- Vision

Encoders

- A device that converts linear or angular position of a shaft to an a signal, making it a linear/angular transducer.
- Use cases
  - Measure position or speed of wheels or steering
  - Integrate wheel movements to get an estimate of position (odometry)
  - Optical encoders are proprioceptive sensors
  - Typical resolution 64-2048 increments per revolution
  - Interpolation for higher resolution
- A regular encoder cannot tell the direction of motion

Quadrature encoder





State	Ch A	Ch B
S_1	High	Low
S_2	High	High
S_3	Low	High
S_4	Low	Low

- Quadrature encoder uses two sensors in quadrature-phase shift. Ordering of which encoder produces rising edge first tells direction. A single slot in outer track generates a reference pulse per revolution.
- Quadrature encoder with 2000 holes per revolution gives 8000 transitions per resolution.
- Very controlled, so no systematic error or cross sensitivity — errors arise from wheels, terrain etc.
- Low cost, high resolution

Heading Sensors

- Heading sensors determine the robot's orientation and inclination.
- Can be properioceptive (gyroscope, accelerometer) or exteroceptive (compass, inclinometer).
- Together with velocity information, can be used to obtain a position estimate via dead reckoning

Compass

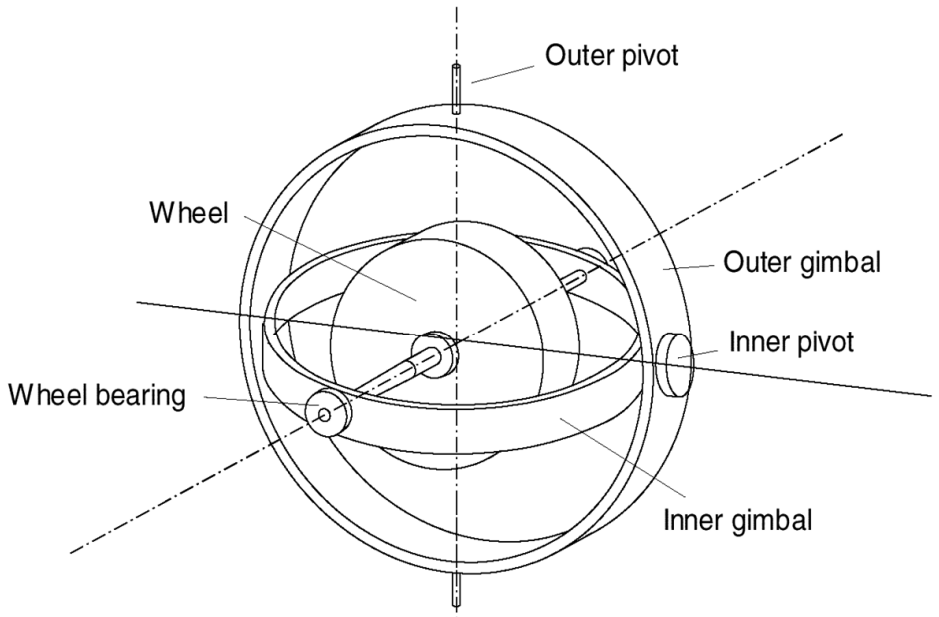
- Used by humans since before 2000 B.C.E, by birds for migration
- Utilises the magnetic field on earth
- Various types — mechanical, gyrocompass, hall-effect and magnetoresistive.
- Drawbacks:
  - Earth's magnetic field is very weak
  - Easily disturbed by magnetic objects nearby
  - Low bandwidth (0.5 Hz), susceptible to vibrations
  - Not suitable for absolute orientation indoors
  - But locally useful indoors

Gyroscope

- Gyroscopes are heading sensors that preserve their orientation in relation to a fixed reference frame.
- Provide an absolute measure for the heading of a mobile system.
- Two categories: mechanical and optical.

Mechanical Gyroscope

- Inertial properties of a fast spinning rotor (angular momentum) keeps axis of gyroscope inertially stable.
- No torque can be transmitted from the outer pivot to wheel axis.
- Spinning axis therefore space-stable
- Friction in axes bearings introduce torque and so drift.
- High quality gyro: 0.1^o in 6 hours. Cost: around £70 000.
- Alignment with earth necessary to prevent errors.
- Rate gyro measures angular speed using torsional springs on gimbals.



Optical Gyroscopes

- Optical gyroscope measures angular speed only.
- Uses two light sources (lasers) travelling in opposite direction.
- The beam travelling in direction opposite to rotation has slightly shorter path.
- Phase difference between beams can be measured.
- Typically around 3-5km of optical fibre. Bandwidth 100KHz, resolution < 0.0001 degrees/hour

Some videos

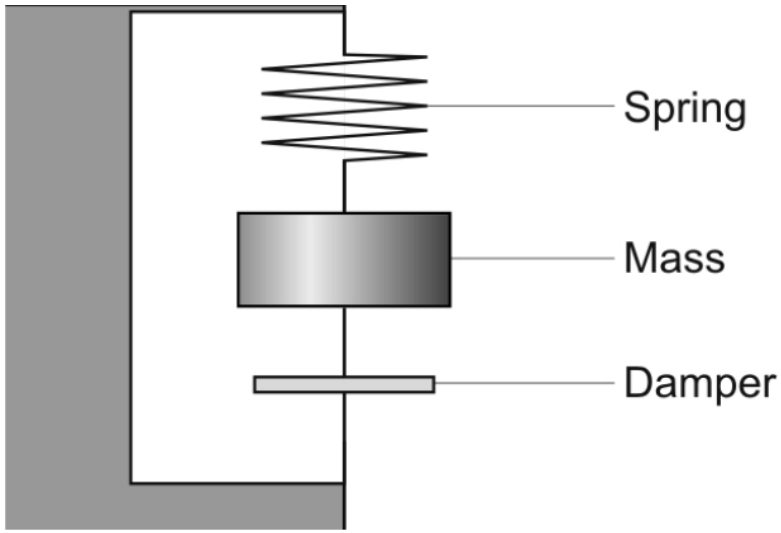
- http://www.youtube.com/watch?v=A4hzCcFikm0
- http://www.youtube.com/watch?v=AriuYTqxAMg
- http://www.youtube.com/watch?v=D-fhQgktRyc
- http://www.youtube.com/watch?v=blRPGKn5Cak

Accelerometers

- Accelerometers measure all external forces acting on them (including gravity).
- Acts like a spring-mass-damper system, with position of proof mass being measured.

$$F_{applied} = F_{intertial} + F_{damping} + F_{spring} = m\ddot{x} + c\dot{x} + kx$$

- at steady state $a_{applied} = \frac{kx}{m}$



Accelerometer

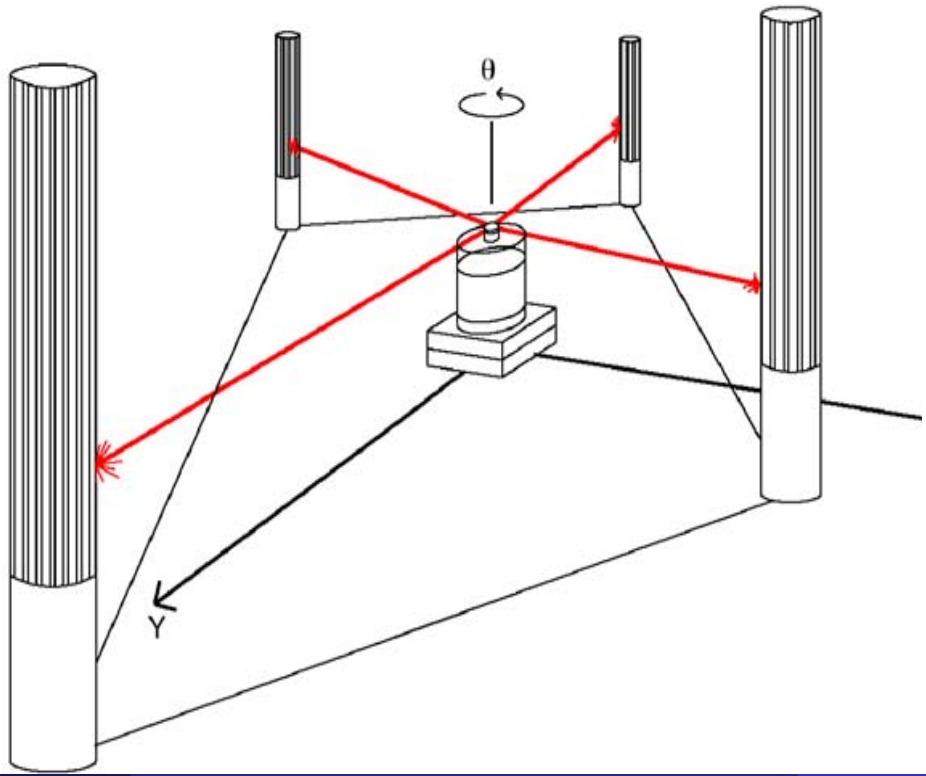
- On Earth's surface, accelerometer will always indicate 1g along vertical axis
- Gravity must be subtracted — zero output during free fall
- Bandwidth up to around 50KHz
- Measurement of accelerometer is only along 1 axis; orthogonal accelerometers are needed to obtain 3D accelerometer.
- Micro ElectroMechanical Systems (MEMS) accelerometers exist, as do capacitive and piezoelectric accelerometers. The latter can only capture changes in acceleration.

Inertial Measurement Unit (IMU)

- An IMU uses measurement systems such as gyroscopes and accelerometers to estimate relative position (x,y,z) orientation (roll, pitch, yaw), velocity and acceleration of a moving vehicle.
- To estimate motion, gravity must be subtracted and initial velocity known.
- IMUs are very sensitive to measurement errors (e.g. drift in gyro), double integration of accelerometer data.
- Over time, all IMUs drift — an external reference (e.g. GPS or camera) must be used to correct it.

Beacons

- One way to solve localisation problem is through the use of beacons — signalling guiding devices with a precisely known location.
- Humans use such beacons when localising — stars, mountains, sun, lighthouses.
- Beacons require changes in the environment, and cannot adapt to changing environments



GPS

- Global Positioning System (GPS) useful for outdoor localisation.
- Satellites act as beacons.
- 24 satellites orbiting every 12 hours at around 20000km altitude.
- 4 satellites located in each of 6 orbits with 60 degrees of orientation between each one.
- Each satellite continuously transmits orbital location (ephermis) and current time data
- Receiver computes location through trilateration and time correction.
- Challenges:
  - Time synchronisation between satellites and GPS receiver
  - Real time update of the exact satellite location
  - Time of flight measurement
  - interferences with other signals

GPS

- Time synchronizatoin
  - Each satellite has an atomic clock
  - Monitored from ground station.
  - Precise time is critical
- Signals travel at speed of light (0.3m/nanosecond). Position accuracy proportional to precision of time measurement.
- Real time update of satellite location.
  - Satellite position is monitored from ground station.
  - Master station analyses measurements and transmits actual position to each of the satellites.
- Exact measurement of time of flight.
  - Receiver correlates a pseudocode with same code coming from satellite.
  - Delay time for best correlation represents time of flight
  - Clock on receiver not very precise
  - 4 satellites needed to identify x, y, z and clock correction.
- accuracies of a few meters (outdoors)
- DGPS uses a known base station at some location. Compares own data with base station data. Accuracy: <1m to a few cm.

Active Range Sensors

- Range sensors provide measurements of distance from robot to objects in its vicinity.
- Sonar
- Laser range finder
- Time of flight camera
- Structured light

Time of flight active ranging

- Ultrasonic sensors and laser range sensors use propagation speed of sound or electromagnetic waves to determine distance.

$$d = c \cdot t$$

- d distance travelled (round trip); c is speed of wave propagation; t is time of flight.

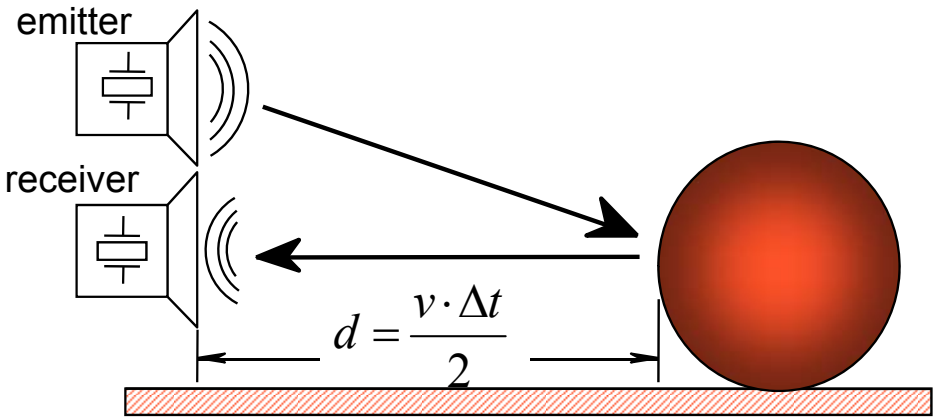
Time of flight

- Propagation speed of sound $0.3m/ms$ (300m/s). Of light $0.3m/ns$ (300000000m/s)
- Electromagnetic signals travel 10^6 times faster.
- 3m: 10ms for ultrasonic, 10ns for laser range sensor.
- Measuring time of flight with electromagnetic signals is very difficult. Expensive and delicate sensors.

Time of flight

- Quality of time of flight range sensors depends on
  - inaccuracies in time of flight measurements
  - Opening angle of transmitted beam
  - Interactions with target
  - Variation of propagation speed
  - Speed of mobile robot and target

Ultrasonic Sensor



<http://www.robot-electronics.co.uk/shop/Ultrasonic_Rangers1999.htm>

- Sound pulse is generated by the emitter, reflected by object and sensed by receiver.
- Precision influenced by angle to object.
- Range: 12 cm-5 m.
- Resolution 2cm
- Accuracy 98%
- Inexpensive
- Useful for distance measurement (and can handle transparent surfaces).

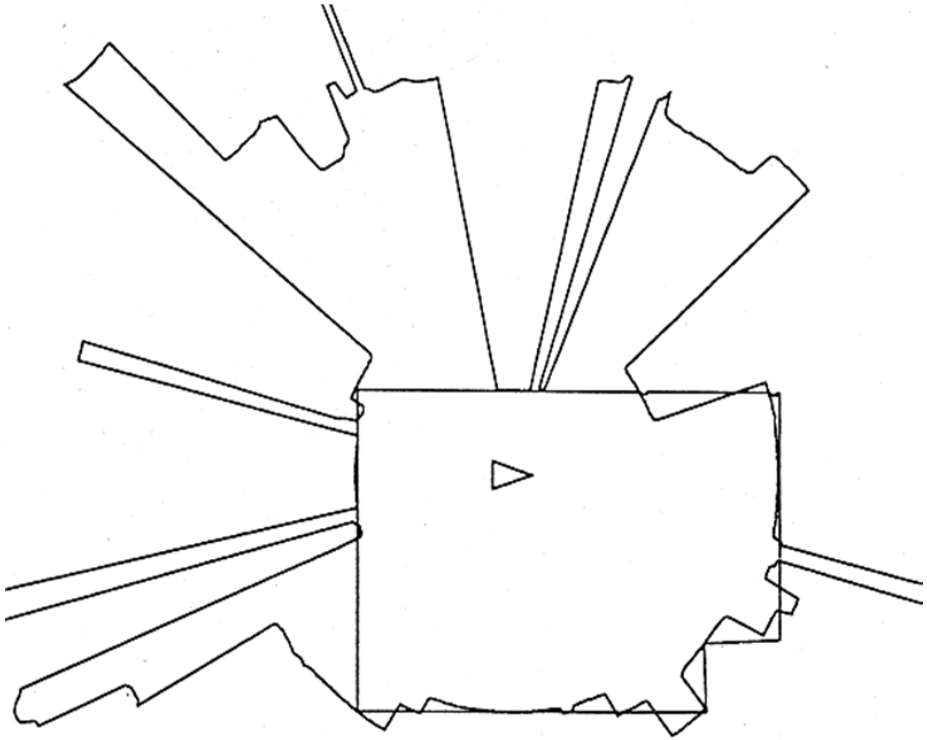
Ultrasonic Sensor

$$d = \frac{c \cdot t}{2}$$

$$c = \sqrt{\gamma \cdot R \cdot T}$$

- γ is the adiabatic index; R is the gas constant; T is temperature in Kelvin
- At 20^o $c \sim 343 m/s$
- Typical frequency 40-180kHz (lower frequency yields higher range)

Ultrasonic Sensor



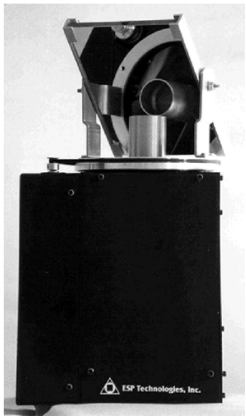
- Sound beam propagates in a cone
- We therefore identify regions of constant depth
- Soft suffices can absorb sound
- Surfaces not perpendicular to the direction of sound lead to specular reflection

Ultrasonic Sensors

- Bandwidth
  - Measuring the distance to an object 3m away takes 20ms. Operating speed is therefore 50Hz
  - If the robot has a ring of 20 sensors, firing sequentially to minimise interference, cycle time is 0.4 seconds — 2.5Hz.
- Update rate has a measurable impact on maximum speed while sensing and avoiding obstacles.

Laser rangefinder

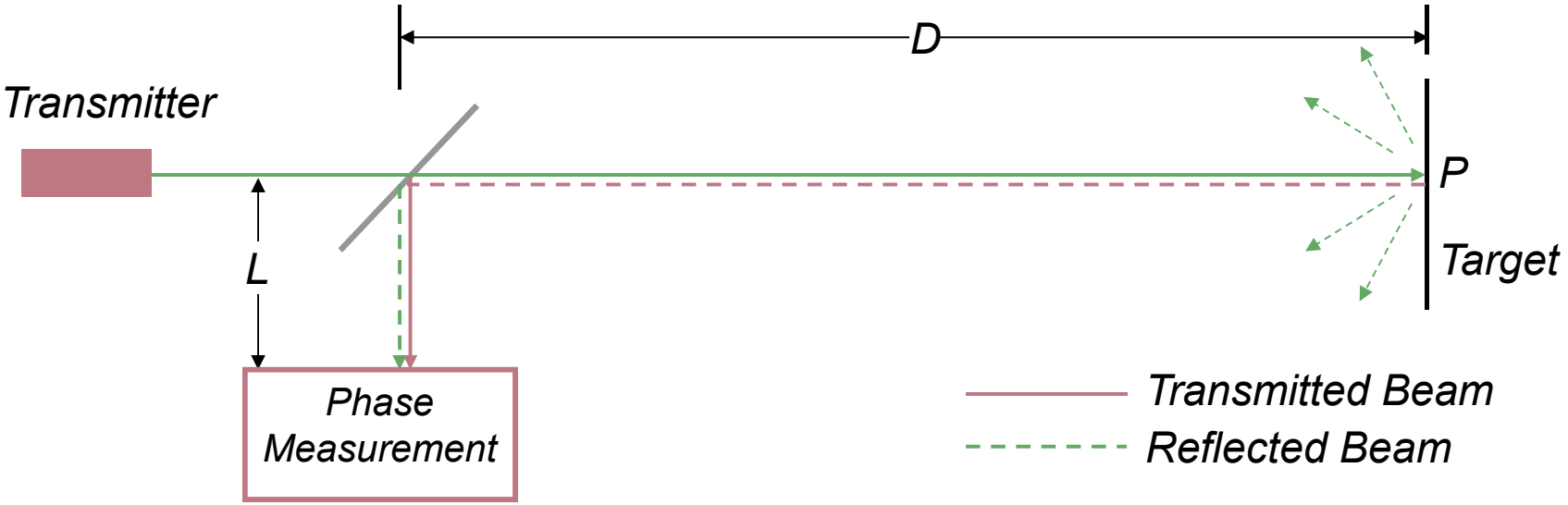
SICK



Alaska-IBEO

Hokuyo

Laser rangefinder

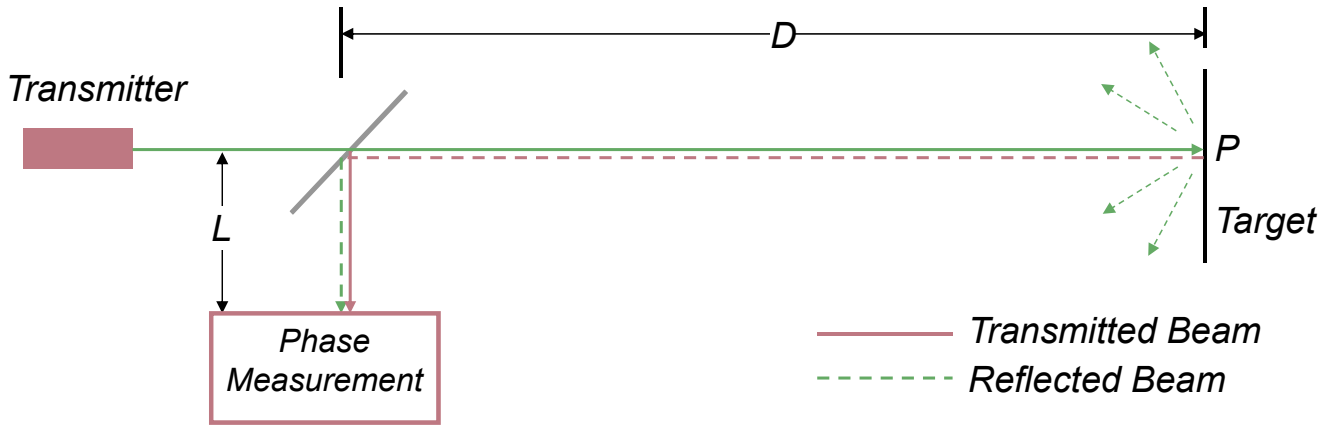


- Receiver detects round-trip time.
- Mechanical component sweeps a mirror around scene to obtain 2 or 3D measurements.

LIDAR: Principle of operation

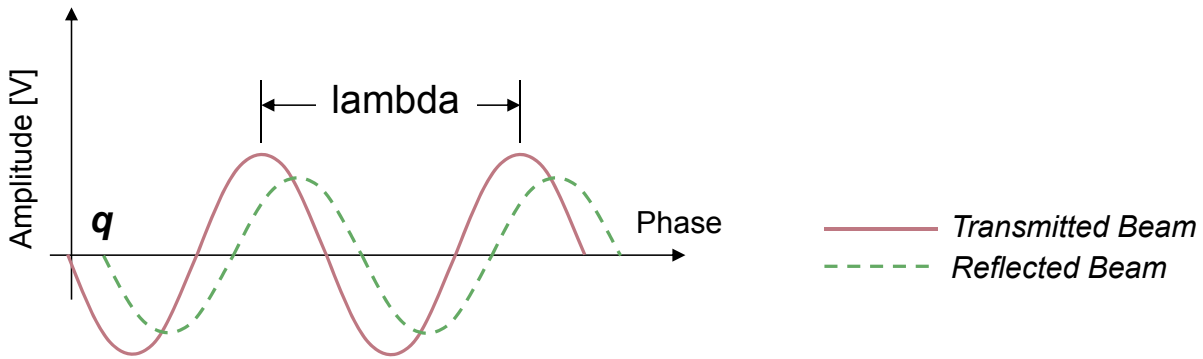
- Pulsed laser
  - measures elapsed time directly
  - resolves time in picoseconds
- Phase shift measurement to obtain range estimation

Phase shift measurement



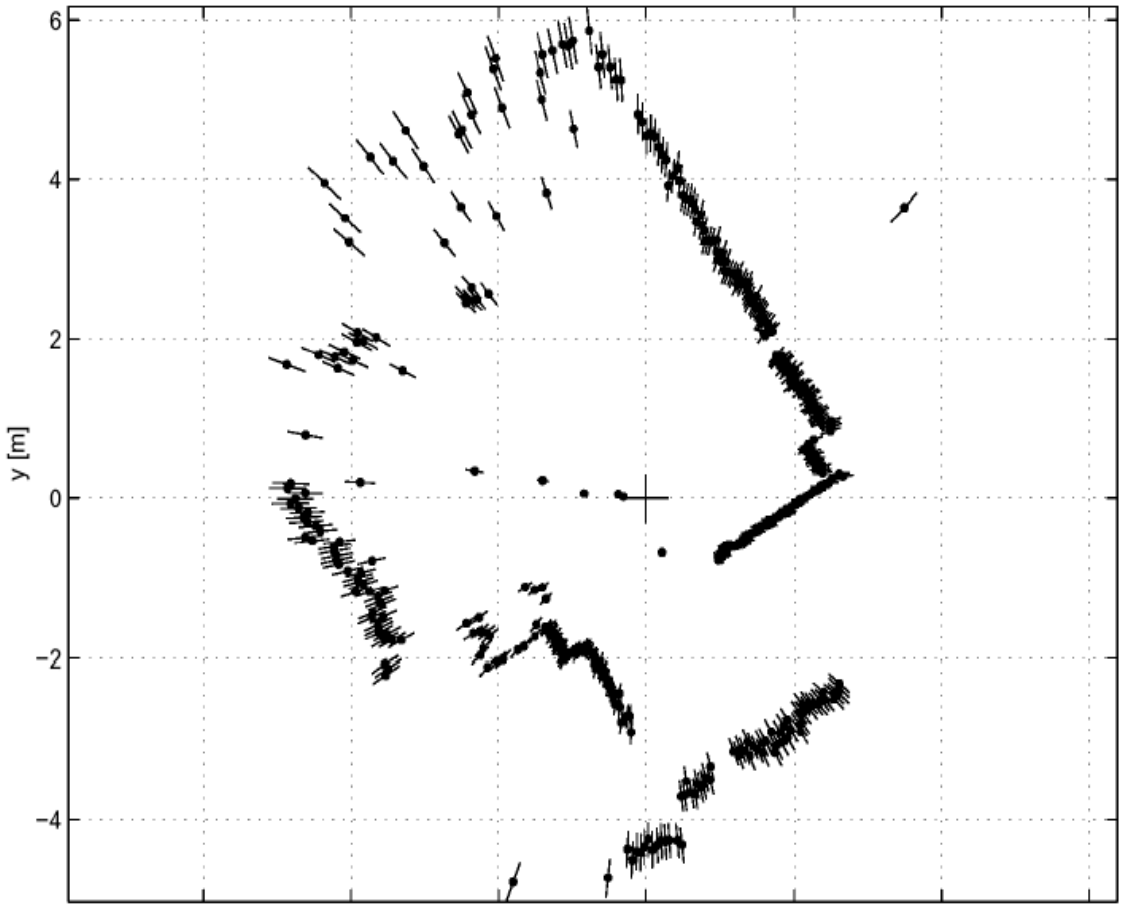
$$D' = L + 2D = L + \frac{\theta\lambda}{2\pi} \qquad \lambda = c/f$$

- c is the speed of light; f the light frequency; D' is the distance covered by the emitted light; θ is the phase difference
- Potential for ambiguity: if $f = 5MHz$, $\lambda = 60m$ then θ at 5m=65m



Errors

- Uncertainty of range is inversely proportional to the square of the received signal amplitude.
- Dark distance objects produce poor estimates compared to bright close objects.



Sample Rangfinders

- SICK LMS 200 (5000 euro)
  - angular resolution 0.25 degrees
  - Depth resolution 10-15mm; typical accuracy is 35mm
  - Range 5cm- 20m+ (80m max)
  - 75 180-degree scans per second
- Velodyne HDL-64E uses 64 laser emitters (50 000 euro)
  - Turn rate of up to 15 Hz
  - Field of view 360 azimuth, 26.8 elevation
  - Angular resolution 0.09 and 0.4 degrees respectively
  - Over 1.3 million data points per second
  - accuracy better than cm, depth up to 50m

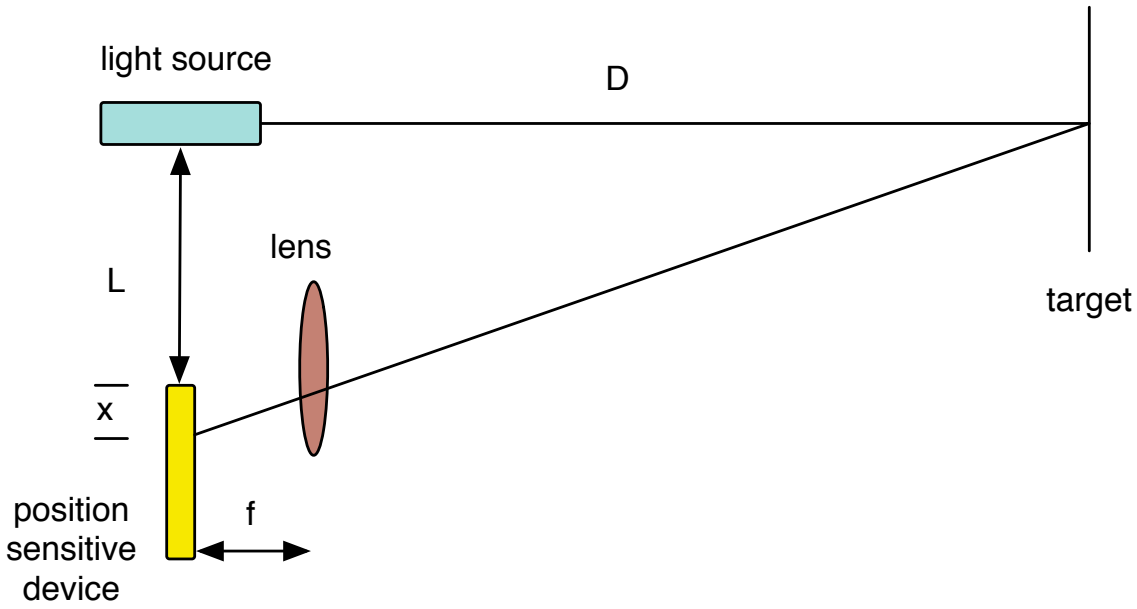
Time of flight camera

- Works similar to lidar, but captures all of scene at same time with no moving parts.
- Uses an infrared lighting source to determine distance of each pixel on sensor
- Essentially a big flash timing how long light takes to get back.
- High data rate (100Hz)
- Compact
- High non-uniform noise
- Low resolution (200x200 pixels max)

Triangulation Ranging

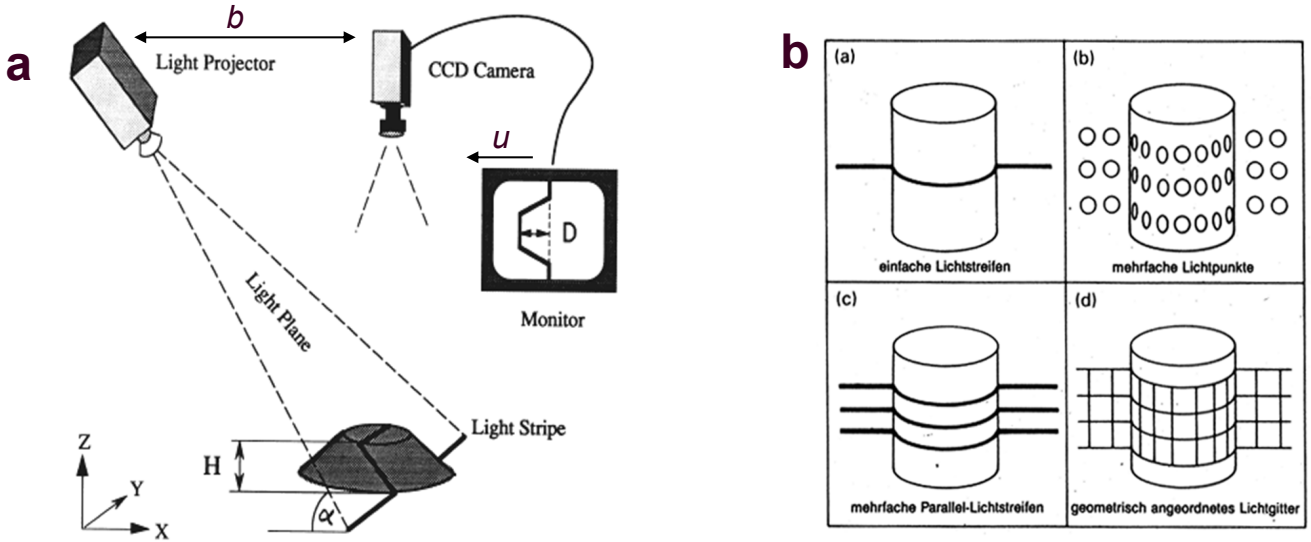
- Uses geometrical properties of the image to establish distance measurement.
- Well defined pattern is projected onto the environment.
- Reflected light is captured by a line or matrix (i.e. camera) sensor device
- Triangulation allows distance to be computed
- If size of object is known, triangulation is possible without light projection

1D triangulation



- Using similar triangles we can compute distance as a function of x (after calibration)
- $D = \frac{fL}{x}$

2D or 3D triangulation



- We can extend the basic idea to multiple dimensions by projecting structured light onto the scene.
- Range to an illuminated point can be determined from geometry.

Accuracy

- The smaller the baseline L, the more compact the sensor can be.
- The larger L, the better range resolution is.
- Risk for large L that the illuminated point is not visible.
- Larger focal length f can provide enlarged field of view or improved range resolution.
- Large focal length means large sensor head.

Where are we

- We have examined the properties of many common sensors.
- Other sensors do exist, e.g. dopler speed sensors
- Trade-offs between price, accuracy, size etc must be taken into account.
- Choose the right sensor for the task!

Videos

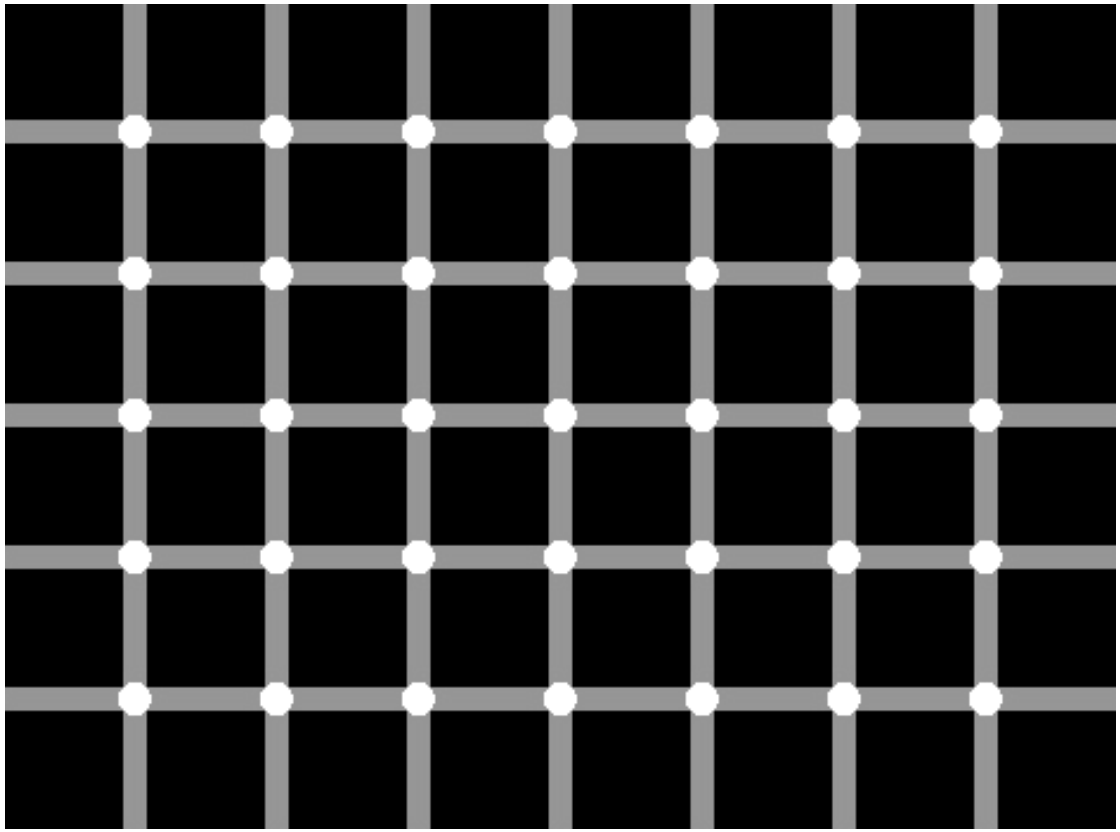
- http://www.youtube.com/watch?v=3CR5y8qZf0Y
- http://www.youtube.com/watch?v=v9oeOYMRvuQ

Computer Vision, Introduction

- The human visual system processes and interprets huge amounts of data ($\sim$ 3GB/s)
- Humans can interpret images successfully under a wide range of conditions

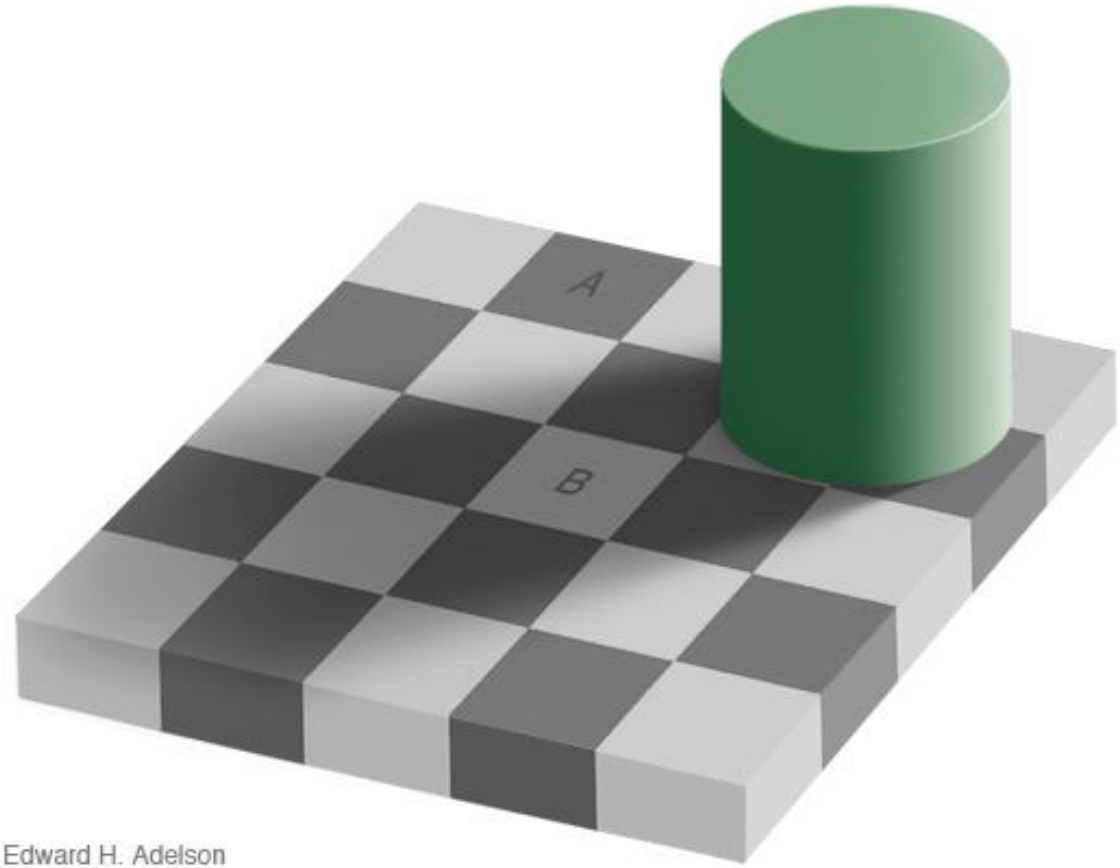
Humans

- But we don't always get it right. How many black dots are there?



Humans

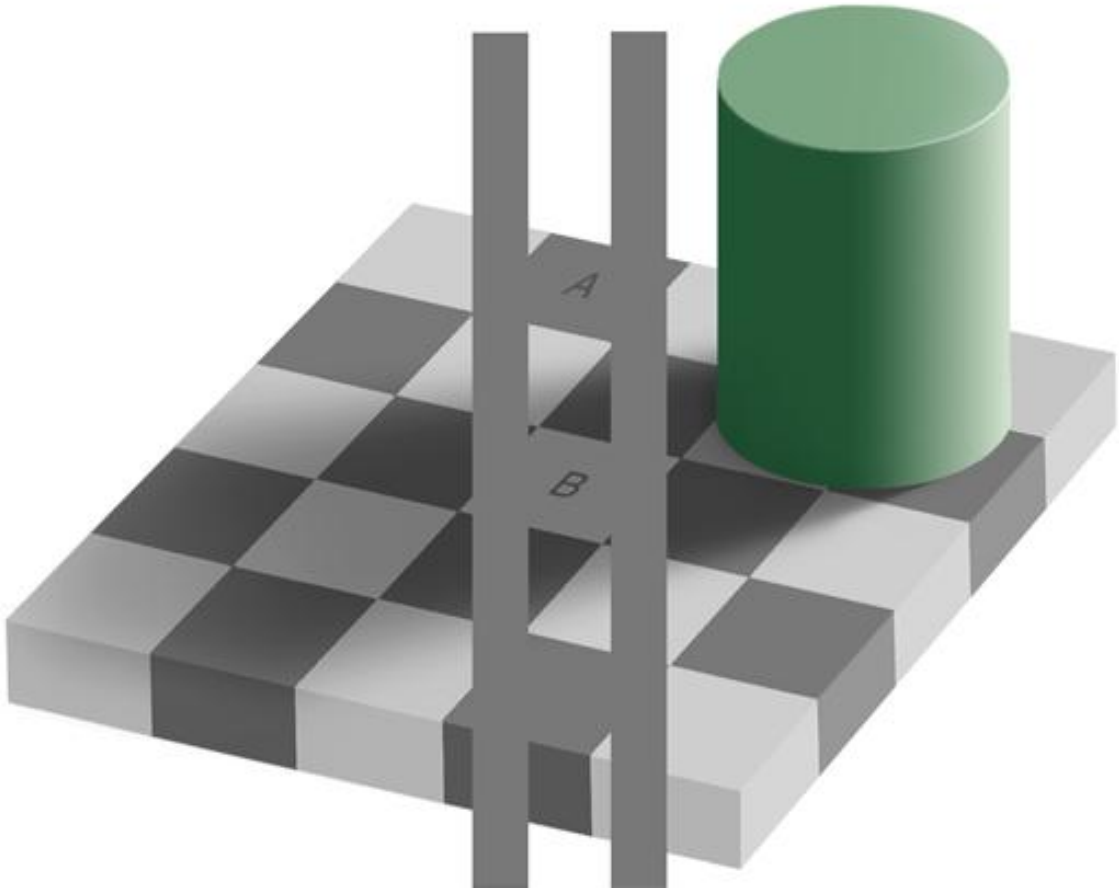
- But we don't always get it right. Which is darker, A or B?



Edward H. Adelson

Humans

- But we don't always get it right. Which is darker, A or B?



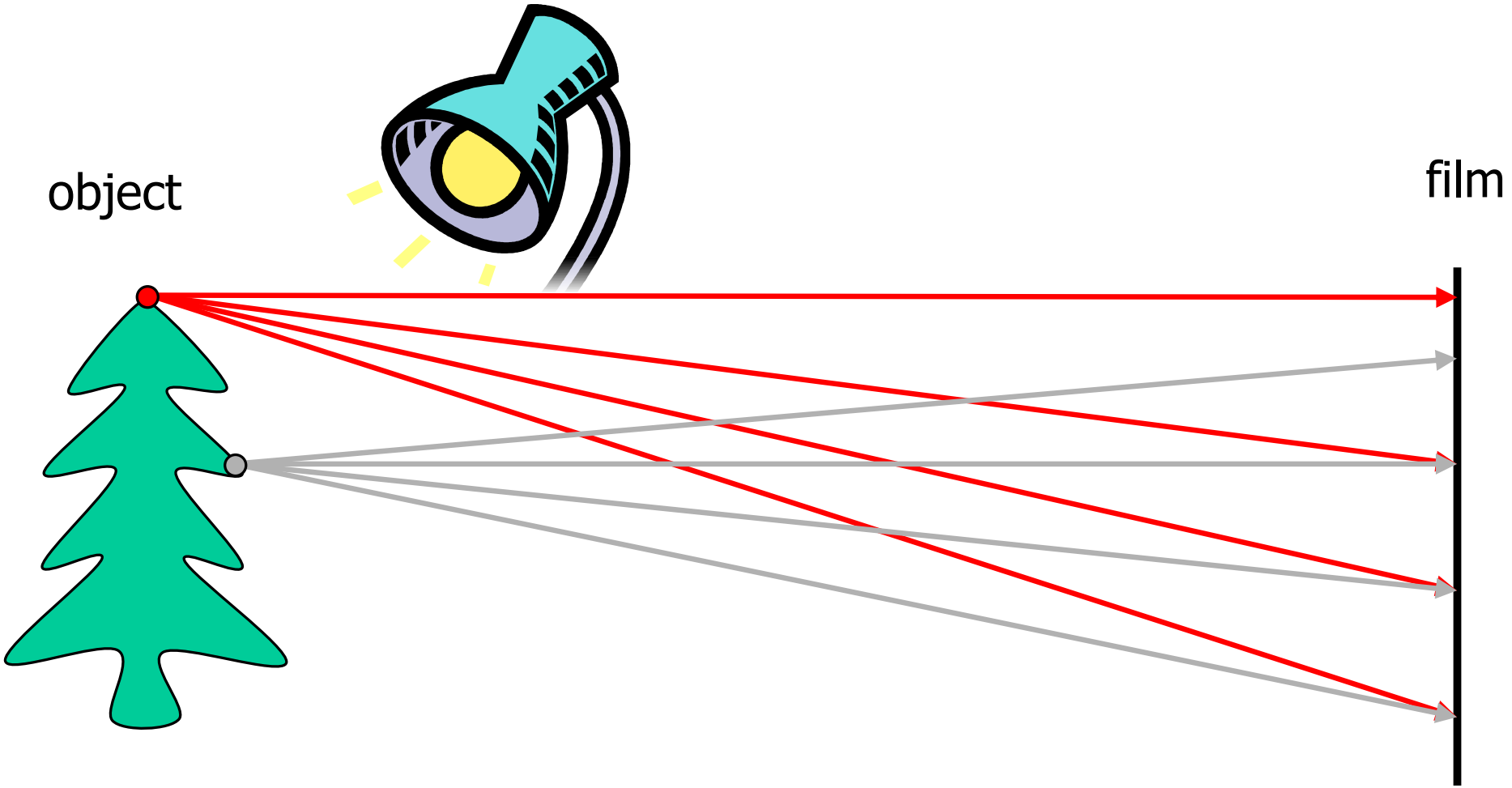
Robot Vision

- We can't hope to copy biology of human vision (60 billion neurons to process). But learn from it.
- Cycle: capture light $\rightarrow$ convert to image $\rightarrow$ process to get relevant information.
- Popular as its compact, low cost, powerful.

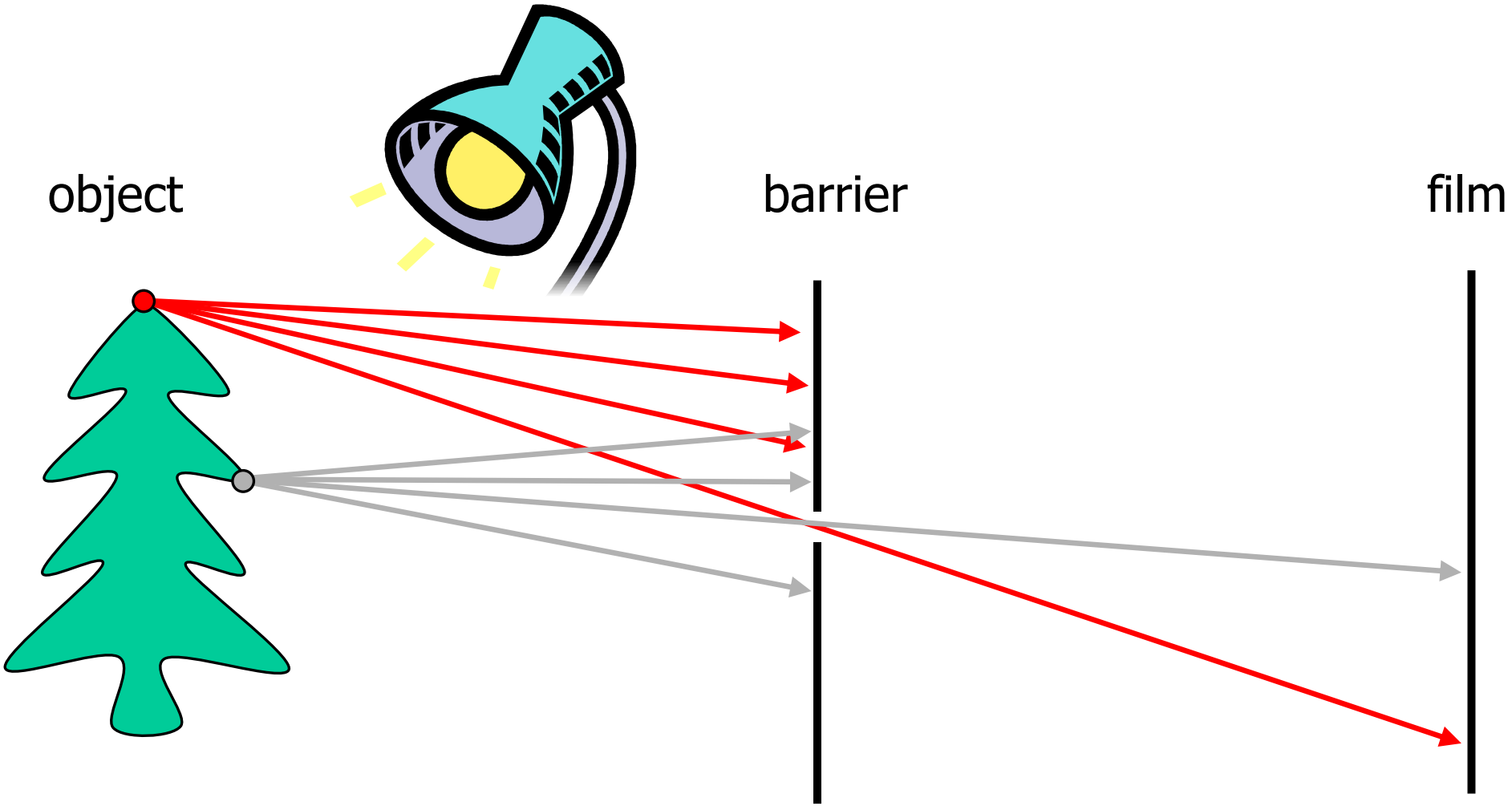
We will cover

- Pinhole camera model
- Perspective projection
- Stereo vision
- Optical flow
- Color tracking

Pinhole Camera

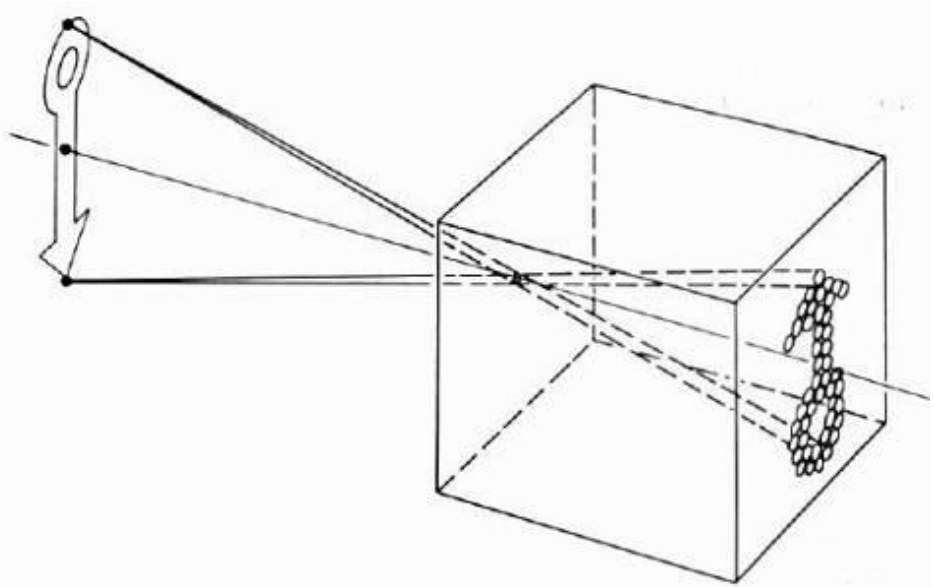


Pinhole Camera



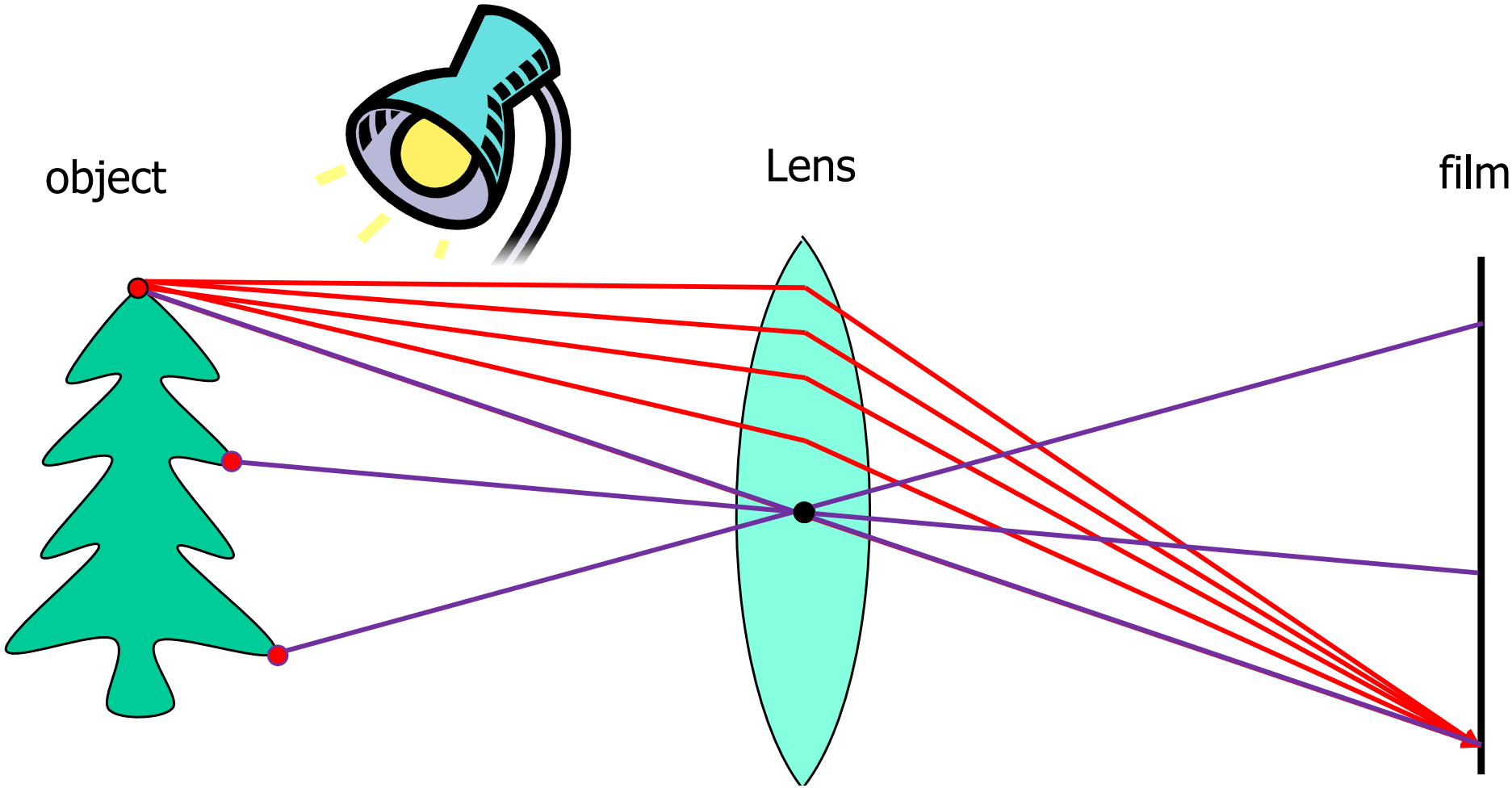
Pinhole Camera Model

- Captures beam of rays — all rays through a single point.
- Point is called centre of projection or optical centre
- Image is formed on the image plane
- The hole through which light passes is the aperture
- Small aperture means less light gets through (longer exposure)
- Large aperture means undesirable angles of light get to same point in plane (blur)

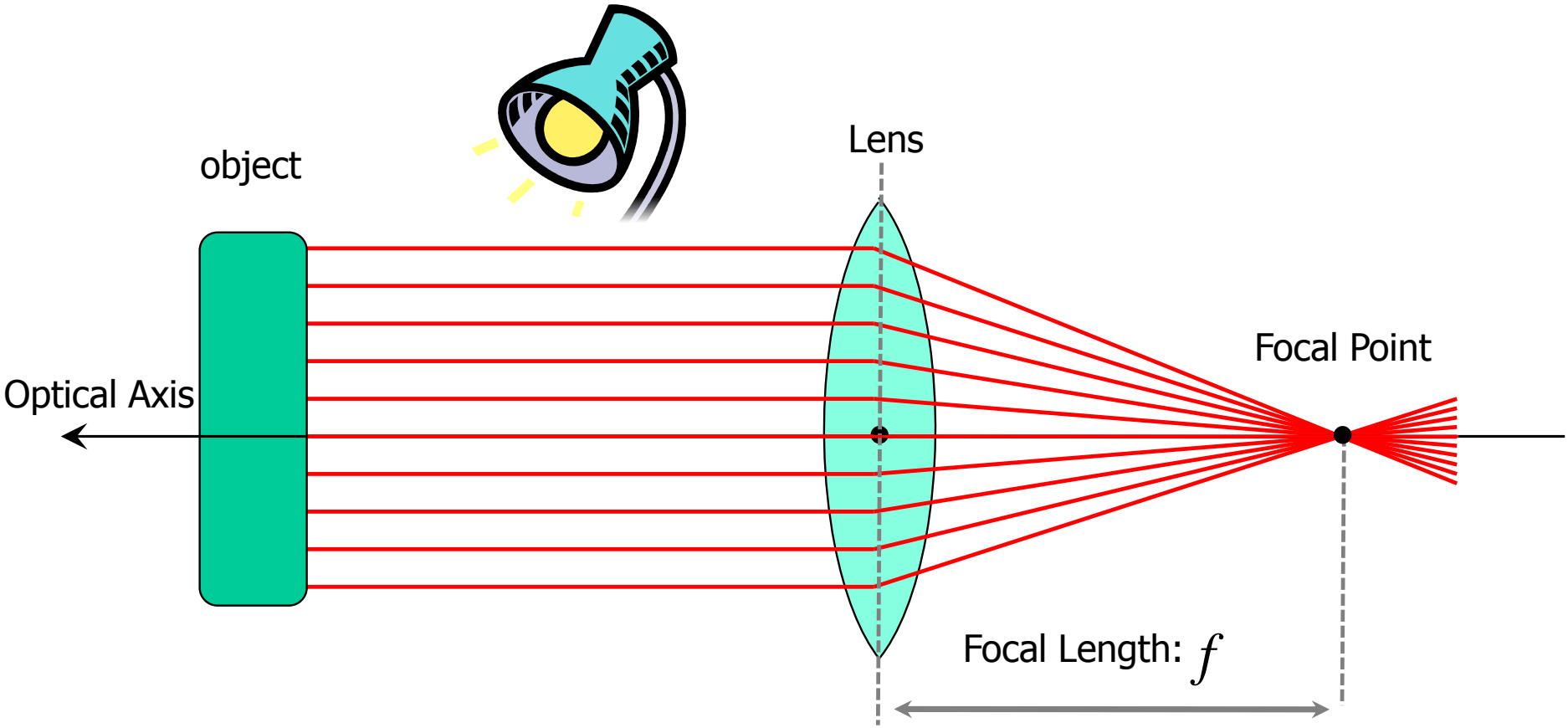


Lenses

- In an ideal pinhole, only one ray of light reaches each point.
- A lens can focus multiple rays coming from the same point
- Rays passing through the optical centre are not deviated

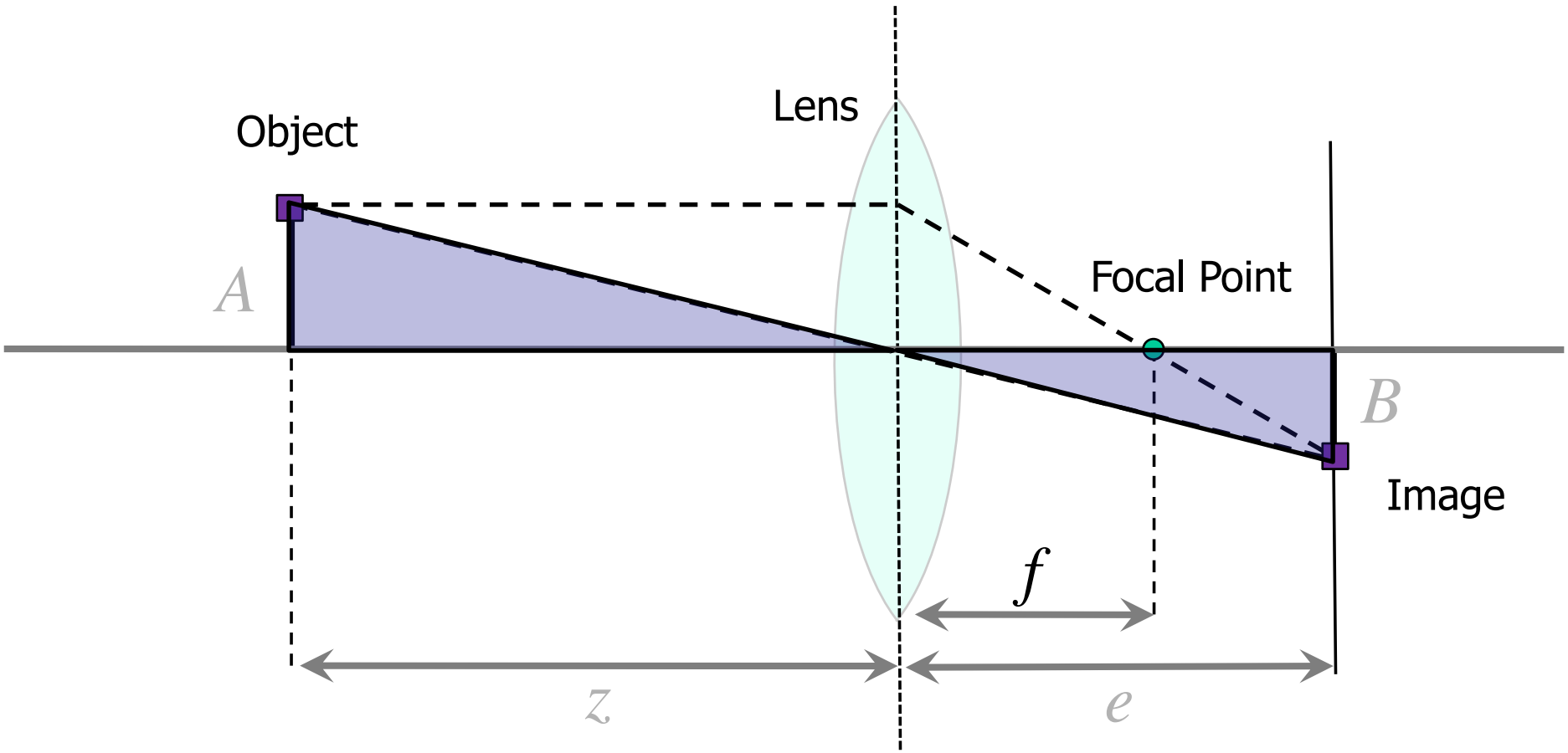


Focal Point



- All rays parallel to the optical axis converge at the focal point

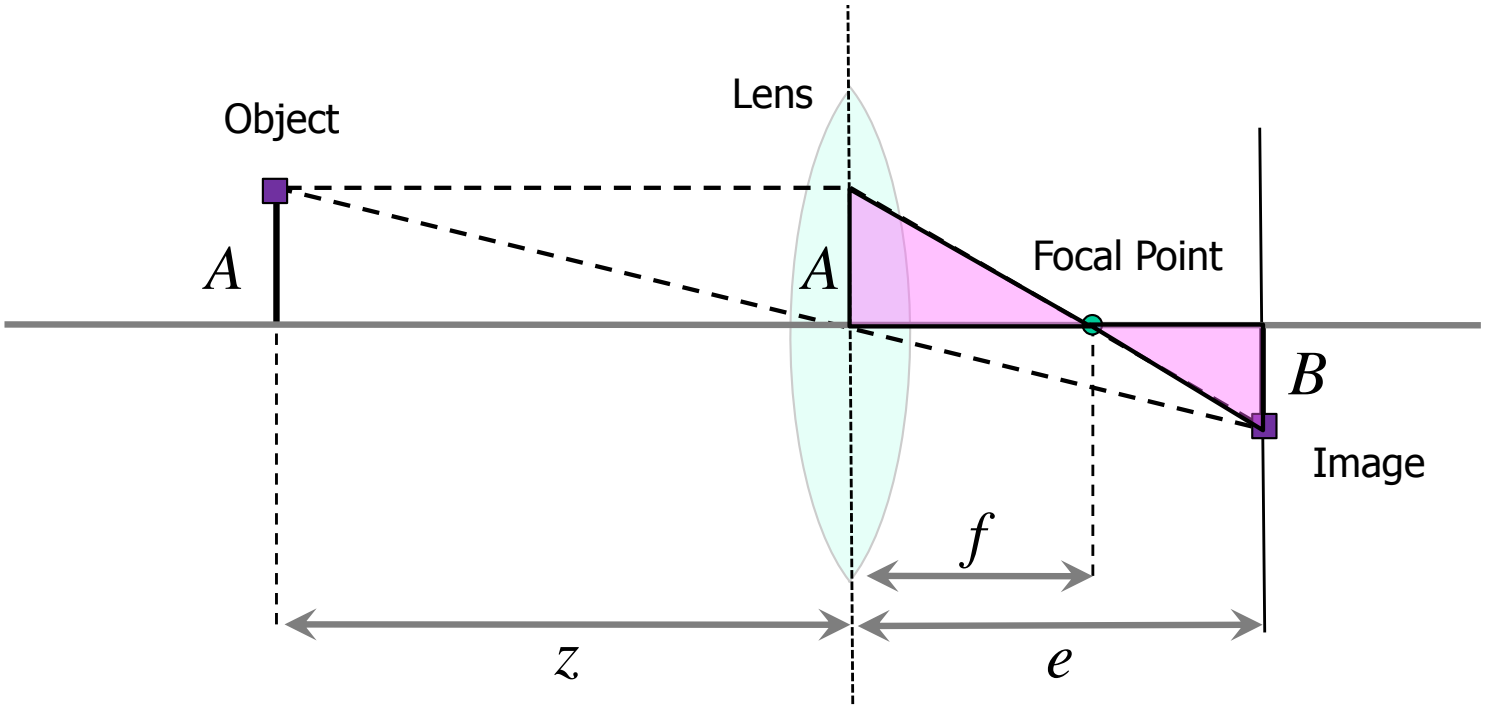
Thin lens equation



- Similar triangles

$$\frac{B}{A} = \frac{e}{z}$$

Thin lens equation

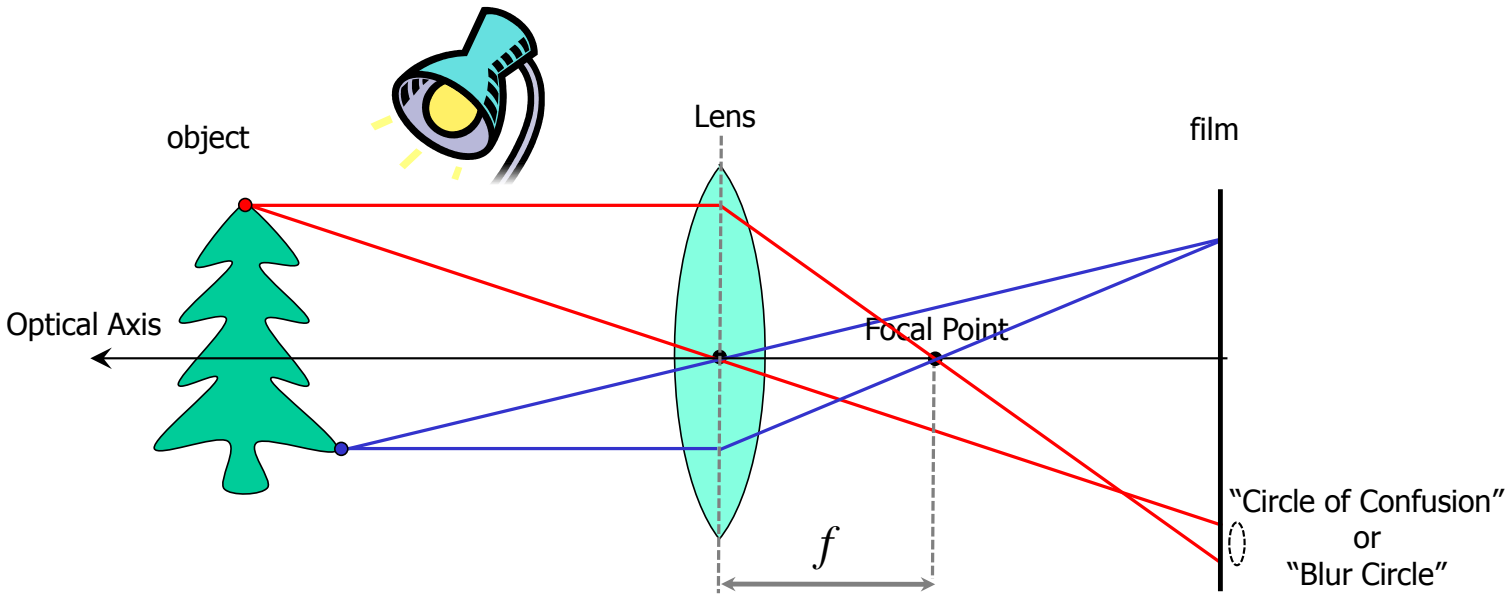


$$\frac{B}{A} = \frac{e}{z} \qquad \frac{B}{A} = \frac{e-f}{f} = \frac{e}{f} - 1$$

$$\frac{e}{f} - 1 = \frac{e}{z} \Rightarrow \frac{1}{f} = \frac{1}{z} + \frac{1}{e}$$

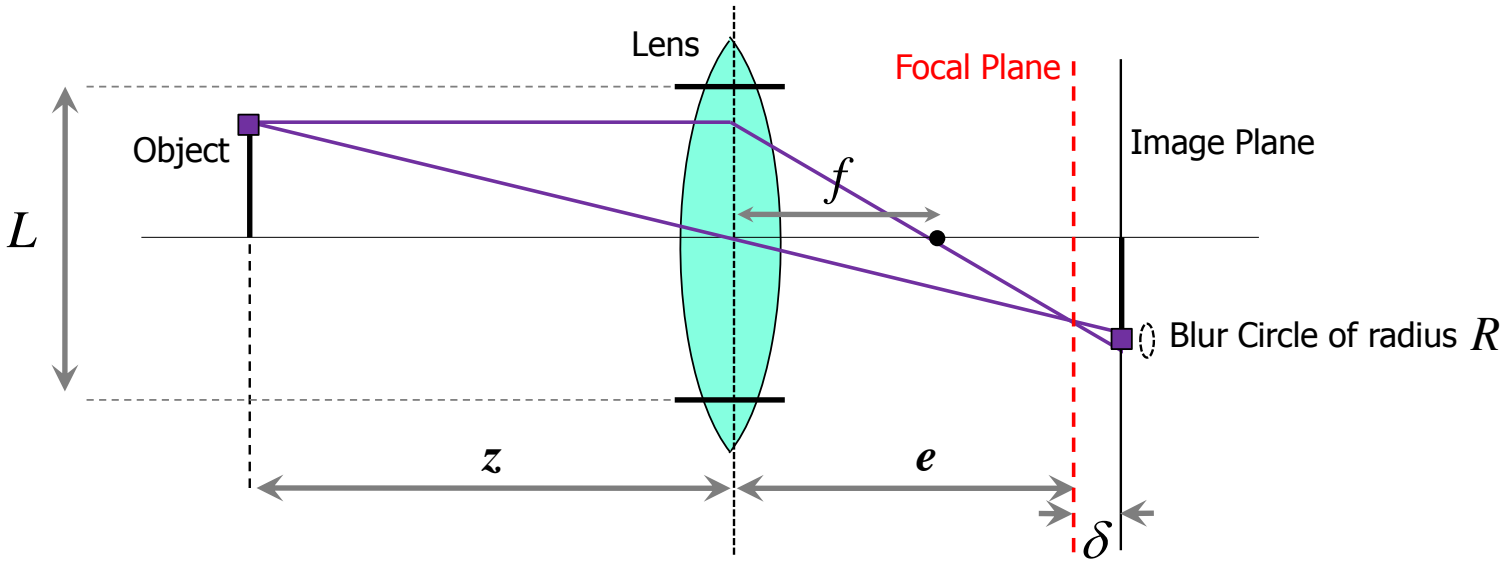
- Any object point satisfying this equation is in focus.
- Can be used to estimate distance to object.

Focus



- There is a specific distance at which objects are in focus.
- Other points project to a blur circle

Focus

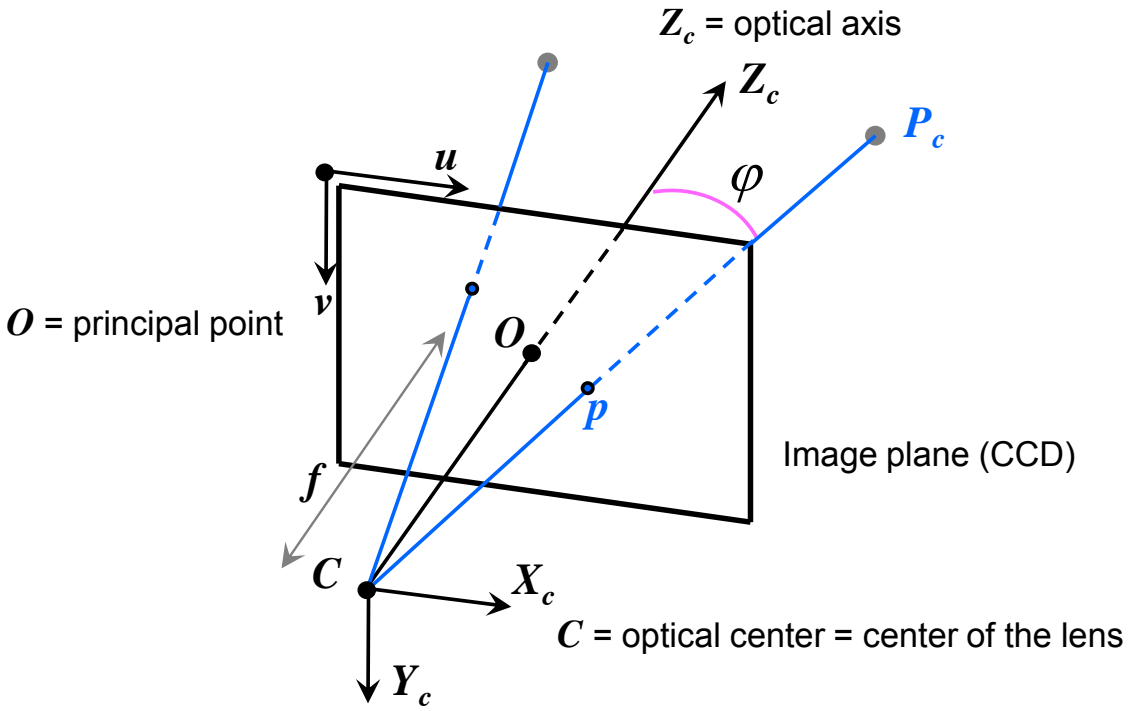


- Blur circle radius $R = \frac{L\delta}{2e}$
- Shrinking L shrinks R; larger aperture gives worse blur.
- If R is smaller than image resolution, then no blur will be seen.
- Sensitivity to blur increases when objects are close to the lens.

From Pinhole to Perspective

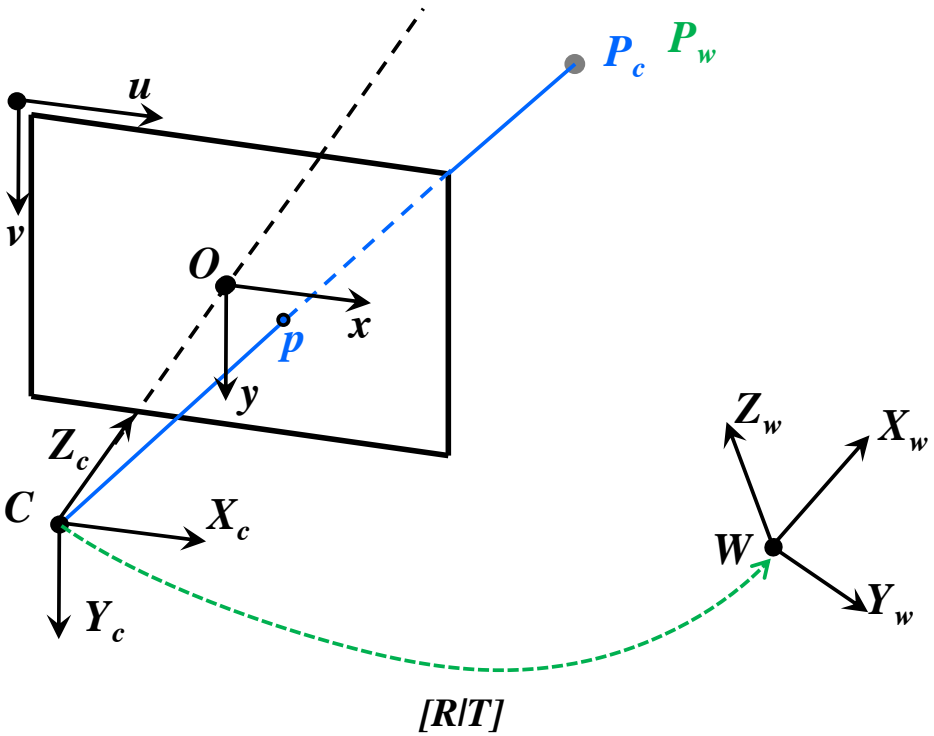
- Adjust the image plane so that objects at infinity are in focus. $z \gg f, z \gg L$
- So $1/z \sim 0$, $f \approx e$
- If f stays constant $h' = \frac{f}{z} h$
- Apparent size of object depends on distance from observer — perspective.
- Gives humans very strong depth cues, but can be misled (`http://www.youtube.com/watch?v=TtdOYjXF0no`)

Perspective Camera



- For convenience, we represent the image plane in the front so that the image is not flipped.
- Camera measures angles, not distances (all points along p, P_c appear at p)
- We want to move from world coordinates to pixel coordinates.

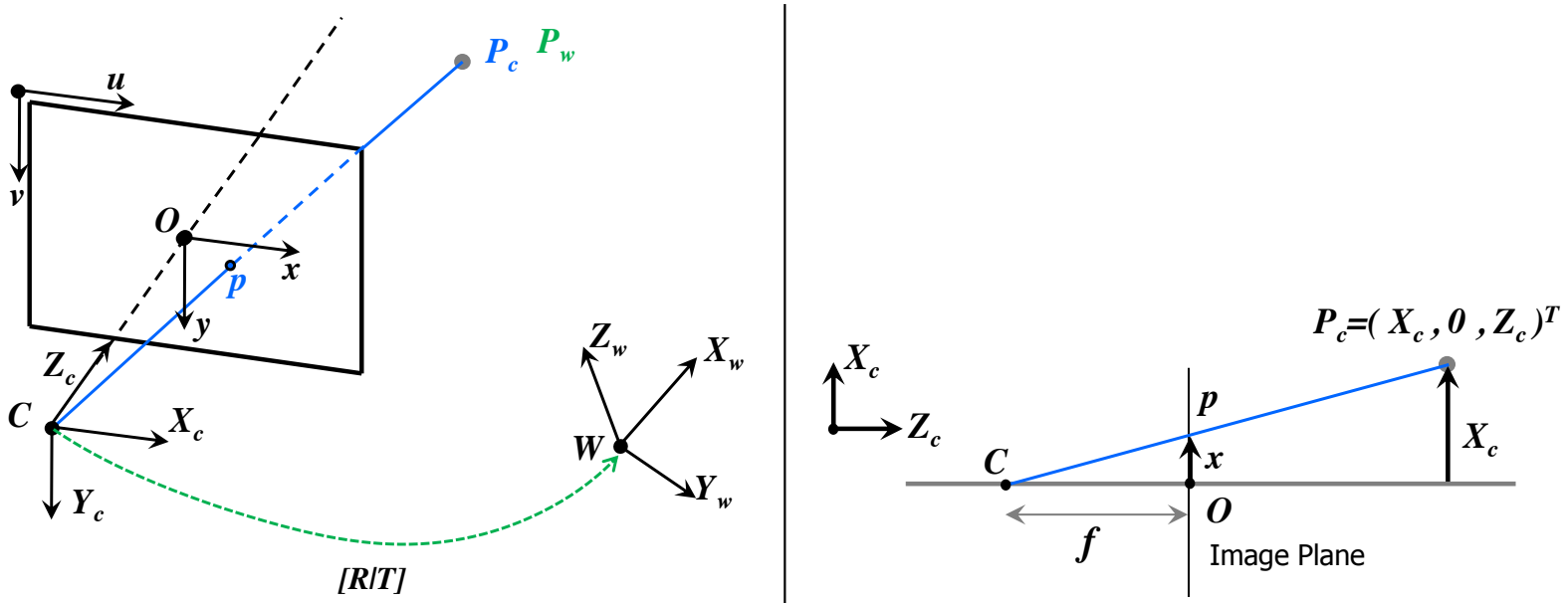
From world to pixel coordinates



- We want to find pixel coordinates (u, v) of point P_w in the world frame.

1. Convert world point P_w to camera point P_c
2. Convert P_c to image plane coordinates (x, y)
3. Convert P_c to discretised pixel coordinates (u, v)

From the camera frame to the image plane



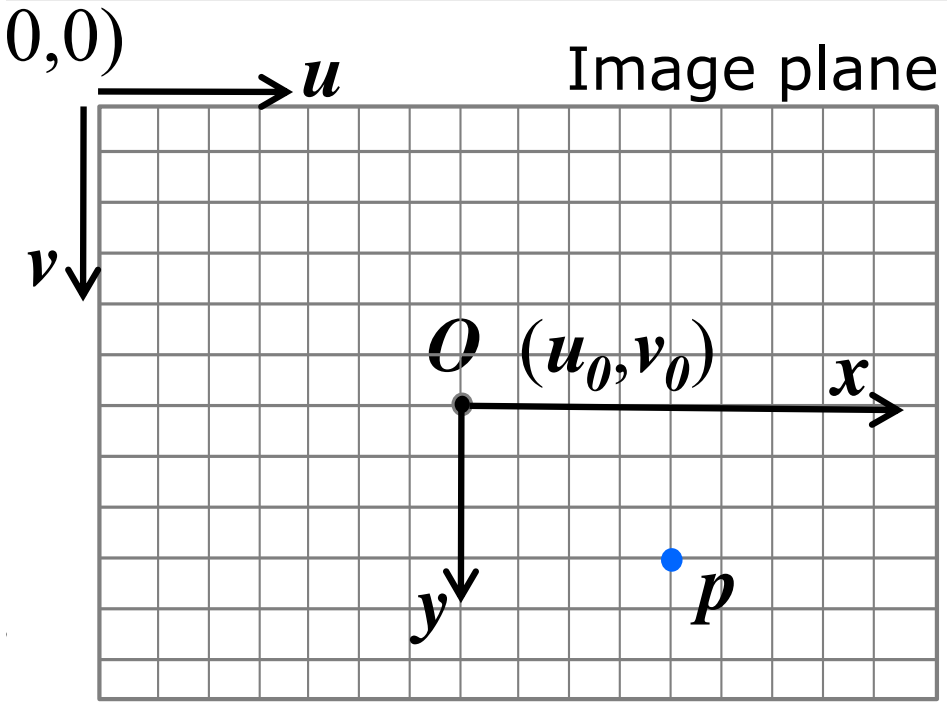
- The point in the camera's frame of reference $P_c = [X_c, 0, Z_c]^T$ projects to a point $p = (x, y)$ on the camera plane.
- Using similar triangles $\frac{x}{f} = \frac{X_c}{Z_c} \rightarrow x = \frac{fX_c}{Z_c}$
- Extending to y dimension we obtain $y = \frac{fY_c}{Z_c}$
- So we have image plane coordinates for a point in the camera frame.

From the camera frame to pixel coordinates

- We want to convert from (x, y) to (u, v)
- Need to consider
  - pixel coordinate of optical center $O = (u_0, v_0)$
  - Scale factors for pixel size in both dimensions k_u, k_v

$$u = u_0 + k_u x \rightarrow u_0 + k_u \frac{fX_c}{Z_c}$$
$$v = v_0 + k_v y \rightarrow v_0 + k_v \frac{fY_c}{Z_c}$$

- Trick: use homogeneous coordinates to obtain a linear mapping from 3D to 2D by introducing an extra scale element λ



$$p = \begin{bmatrix} u \\ v \end{bmatrix} \Rightarrow \tilde{p} = \begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix}$$

Normally, $\lambda = 1$

From the camera frame to pixel coordinates

$$u = u_0 + k_u \frac{fX_c}{Z_c}$$
$$v = v_0 + k_v \frac{fY_c}{Z_c}$$

- Expressed in matrix form with homogeneous coordinates

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

- We can set $\alpha_u = k_u f$, $\alpha_v = k_v f$ to represent focal length in the u and v directions respectively.
- The 3×3 matrix is the matrix of intrinsic parameters.
- If the u and v axes are not aligned, we can introduce an additional $\alpha_u cot\theta$ term.

From the World Frame to the Camera Frame

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

In homogeneous coordinates, we have a extrinsic parameter matrix.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Rewriting

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

From the World Frame to the Camera Frame

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

In homogeneous coordinates, we have a extrinsic parameter matrix.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Rewriting

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|T] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

From the World Frame to the Camera Frame

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

In homogeneous coordinates, we have a extrinsic parameter matrix.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Rewriting

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

From the World Frame to the Camera Frame

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

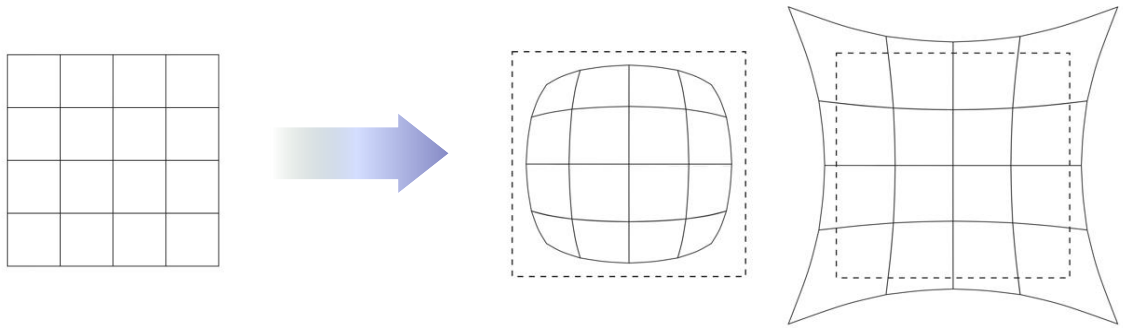
In homogeneous coordinates, we have a extrinsic parameter matrix.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Rewriting

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Distortion



- Straight lines in the world are not always straight lines in the image.
- We want to map from ideal/undistorted (u, v) to observed (distorted) coordinates (u_d, v_d)
- Simple distortion model:

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = (1 + k_1 r^2) \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$

$$r^2 = (u - u_0)^2 + (v - v_0)^2$$

- k_1, the radial distortion parameter, is another intrinsic parameter.
- More complex distortion models exist and are useful for very wide angle lenses.

Distortion

- Straight lines in the world are not always straight lines in the image.
- We want to map from ideal/undistorted (u, v) to observed (distorted) coordinates (u_d, v_d)
- Simple distortion model:

$$\begin{bmatrix} u_d \\ v_d \end{bmatrix} = (1 + k_1 r^2) \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$

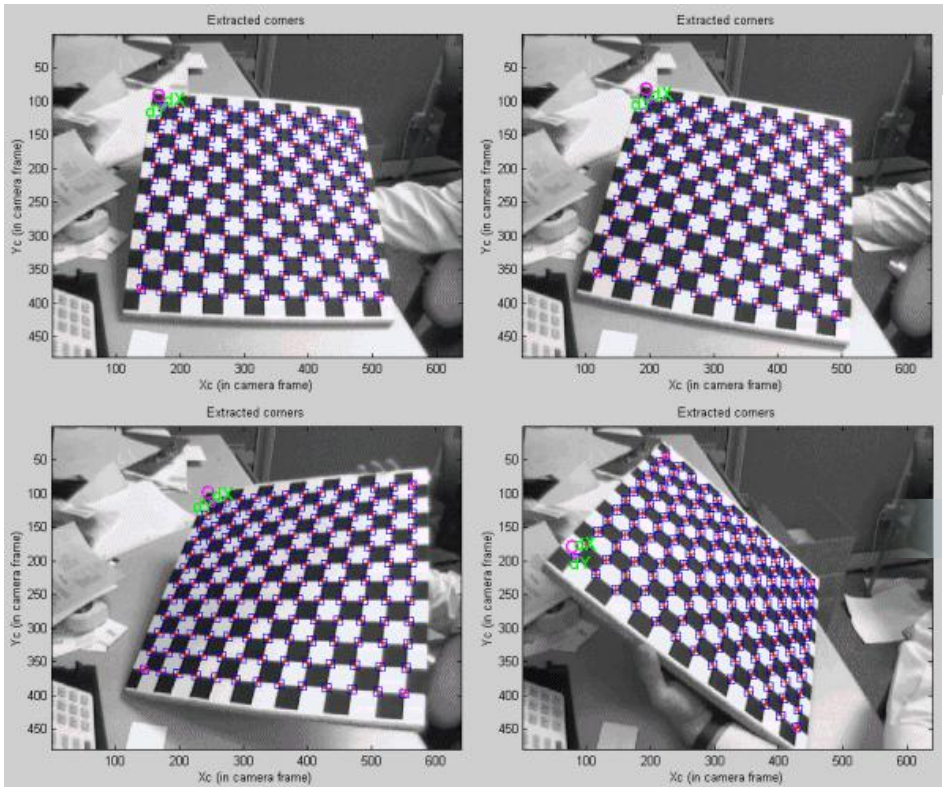
$$r^2 = (u - u_0)^2 + (v - v_0)^2$$

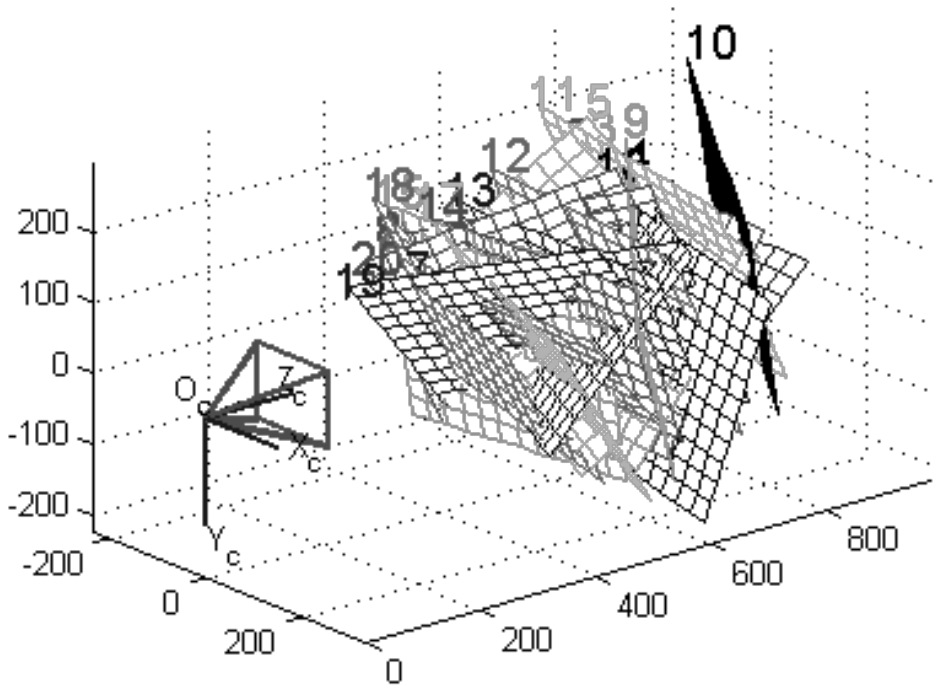
- k_1, the radial distortion parameter, is another intrinsic parameter.
- More complex distortion models exist and are useful for very wide angle lenses.

Camera Calibration

- Given an arbitrary point in space, we want to be able to know where it will appear in an image; and vice-versa.
- This will allow us to accurately identify the position of points in the world.
- To do so, we need to compute the unknown parameters of K, R and T.
- Basic approach: utilize known correspondences between p and P
- Problem: p is (u, v), P is $(X_w, Y_w, Z_w)^T$ — 2-D vs 3-D.
- Solution: use perspective.

Camera Calibration





Camera Calibration

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R|T] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

- 11 values to estimate (since scale doesn't matter, m_{34} can be set to 1).

$$u_i = \frac{\lambda u_i}{\lambda} = \frac{m_{11}X_i + m_{12}Y_i + m_{13}Z_i + m_{14}}{m_{31} + m_{32} + m_{33} + m_{34}}$$

$$v_i = \frac{\lambda v_i}{\lambda} = \frac{m_{21}X_i + m_{22}Y_i + m_{23}Z_i + m_{24}}{m_{31} + m_{32} + m_{33} + m_{34}}$$

- Each point gives us 2 constraints so to solve for 11 unknowns, we need 6 points.
- Result is the 3×4 projection matrix. We need K, R, T.
- We use QR factorisation to decompose the $m_{11} : m_{33}$ sub matrix into an upper triangular matrix K and rotation matrix R
- T can be obtained from $T = K^{-1}[m_{14}\ m_{24}\ m_{34}]^T$

Camera Calibration

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{33} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

- 11 values to estimate (since scale doesn't matter, m_{34} can be set to 1).

$$u_i = \frac{\lambda u_i}{\lambda} = \frac{m_{11}X_i + m_{12}Y_i + m_{13}Z_i + m_{14}}{m_{31} + m_{32} + m_{33} + m_{34}}$$

$$v_i = \frac{\lambda v_i}{\lambda} = \frac{m_{21}X_i + m_{22}Y_i + m_{23}Z_i + m_{24}}{m_{31} + m_{32} + m_{33} + m_{34}}$$

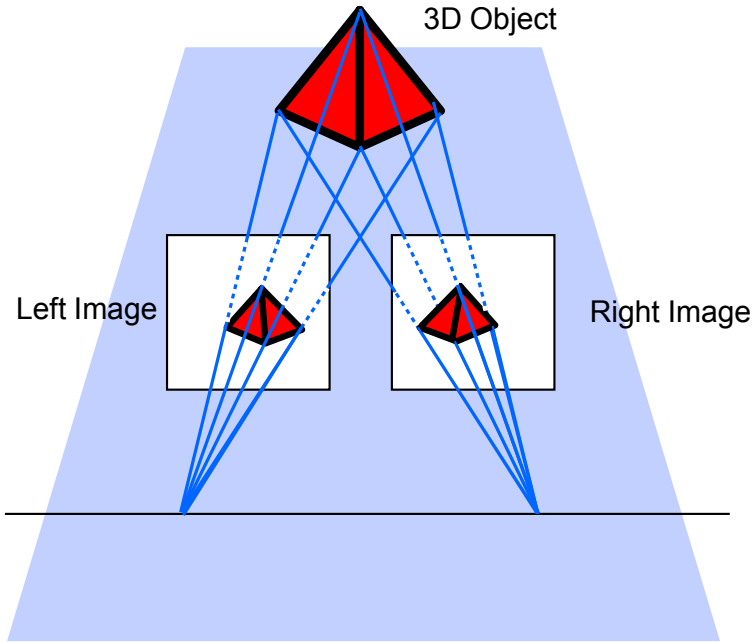
- Each point gives us 2 constraints so to solve for 11 unknowns, we need 6 points.
- Result is the 3×4 projection matrix. We need K, R, T.
- We use QR factorisation to decompose the $m_{11} : m_{33}$ sub matrix into an upper triangular matrix K and rotation matrix R
- T can be obtained from $T = K^{-1}[m_{14}\ m_{24}\ m_{34}]^T$

Camera Calibration

- Projection aids us in computing world coordinates.
- See "A Flexible New Techniuqe for Camera Calibration" by Zhang for more details (`http://research.microsoft.com/en-us/um/people/zhang/Papers/TR98-71.pdf`)
- First calibration method to use simple targets rather than known 3D objects, very easily useable.
- Widely implemented, e.g. in the OpenCV toolkit (`http://opencv.org`).
- `https://www.youtube.com/watch?v=0xBa_5HvZyI`
- `https://www.youtube.com/watch?v=91tYEgpmN4M`

Where are we?

- We can correct a distorted camera to compute which ray an image point lies on.
- We cannot capture the 3D structure of a scene from a single image though.
- We can however observe a scene from two different view points, solve for the intersection of the rays, and recover 3D structure.



Measuring Distance

- Structure from stereo (Stereo-vision): use two cameras with known relative position and orientation.

- Structure from motion: use a single moving camera: 3D structure and camera motion can be estimated up to some scale (if distance between points is unknown).

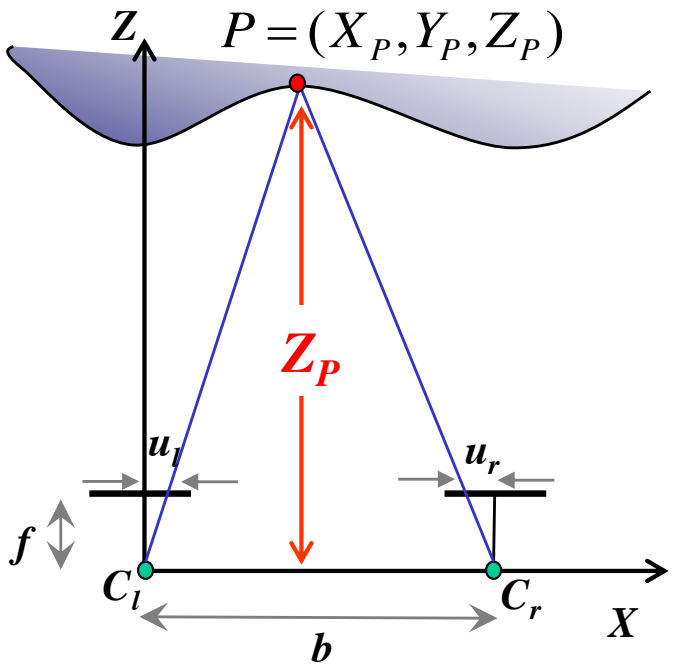
Simple Stereo Vision

- Assume both cameras are identical and aligned on a horizontal axis, separated by a baseline b.
- Convention states that the left camera is at the origin.
- From similar triangles we have

$$\frac{f}{Z_p} = \frac{u_l}{X_p} \qquad \frac{f}{Z_p} = \frac{u_r}{b - X_p}$$

$$Z_p = \frac{bf}{u_l - u_r}$$

- $u_l - u_r$ is referred to as the disparity — difference in image location of the projection of a 3D point in two image planes.



Disparity

$$Z_p = \frac{bf}{u_l - u_r}$$

- Distance is inversely proportional to disparity — distance to near objects can be measured more accurately.
- Disparity is proportional to b — increasing b improves accuracy
- Increased b may cause objects to appear in one camera and not the other.
- If b is unknown, Z_p can be computed up to some scale.
- A point visible to both cameras produces a pair of image points called a conjugate pair or correspondence pair.
- Conjugate pairs lie along an epipolar line

Disparity Maps

- A disparity map holds the disparity value at every pixel:
  - For each pixel, finds the correspondent points in the original image.
  - Compute the disparity between these points.
  - Increased disparity $\rightarrow$ brighter colour.
- Close objects have higher disparity so appear brighter.
- Useful for collision avoidance.

Left image

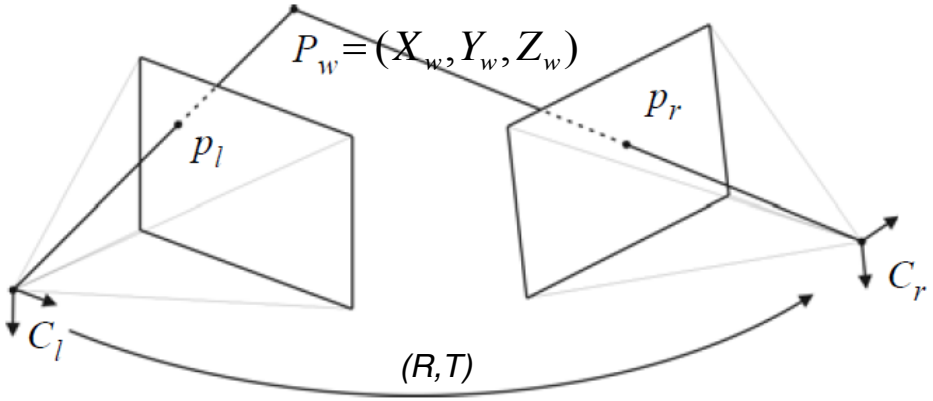
Right image

Disparity map

Stereo Vision - The General Case

- Cameras are never perfectly aligned or identical.
- To utilise a stereo camera we need to determine
  - The relative pose (i.e. rotation, translation) between the cameras (extrinsic parameters)
  - The focal length, optical centre and radial distortion of each camera (intrinsic parameters)
- We can use the calibration methods discussed before to compute these parameters.

Stereo Vision - The General Case



- For the left camera, $R_l = I$, $T = [0, 0, 0]^T$, so

$$\lambda_l \begin{bmatrix} u_l & v_l & 1 \end{bmatrix}^T = K_l \begin{bmatrix} X_w & Y_w & Z_w \end{bmatrix}^T$$

- For the right camera

$$\lambda_r \begin{bmatrix} u_r & v_r & 1 \end{bmatrix}^T = K_r R \begin{bmatrix} X_w & Y_w & Z_w^T \end{bmatrix} + T$$

- Unknowns: $\lambda_l, \lambda_r, X_w, Y_w, Z_w$
- We have 6 equations — problem is overdetermined, can be solved using least squares or by computing 3d point that minimises the distances between rays passing through p_l and p_r.

The Correspondence Problem

- To use the equations, we must identify conjugate pairs p_l and p_r in the left and right camera images, originating at the same scene point P_w.
- The correspondence problem:

  > *Given two images of the same scene from different perspectives, how can we identify the same object points in both images?*

- Searching for correspondence requires us to look for small patches of identical pixels in the other image.
- Problems:
  - Occlusion: points exist in one image which do not have a correspondent in the other.
  - Photometric distortion: Due to reflections, some points may be brighter in one camera than the other.
  - Projective Distortion: As cameras move further apart, identical objects look increasingly different.

Correspondence Constraints

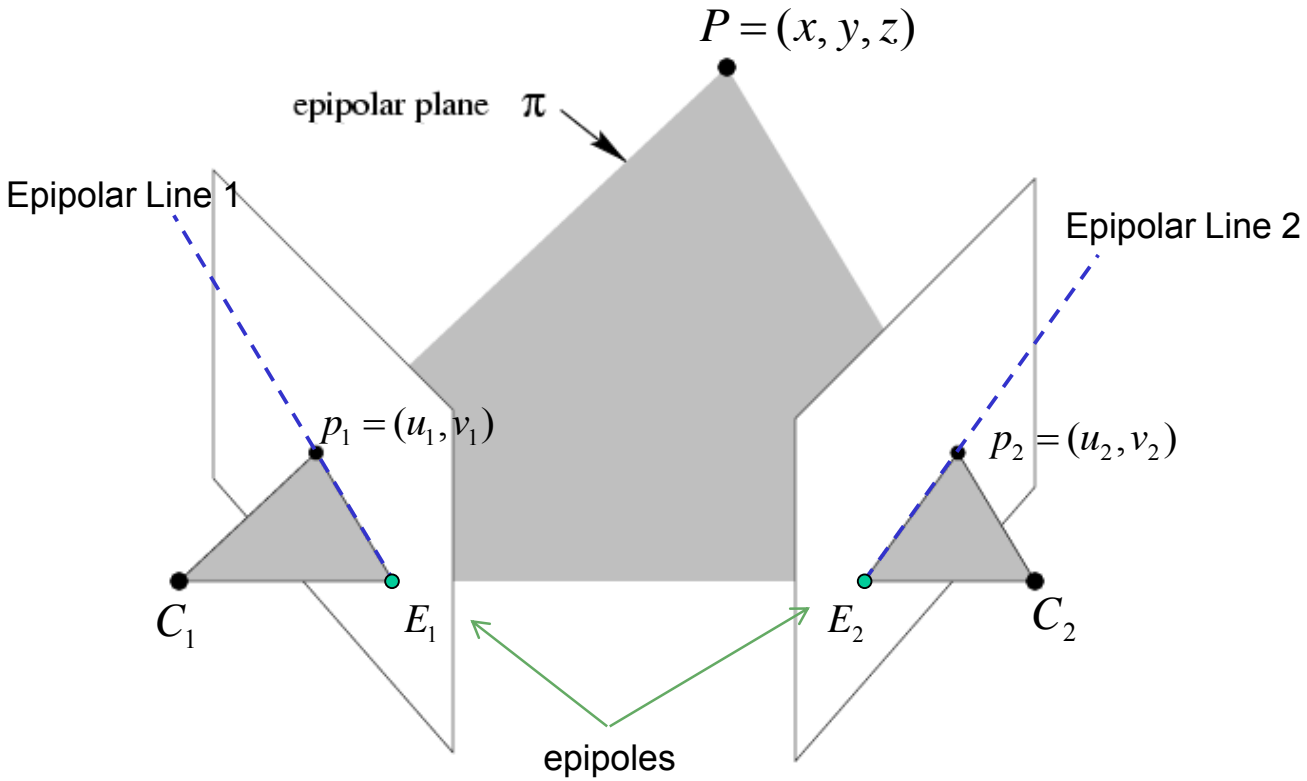
- Several constraints can be exploited to improve correspondence search:
  - Similarity: a feature in the image will appear "similar" in the other image.
  - Continuity: far from image borders, the depth of scene points on a continuous surface varies continuously, limiting the disparity gradient
  - Unicity: A point in one image can be pared with only one point in the other image.
  - Monotonic order: For opaque objects, if a point p_l in the left image corresponds with a point p_r, then a point to the left (right) of p_l can only be found to the left (right) of p_r
  - Epipolar constraint (discussed later)

Correspondence Search

- Two categories of methods exist for correspondence search.
  - Area based methods consider a small patch in one image, and look for the most similar patch in the second image using some correlation measure.
    - Search done for every pixel.
    - Performs poorly in uniform regions
    - Sum of absolute differences (SAD), sum of squared differences (SSD) and normalised cross correlation are the most commonly used methods.
  - Feature-based methods extract features from the image and search for these in the other image. E.g. edges; corners and line segments (though not necessarily geometric).
    - Output is sparse (only where features are detected); interpolation is necessary.
    - Much faster than area based techniques.
- Both search methods are exhaustive, scanning all lines and columns of a pixel. Can we do correspondence search in 1D?

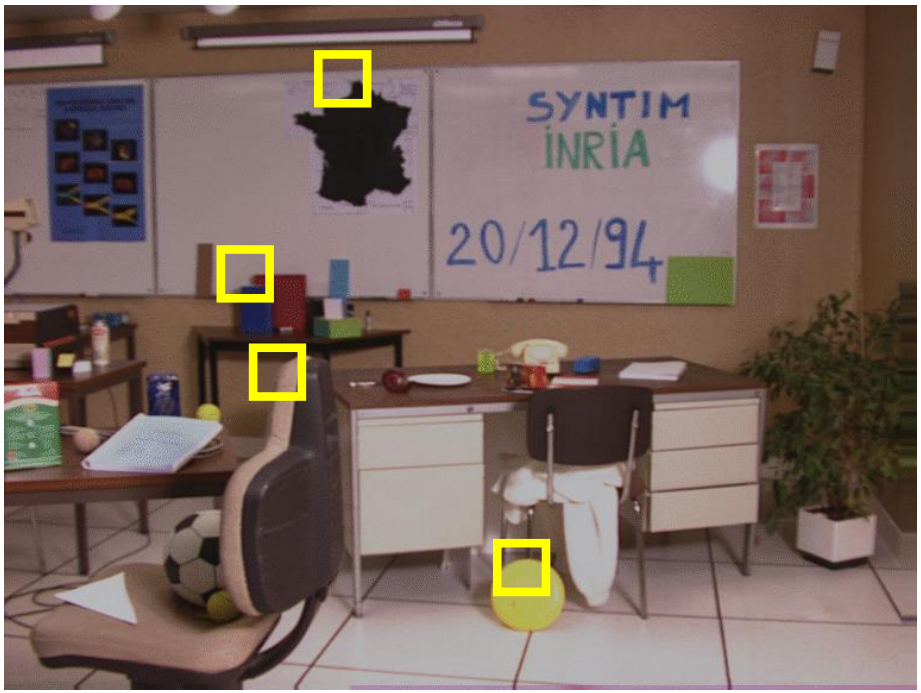
Epipolar Geometry

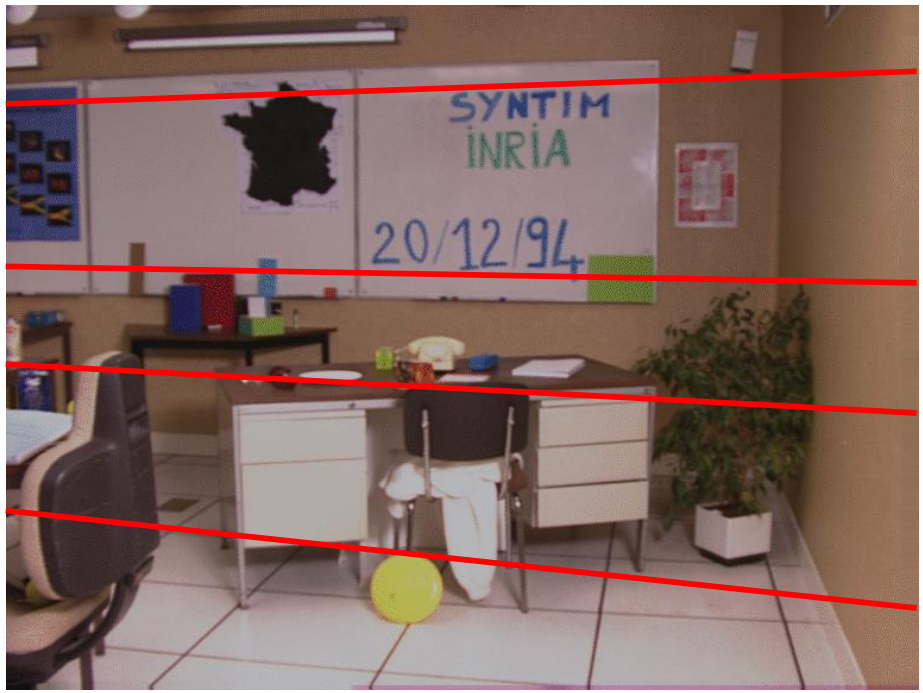
- The epipolar plane is defined by a point P in the world and the optical centres of the camera.



- The epipolar constraint states that corresponding points will always lie along a straight line, allowing us to search for such points in 1D.

Epipolar Constraint





Computing the epipolar line

- To compute the epipolar line, we project the optical ray from C_l to p_l into the second image.

$$\lambda_l \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = K_l \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \lambda_l K_l^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \equiv P_w = \lambda_l K_l^{-1} p_l$$

- We project this into the second image by substituting for P_w to obtain

$$\lambda_r p_r = \lambda_l K_r R K_l^{-1} p_l + K_r t$$

- Searching for the correspondence of a point on the left need only take place on its epipolar line on the right.

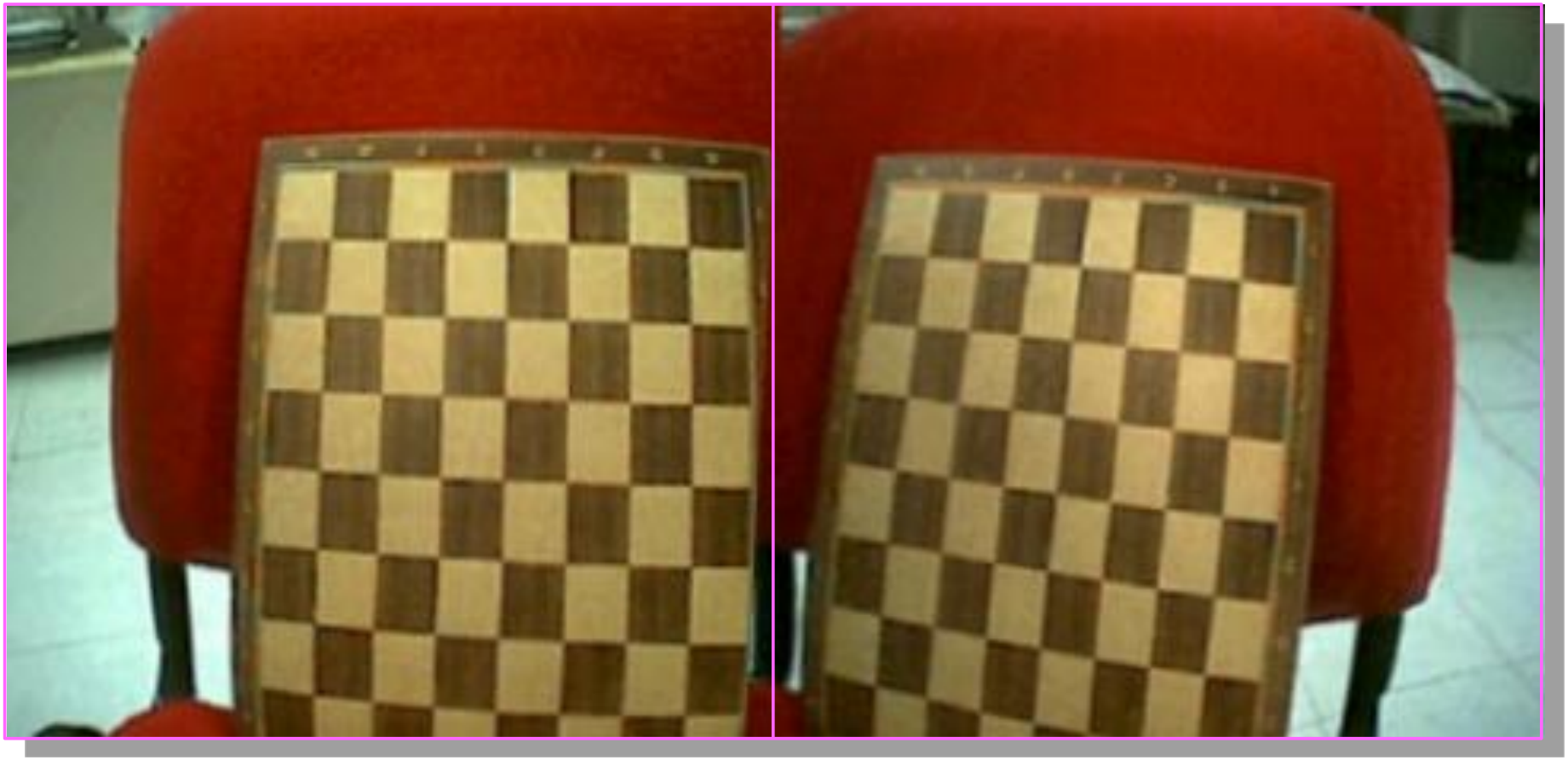
Epipolar Rectification

- We did not take radial distortion into account. Doing so poorly can lead to large errors in the disparity computation.
- Normal process is to undistorted the two images so that epipolar ones are collinear and horizontal.
- This allows correspondence search to be very simple and computationally cheap as only one axis needs to be considered.
- See "A compact algorithm for rectification of stereo pairs", Fusiello et al, Machine Vision and Applications 12(1), pp 16–22, 2000 for details (on myAberdeen).
- Basic steps: rotation; focal length correction ; lens distortion ; translation

Epipolar Rectification

Image from Left Camera

Image from Right Camera



Epipolar Rectification

Image from Left Camera

Image from Right Camera

Epipolar Rectification

Image from Left Camera

Image from Right Camera

Epipolar Rectification

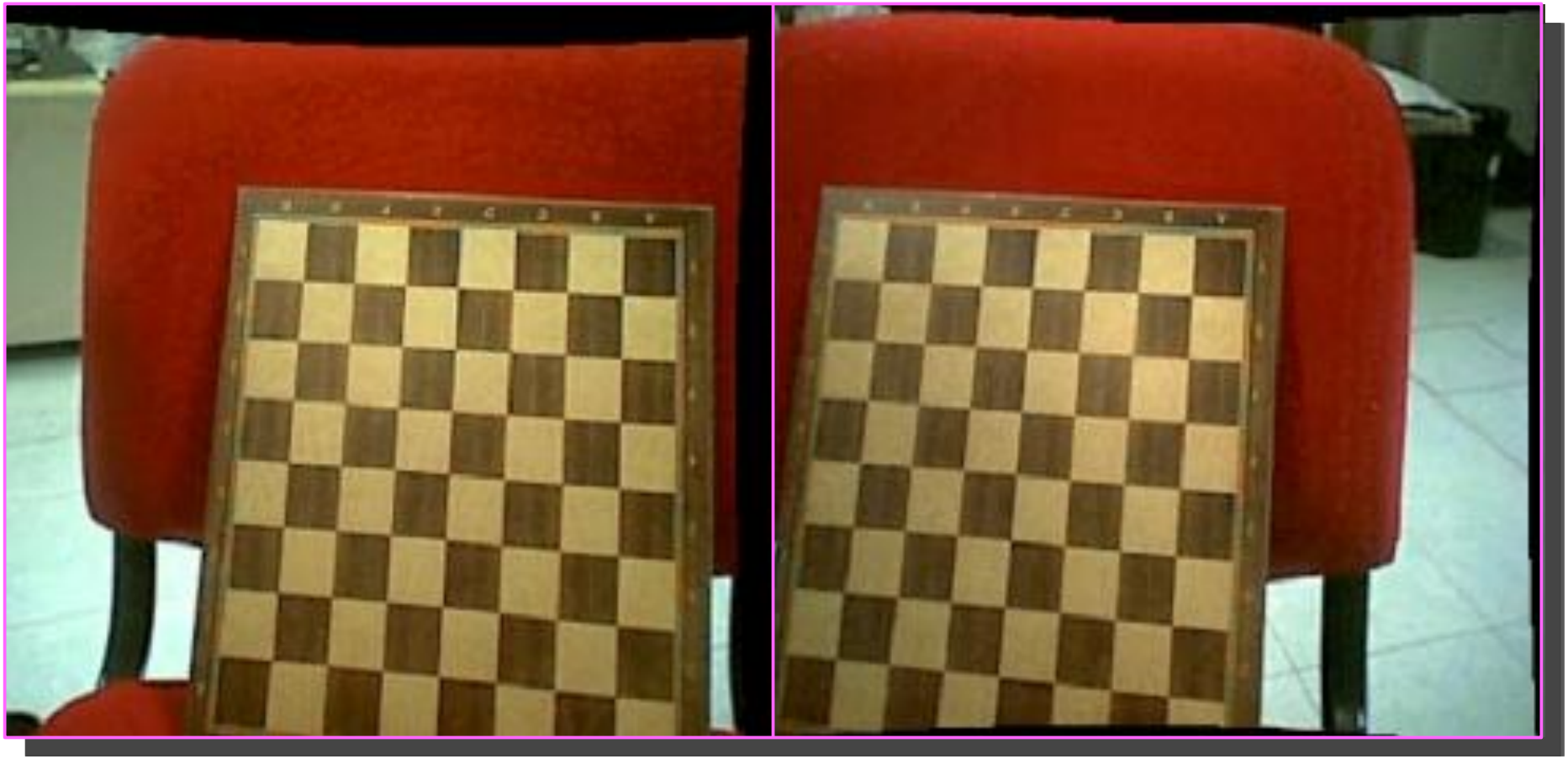
Image from Left Camera

Image from Right Camera

Epipolar Rectification

Image from Left Camera

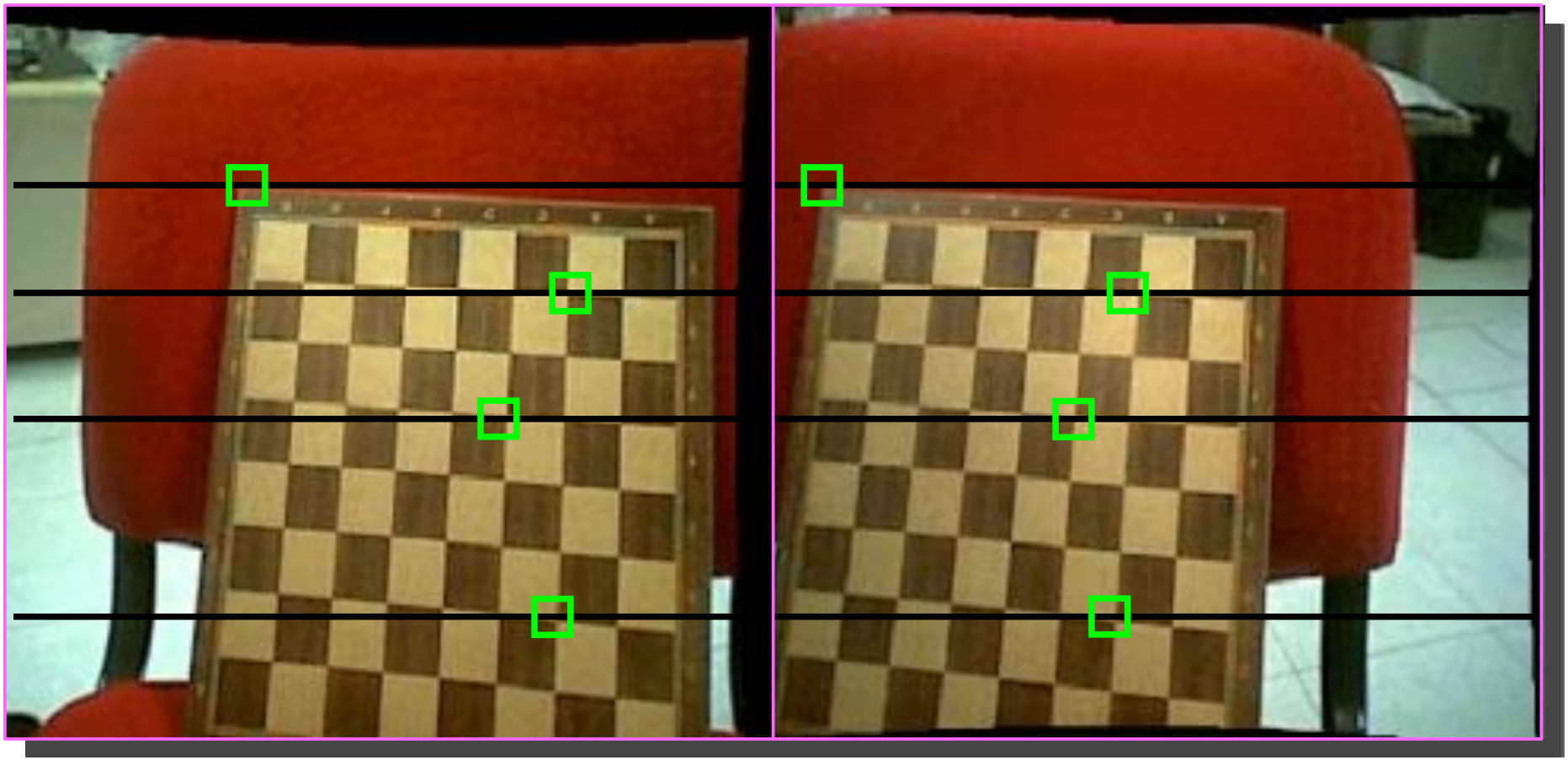
Image from Right Camera



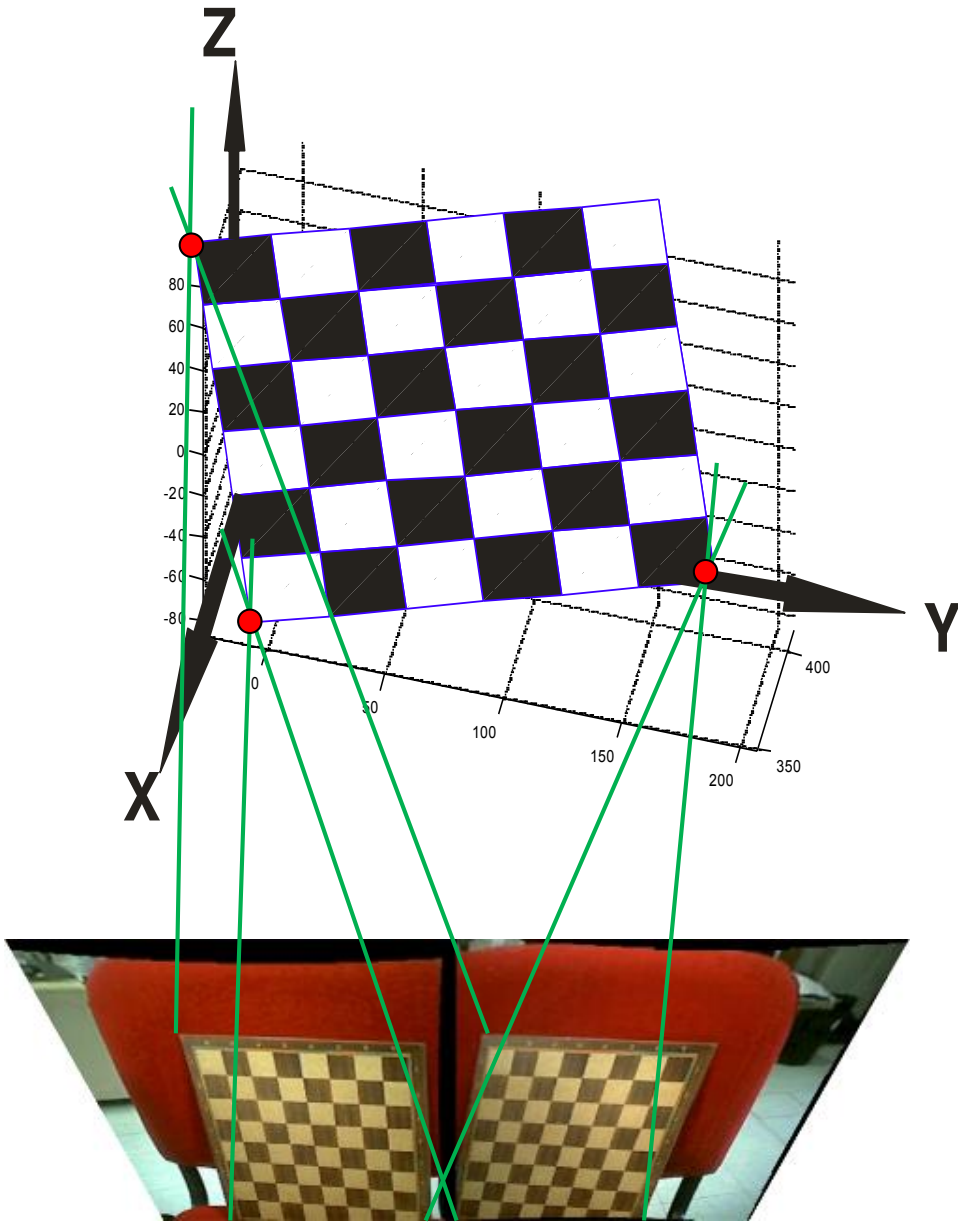
Epipolar Rectification

Image from Left Camera

Image from Right Camera



Stereo Output



Where are we?

- We know how to generate depth from a stereo image.
- Basic steps
  1. Calibrate stereo cameras (via intrinsic and pose parameter computation)
  2. Epipolar rectification to align images and obtain epipolar lines
  3. Search for correspondences
  4. We obtain depth for each point in the image

Structure from Motion

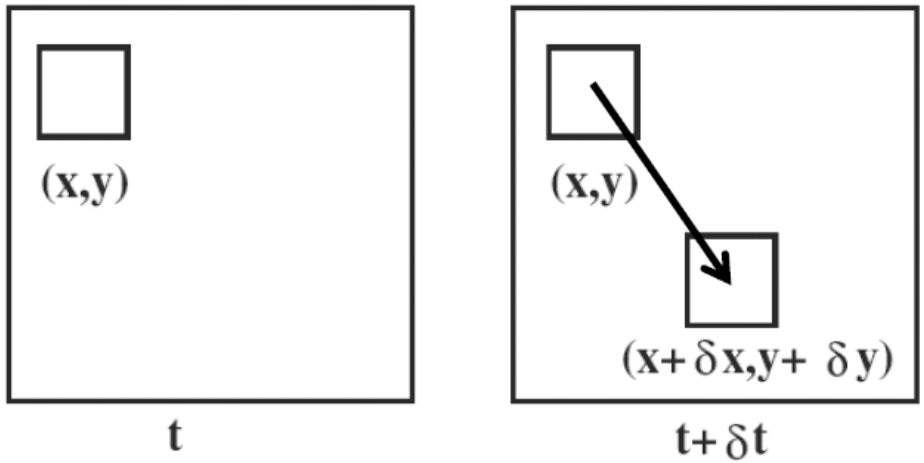
- Structure from motion uses a single camera which moves. We must compute camera motion as well as depth in scene. Can obtain equality to some scale.
- http://http://grail.cs.washington.edu/rome/
- http://www.youtube.com/watch?v=HrgHFDPJHXo&feature=player_embedded
- See also http://www.morethantechnical.com/2012/02/07/structure-from-motion-and-3d-reconstruction-on-the-easy-in
- There are several toolkits available to do structure from motion.

Visual Odometry

- Visual odometry estimates the motion of a robot using visual input alone.
- Used successfully on Mars rovers with stereo cameras
- Can also be done with a single camera (but can't estimate scale)
- Suffers from drift due to integration of relative displacements between consecutive poses (and environment).
- Drift can be cancelled out if a place has already been observed (unlike for normal odometry).

Optical Flow

- Aims to obtain a vector field representing motion over frames (`https://www.youtube.com/watch?v=ysGM3CfBVpU`).
- The motion field assigns a velocity vector to every point in an image.
- Optical flow - the apparent motion of patterns in the image.
- Both are normally, but not always in correspondence (see `http://www.gocognitive.net/sites/default/files/lightFromAbove.demo_.v1.1.swf`).
- Optical flow works by computing the motion vectors of all pixels in an image across multiple images.



Color tracking

- Color sensing provides a unique cue for vision algorithms; when combined with other sensing modalities, it can greatly reduce errors.
- Often implemented in hardware.
- Simplest approach is constant thresholding — a pixel point is selected if each of its R,G,B values fall within some min/max threshold values.
- YUV colour space can also be used, yielding greater satbility to illumination changes.
- `http://www.youtube.com/watch?v=HPB69CfgyMQ&feature=related`
- More complex approaches can be used for floor plane extraction to find parts of the ground that can be safely traversed.

Where are we?

- We have examined the fundamentals of computer vision.
- Pinhole camera model
- Perspective projection
- Stereo vision
- Optical flow
- Color tracking

Image Processing

- Image processing alters an image or provides a set of parameters associated with an image.
  - Filtering, enhancement, edge detection
  - Feature detection
  - Object recognition
  - Segmentation
  - ...
- Many uses, for example identifying identical patches in stereo image in order to do stereopsis.
- We will focus on
  - image filtering, (smoothing, edge detection)
  - Image feature detection and similarity measures (for finding point correspondences)

Image Similarity

- We want to compare the similarity of a $m \times n$ patch at point (u, v) in image I_1 to another patch in I_2 at point (u', v').
- Sum of absolute differences (SAD):

$$SAD = \sum_{k=-a}^{a} \sum_{l=-b}^{b} |I_1(u+k, v+l) - I_2(u'+k, v+l)|$$

- Sum of squared differences (SSD)

$$SSD = \sum_{k=-a}^{a} \sum_{l=-b}^{b} (I_1(u+k, v+l) - I_2(u'+k, v+l))^2$$

- Typically computed on tray intensity levels, and test compares all possible u, v points in one image against all u', v' points in other image.

Zero Normalised Cross Correlation

- Consider two identical scenes with one under bright sunlight and the other taken in the shade.
- SAD and SSD will not show good correlation.
- ZNCC overcomes this.
- Essentially computes the dot product of the two patches (represented as vectors) normalised by the average intensity of the patch in each image.

$$\frac{\sum_{k=-a}^{a}\sum_{l=-b}^{b}(I_1(u+k,v+l)-\mu_1)\cdot(I_2(u'+k,v'+l)-\mu_2)}{\sqrt{\sum_{k=-a}^{a}\sum_{l=-b}^{b}(I_1(u+k,v+l)-\mu_1)^2\sum_{k=-a}^{a}\sum_{l=-b}^{b}(I_1(u'+k,v'+l)-\mu_2)^2}}$$

$$\mu_1 = \frac{1}{mn}\sum_{k=-a}^{a}\sum_{l=-b}^{b} I_1(u+k, v_+ l)$$

$$\mu_2 = \frac{1}{mn}\sum_{k=-a}^{a}\sum_{l=-b}^{b} I_2(u'+k, v'+l)$$

Image Filtering

- We want to accept or reject certain components of an image.
- For example a low pass filter allows low frequencies of an image through; blurring the image.
- We will typically use spatial filters instead of frequency based filters.

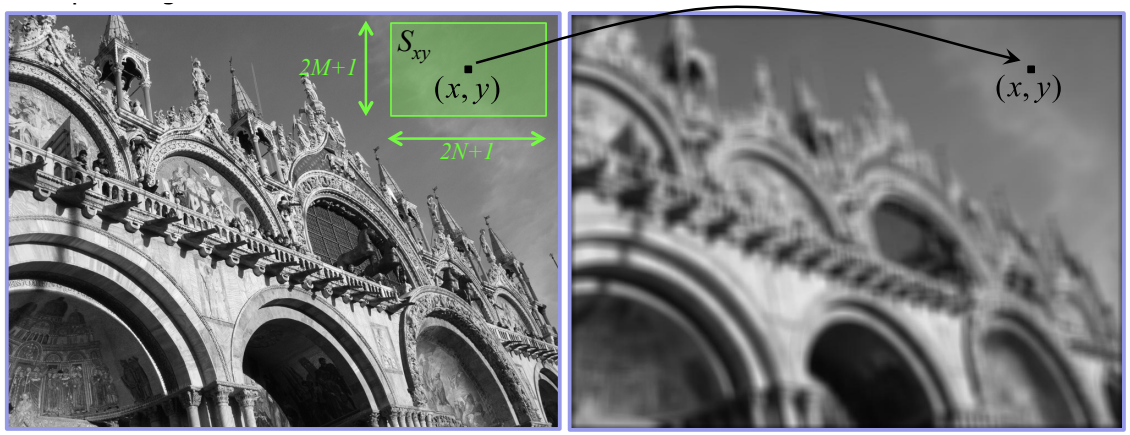
Lowpass filtered image

Highpass filtered image

Spatial Filter

- A spatial filter operating on an image I consists of
  1. a neighbourhood S_{xy} of the pixel at x, y under examination; and
  2. a predefined operation T performed on the image pixels in the neighbourhood
- Spatial filtering generates a corresponding pixel in a new image I', determined by $T(S_{xy})$
- Example: averaging filter

$$I'(x,y) = mn^{-1} \sum_{r,c \in S_{xy}} I(r,c)$$



Filters

- We focus on linear shift-invariant filters.
  - Linear: every pixel is replaced by a linear combination of its neighbours
  - Shift invariant: the same operation is performed on every point in the image
- A correlation filter:

$$I'(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) I(x+s, y+t)$$

- A convolution filter:

$$I'(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) I(x-s, y-t)$$

- Note that w is also called a kernel, mask or window.
- Convolution is associative: $F * (G * I) = (F * G) * I$
- To create a specific type of filter, the mn coefficients of the kernel must be specified.

Smoothing filter

- The averaging filter is a type of smoothing filter.
- Used for blurring and noise reduction.
- Output is a weighted average of the pixels contained in the filter mask.
- Results in reduced sharp transitions.
- Side effect: edges get blurred.
- Consider a 3×3 averaging filter: $w = 1/9$ where s and t range from -1 to 1.
- Here $w(s,t) = 1/9$ for any $-1 \leq s, t \leq 1$.

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

Smoothing filter

- The averaging filter is a type of smoothing filter.
- Used for blurring and noise reduction.
- Output is a weighted average of the pixels contained in the filter mask.
- Results in reduced sharp transitions.
- Side effect: edges get blurred.
- Consider a 3×3 averaging filter: $w = 1/9$ where s and t range from -1 to 1.
- Here $w(s,t) = 1/9$ for any $-1 \leq s, t \leq 1$.
- It is more efficient to compute this as

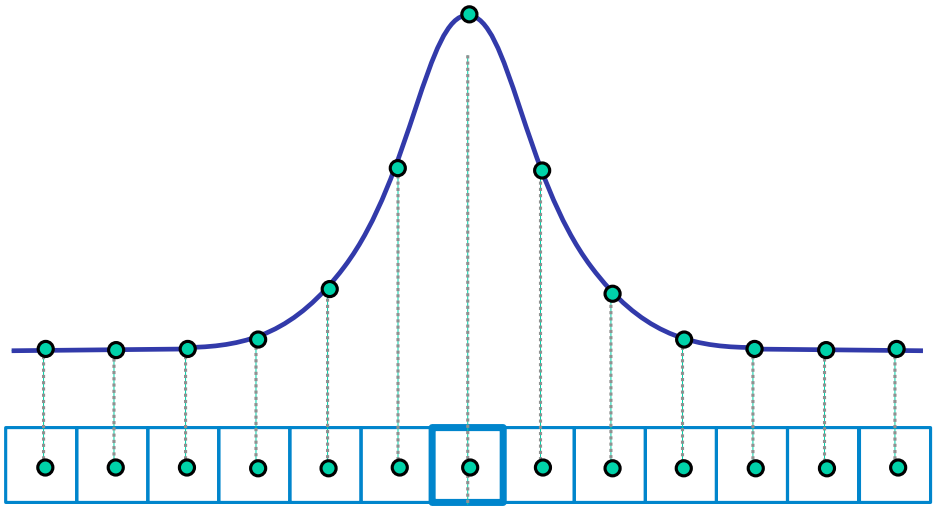
$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gaussian Smoothing

- Common practice for image smoothing uses a Gaussian

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

- $\mu = 0$, σ controls the amount of smoothing.
- Pixels close to the centre have a bigger influence on the averaged value when compared to more distant ones.



Gaussian Smoothing

- In 2 dimensions, we sample around its centre, rounding and ensuring that all coefficients sum to 1.
- E.g. in the 3×3 case with $\sigma = 0.85$ we obtain

$$G = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Symmetry means that it doesn't matter if we convolve or use correlation.
- Result of convolving G to an image I can be written $G_\sigma * I$

Gaussians in 2D

- in 2 dimensions

$$G(x,y) = \frac{1}{2\pi|S|^{1/2}} e^{0.5 \begin{bmatrix} x \\ y \end{bmatrix} S^{-1} [x \; y]}$$

- We usually want to smooth by the same amount in both x and y directions, so

$$S = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

- Multiplying out, we see that

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{y^2}{2\sigma^2}} = G_\sigma(x) G_\sigma(y)$$

- This is therefore a separable filter.

Derivative Filter

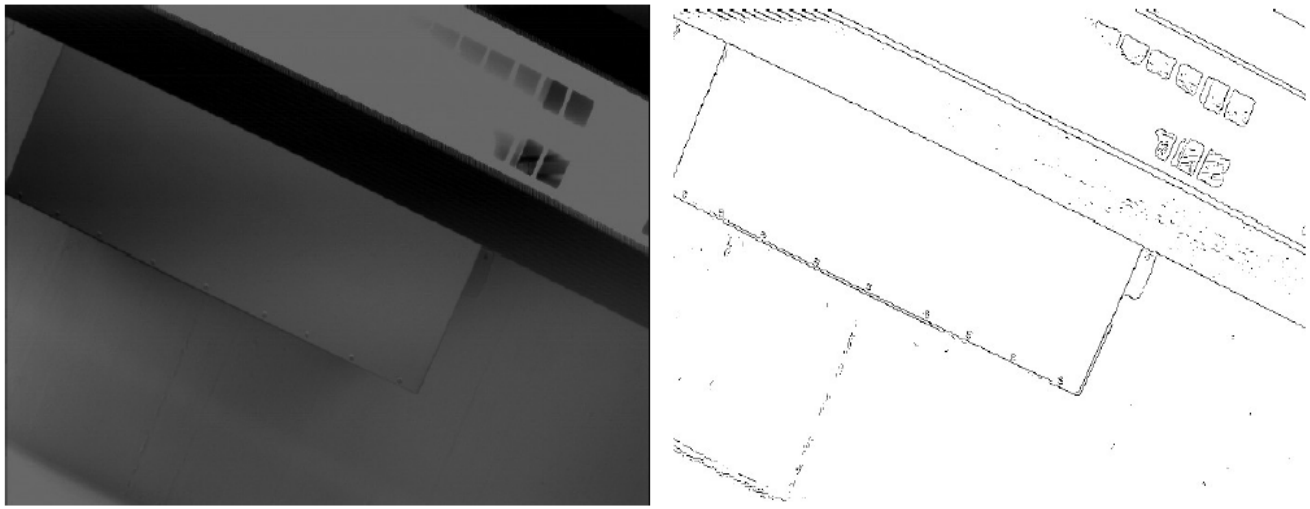
- The derivative of an image aims to quantify how quickly intensities change (along the direction of the derivative).
- 1D case:

$$J(x) = \frac{I(x+1) - I(x-1)}{2}$$

- 2D case is similar (in case of derivatives along an axis).

Edge Detection

- The goal of edge detection is to obtain an idealised line drawing of the scene.
- Significantly reduces amount of information in an image.
- It is usually assumed that edge contours correspond to important scene contours. This assumption is false.

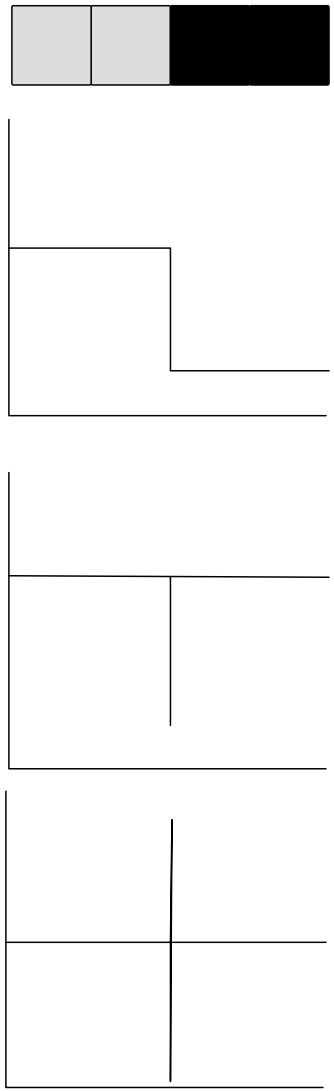


Edge Detection

- Edges correspond to significant changes in image brightness.
- How do we measure this?

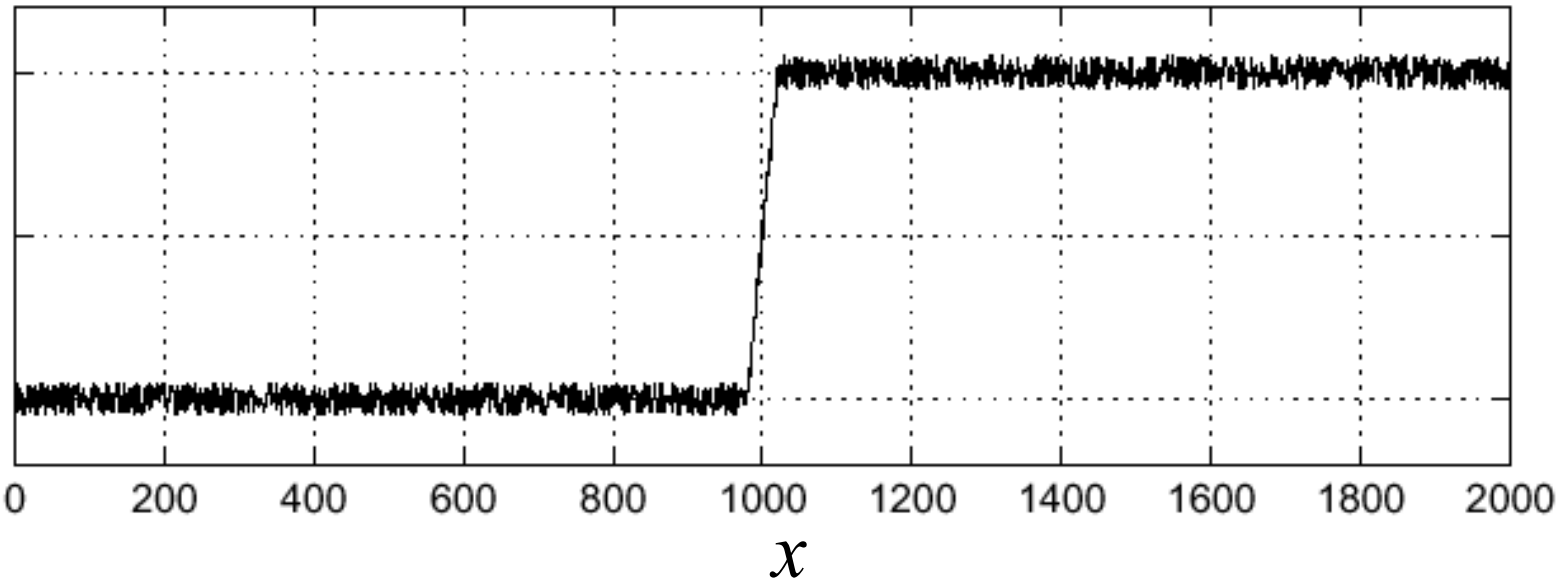
Edge Detection

- Edges correspond to significant changes in image brightness.
- How do we measure this?
- Change is measured by first order derivative.
- A large change in intensity means that the magnitude of the derivative is large.

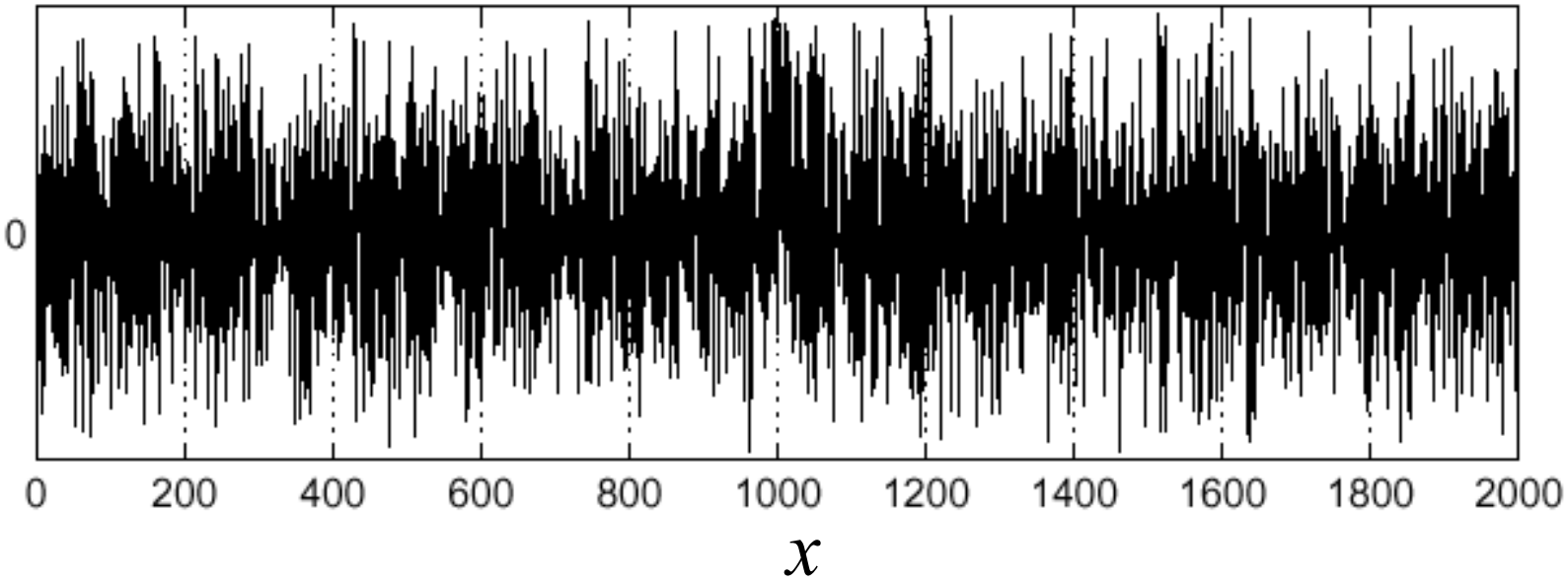


Edge Detection

$I(x)$



$\frac{d}{dx} I(x)$



Edge Detection

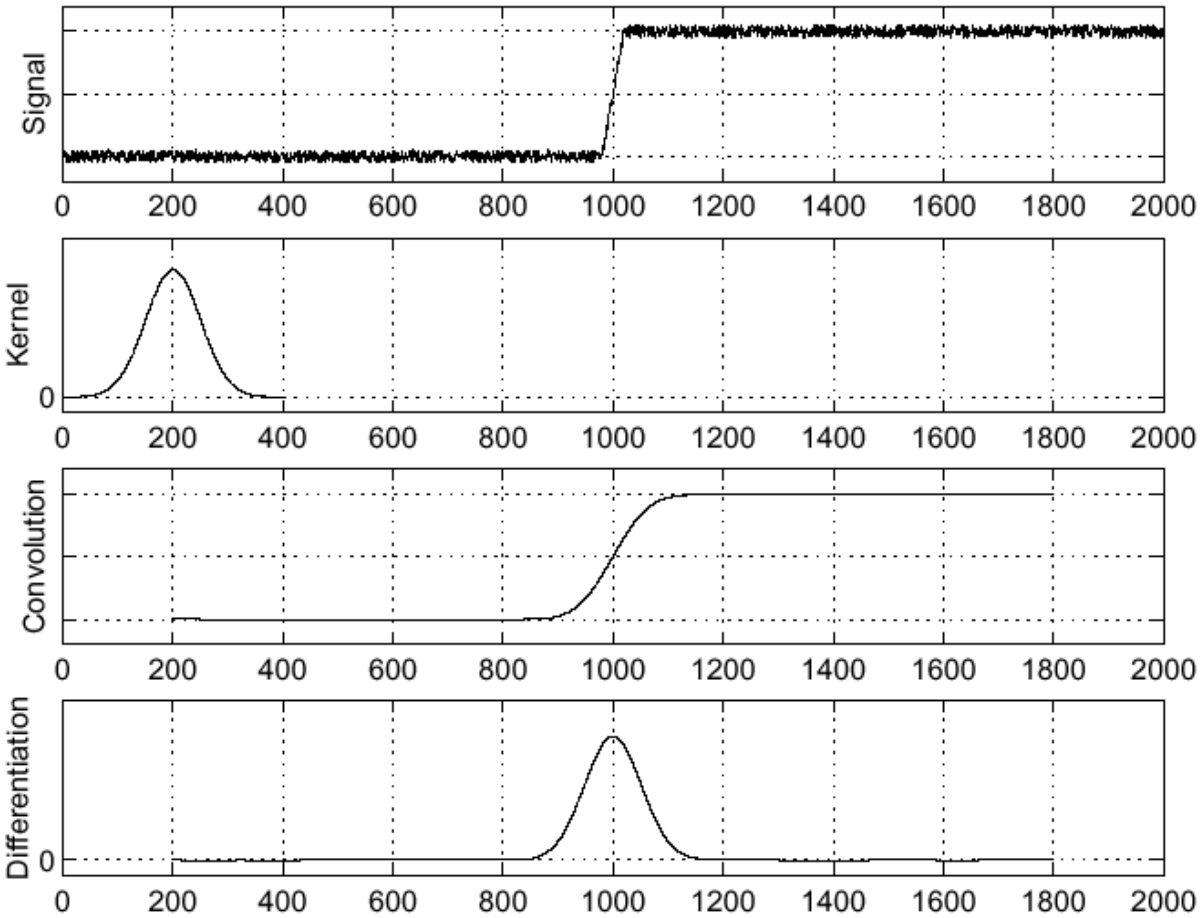
- Smooth first and then detect edges:

$$I(x)$$

$$G_\sigma(x)$$

$$s(x) = I(x) * G_\sigma(x)$$

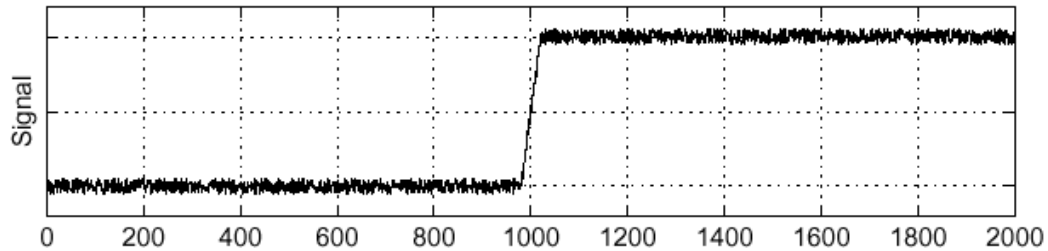
$$s'(x) = \frac{d}{dx}\big(s(x)\big)$$



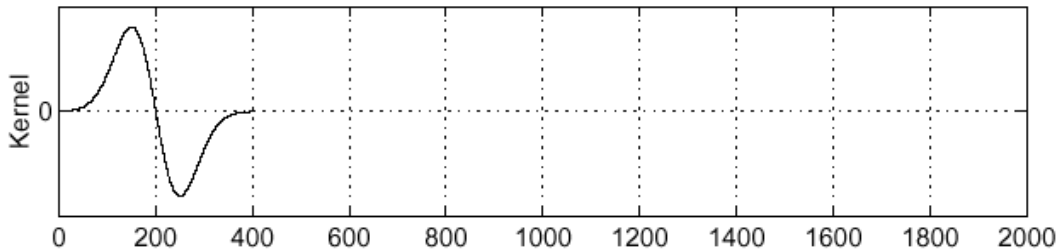
Edge Detection

- $(G * I)' = G' * I$, saving us one operation

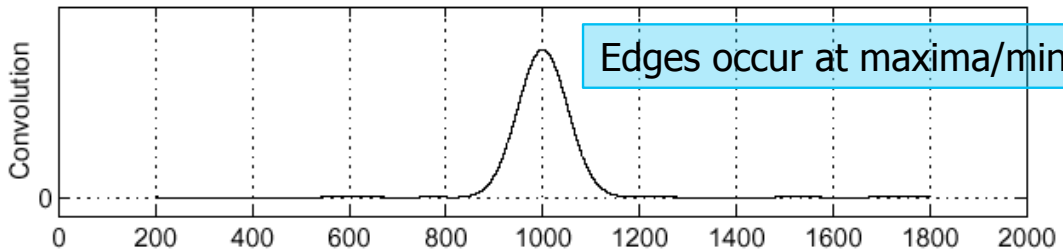
$I(x)$



$$G'_\sigma(x) = \frac{d}{dx} G_\sigma(x)$$



$$s'(x) = G'_\sigma(x) * I(x)$$



Edges occur at maxima/minima of $s'(x)$

Edge Detection

- Gaussian filter is separable: $G_\sigma(x,y) = G_\sigma(x)G_\sigma(y)$
- So gradient of a smoothed image in both directions is

$$\nabla(G_\sigma * I) = \begin{bmatrix} \frac{\partial(G_\sigma * I)}{\partial x} \\ \frac{\partial(G_\sigma * I)}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial G_\sigma}{\partial x} * I \\ \frac{\partial G_\sigma}{\partial y} * I \end{bmatrix} = \begin{bmatrix} G'_\sigma(x)G_\sigma(y) * I \\ G'_\sigma(y)G_\sigma(x) * I \end{bmatrix} = \begin{bmatrix} f_V(x,y) * I \\ f_H(x,y) * I \end{bmatrix}$$

- Edge detection algorithm:
  1. Convolve $I(x,y)$ with $f_H(x,y)$ and $f_H(x,y)$ to obtain gradient components $R_V(x,y)$ and $R_H(x,y)$
  2. Compute $R(x,y) = R_V^2(x,y) + R_H^2(x,y)$
  3. Discard pixels with $R(x,y)$ below some threshold
- Threshold can be computed dynamically, based on an intensity histogram.

Example

Original

Example

Compute R the edge strength

Example

Thresholding

Nonmaxima suppression

- Edges are “fuzzy”.
- Nonmaxima suppression revisits edges, checking if they are a local maximum.
- If so, edge is kept, otherwise value is set to zero.
- This reduces the thickness of edges to 1 pixel.

Example

Where are we?

- We can obtain signals from sensors about the environment.
- We can treat these signals to reduce noise and information.
- But uncertainty still exists.
- We therefore aim to extract information from sensor readings, generating percepts which can then be used to inform the robot's model.
- This process is called feature extraction

Features

- Very low level features
  - Edges
- Low level features
  - lines, points corners, blobs
- High level features
  - doors, tables etc

Features - another taxonomy

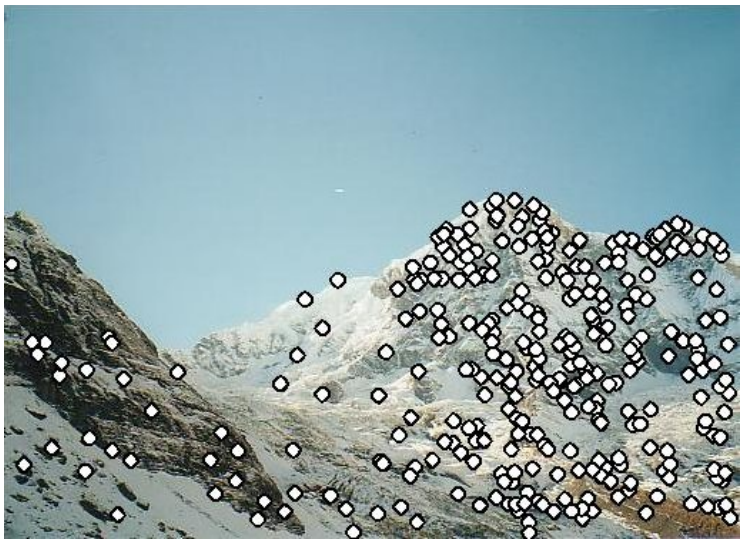
- Features that have semantic interpretation – edges corresponding to lanes in the road; blobs corresponding to blood cells.
- Features without semantic interpretation — what features represent is irrelevant, but their location is important. Used for feature tracking, camera calibration, 3d reconstruction, panorama stitching etc.
- Features with no individual semantic interoperation, but grouped can be used to recognise an object — texture analysis; scene classification; video mining; image retrieval.

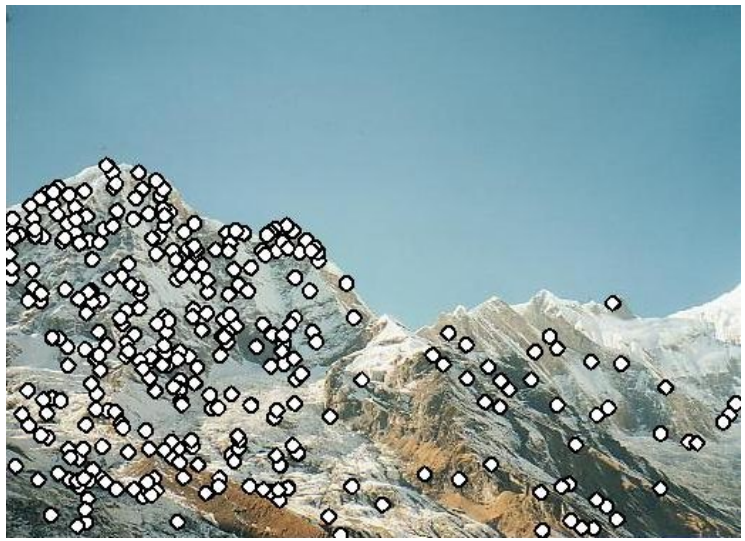
Sample Application — Panorama Building

- We need to align images in order to build a panorama


Sample Application — Panorama Building

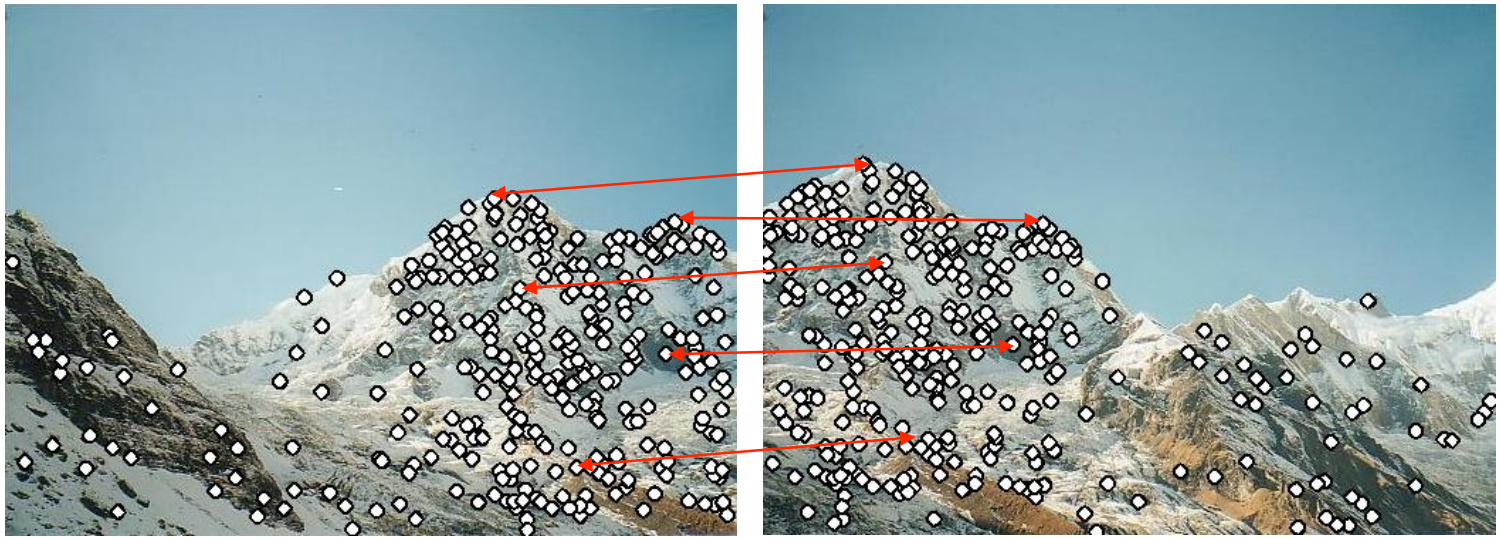
- Detect feature points in both images





Sample Application — Panorama Building

- Detect feature points in both images
- Find corresponding pairs

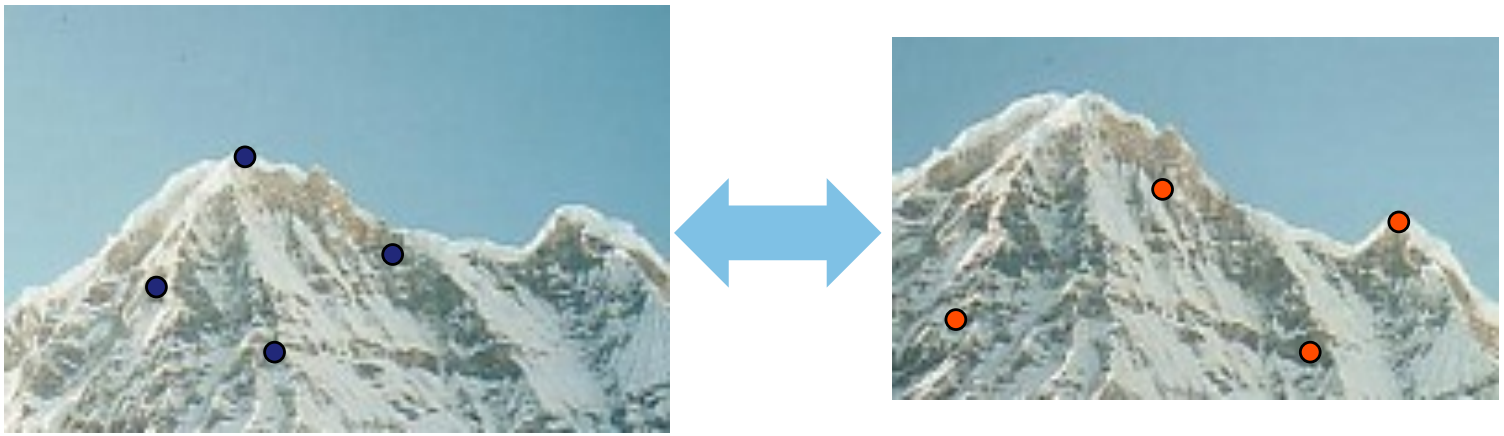


Sample Application — Panorama Building

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

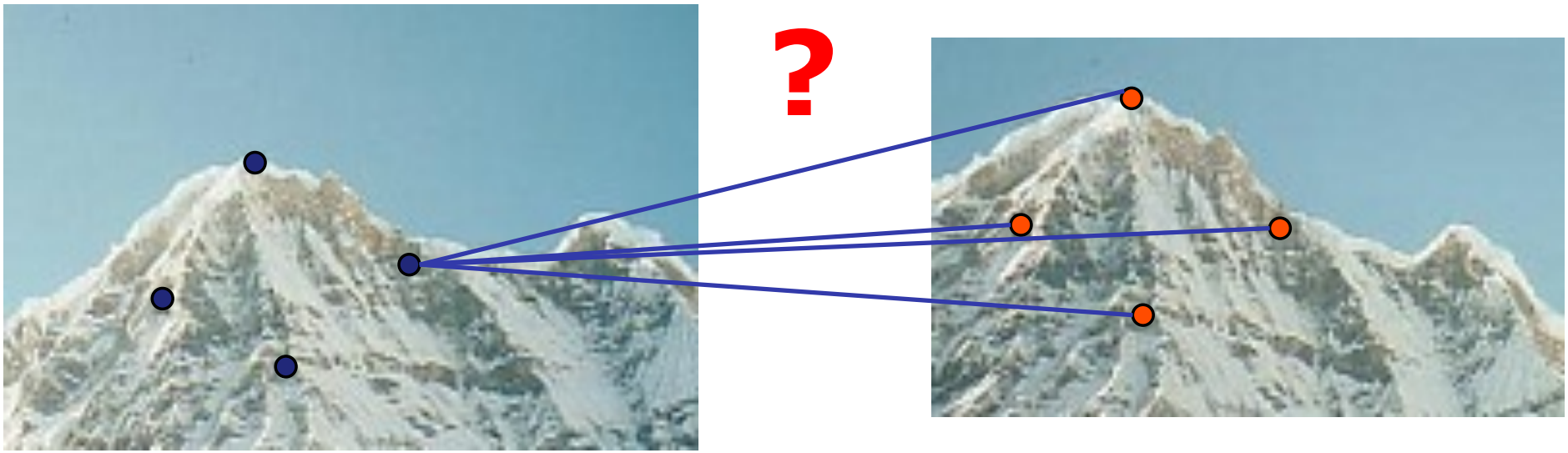
Repeatability

- How can we ensure that we detect the same points independently in both images (if the same points exist)?



- We require a repeatable detector — even under different lighting conditions, rotation and zoom, the same features should be detected in both images.

Distinctiveness



- Information carried by a patch surrounding a feature point should be as distinctive as possible to ensure that features can be distinguished and matched.

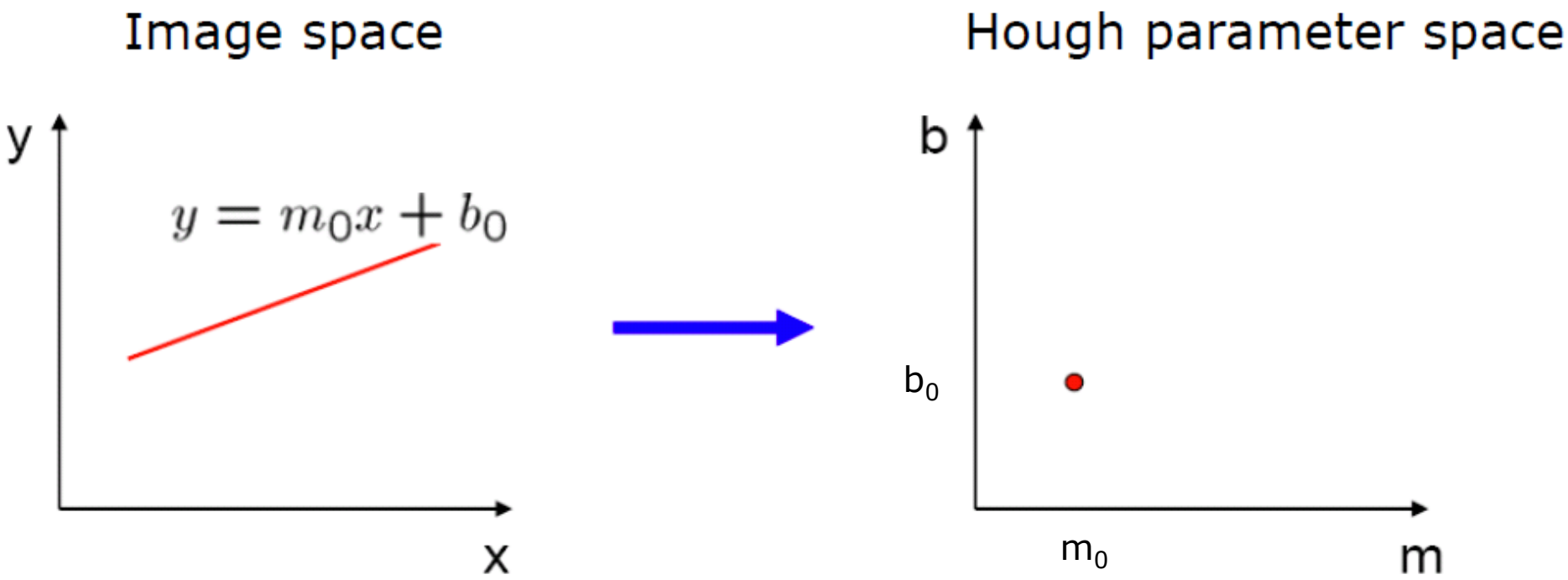
Other Desirable Features

- Localisation accuracy: features should be accurate y localised in position and scale.
- Quantity: while application dependent, typically we require either many, or very few features (e.g. few semantic features such as “door”, or many features for stiching).
- Invariance: camera viewpoint, illumination, scale etc should not affect feature detection.
- Computational efficiency
- Robustness: blur, noise, etc should not affect feature detection.

Line detection

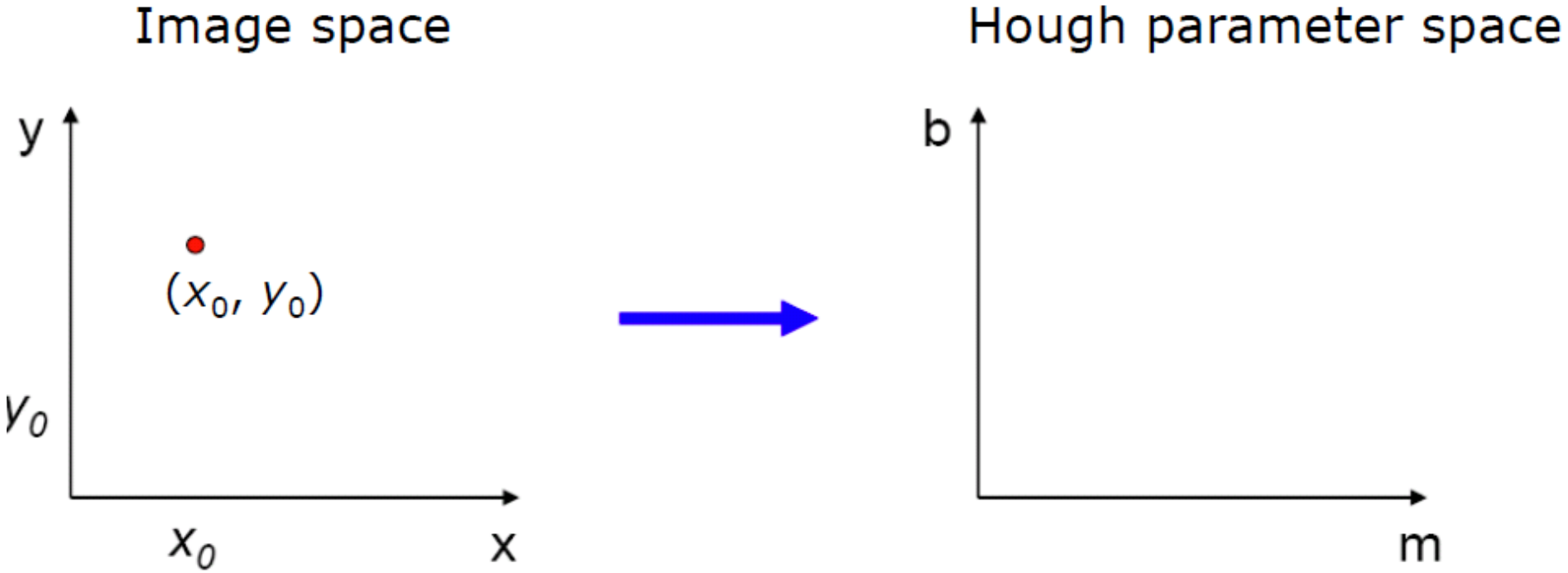
- Lines represent various boundaries in a scene.
- Many uses, e.g. to identify edges of doorways, hallway intersections etc.
- How can we move from points or edges to lines?
- Several popular algorithms, we will focus on the Hough Transform

Hough Transform



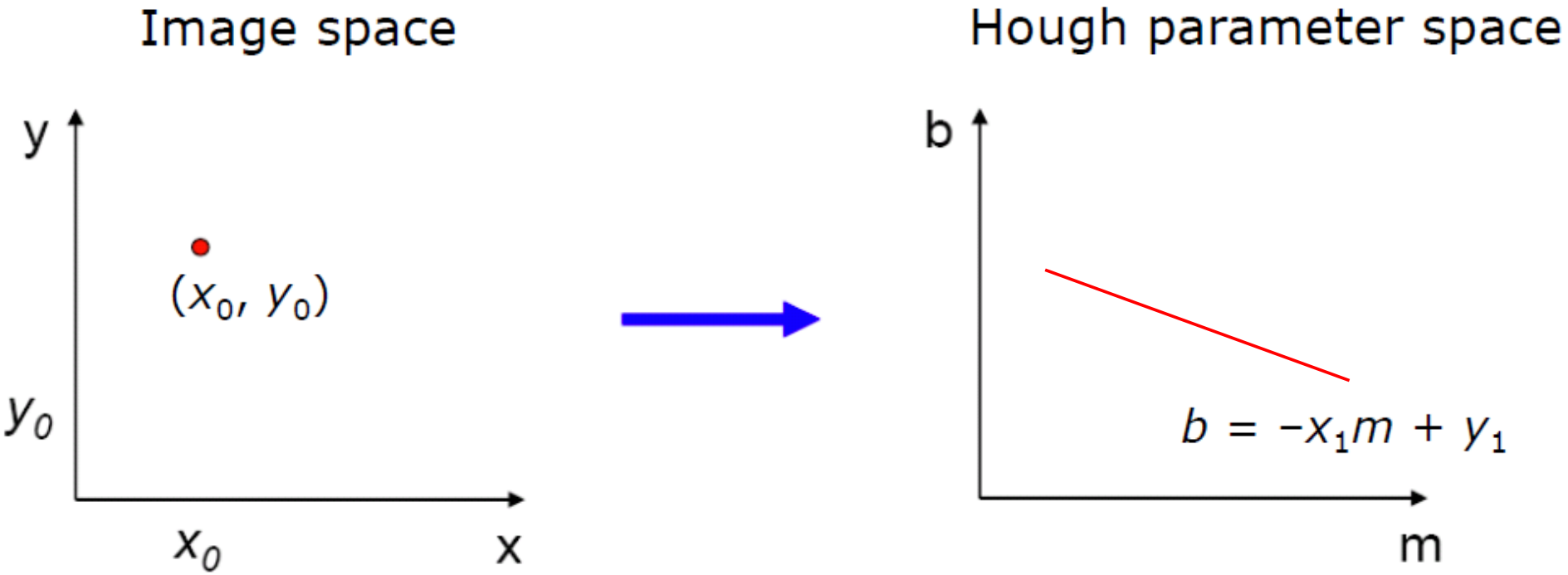
Hough Transform

What does a point (x_0, y_0) in the image space map to in the Hough space?



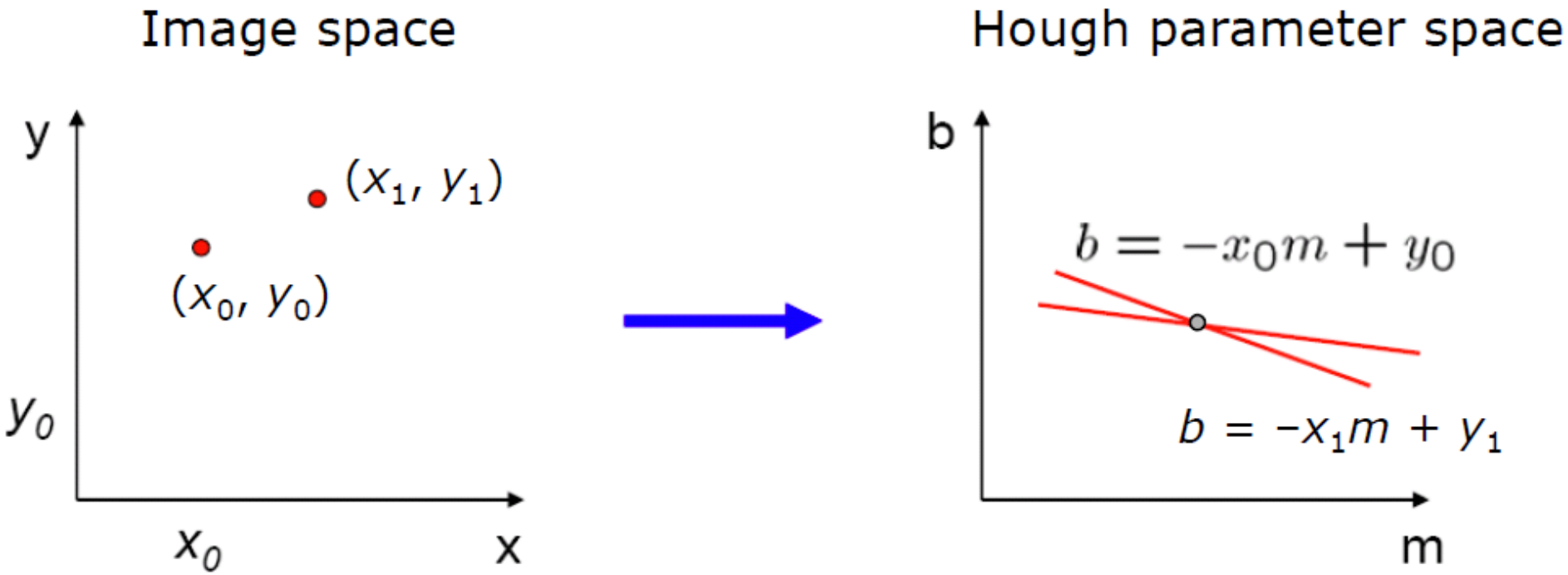
Hough Transform

What does a point (x_0, y_0) in the image space map to in the Hough space?



Hough Transform

- Consider the line containing both points (x_0, y_0) and (x_1, y_1)



Hough Transform

- Partition the Hough parameter space into a grid, and let each point in the image space "vote" for points in the Hough parameter space.

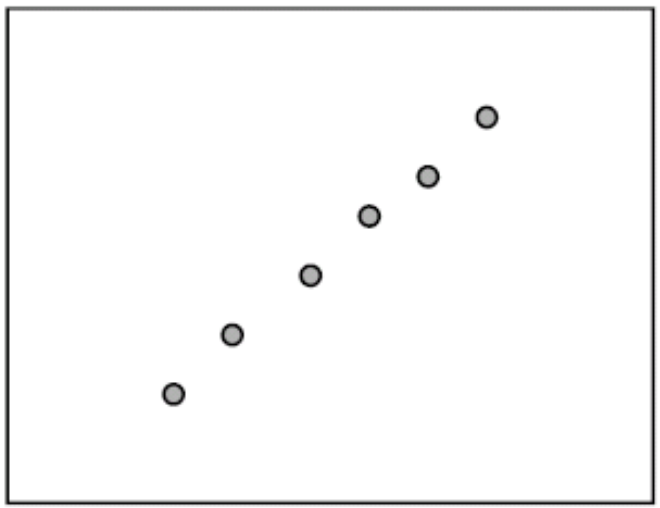
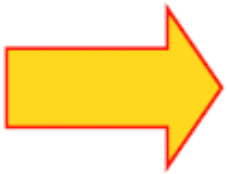
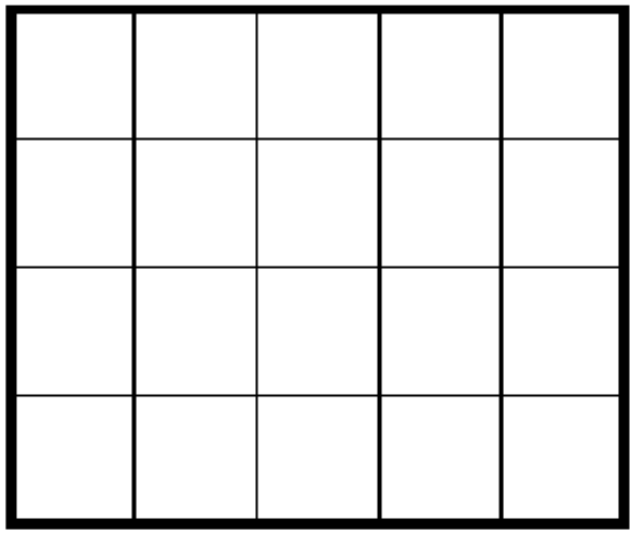


Image space

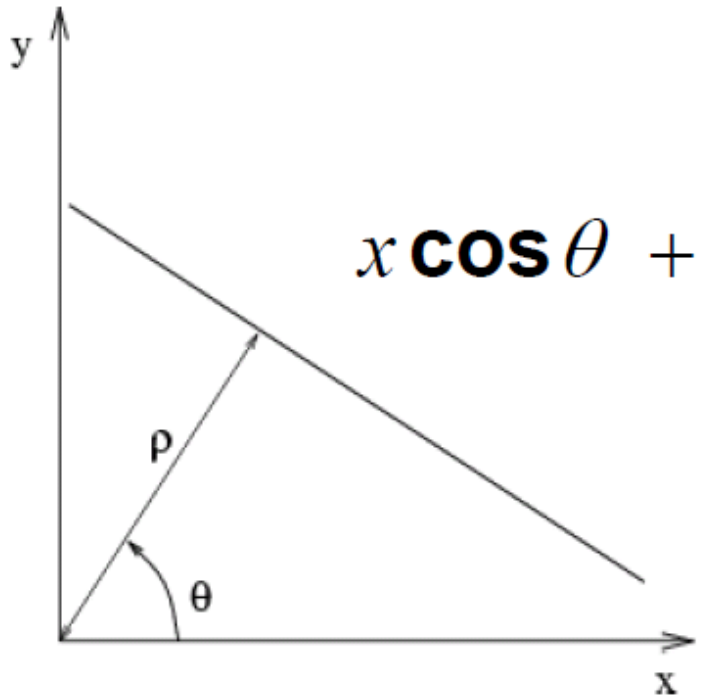




Hough parameter space

Hough Transform

- There's a problem with using the m, c Hough parameter space:
  - The parameter domain is potentially infinite
  - Vertical lines require m to be infinite
- Instead, we note can use a polar line representation

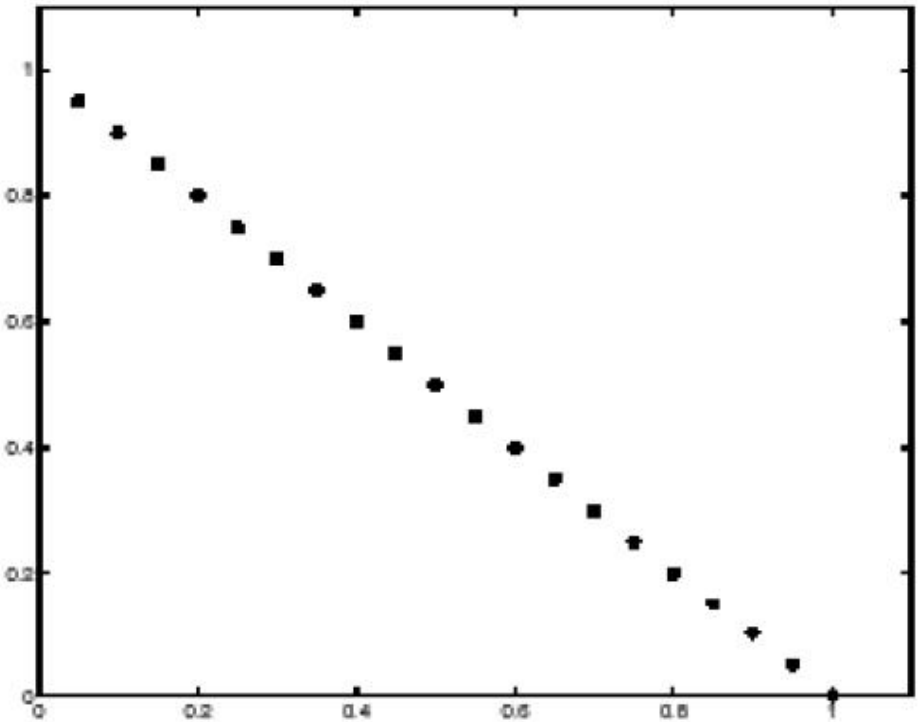


$$x\cos\theta + y\sin\theta = \rho$$

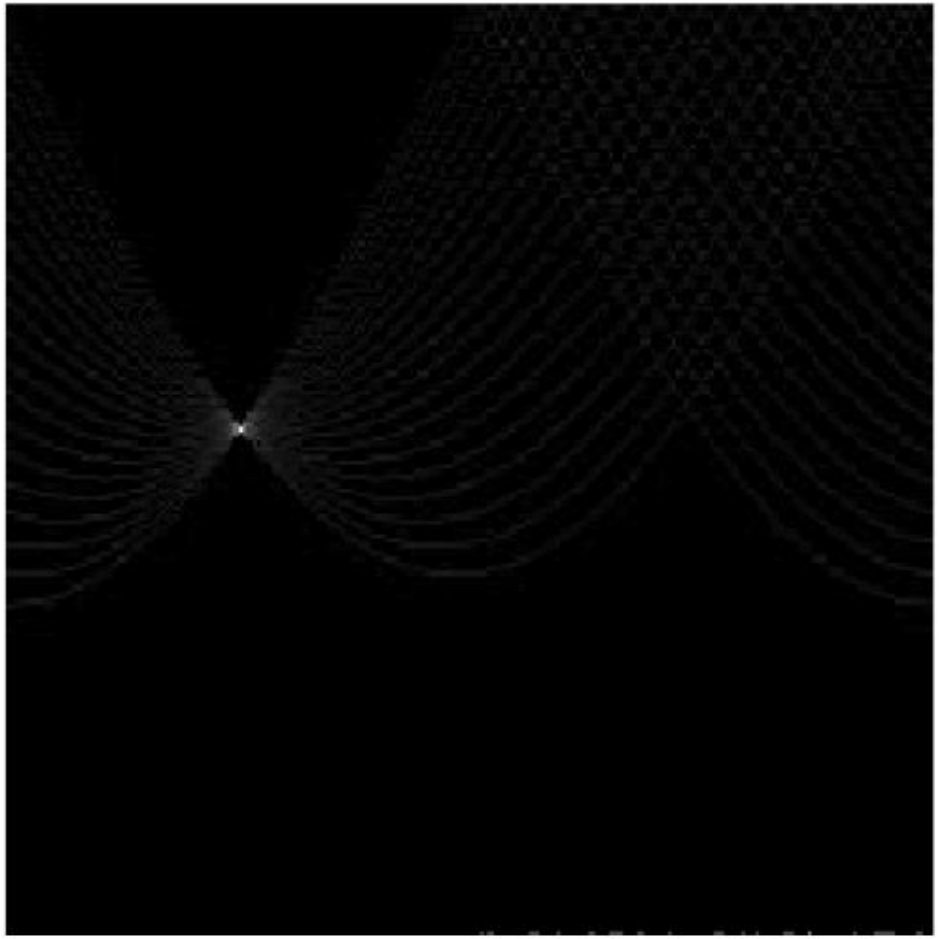
Hough Algorithm

1. Create a 2D accumalator array H, initialised with 0
2. For each point of interest (x,y) in the image
   - Compute $\rho = x cos\theta + y sin\theta$
   - $H(\theta, \rho) + +$
3. Find the values of (θ, ρ) which are a local maximum
4. The detected line is then $\rho = x cos\theta + y sin\theta$

Hough Transform

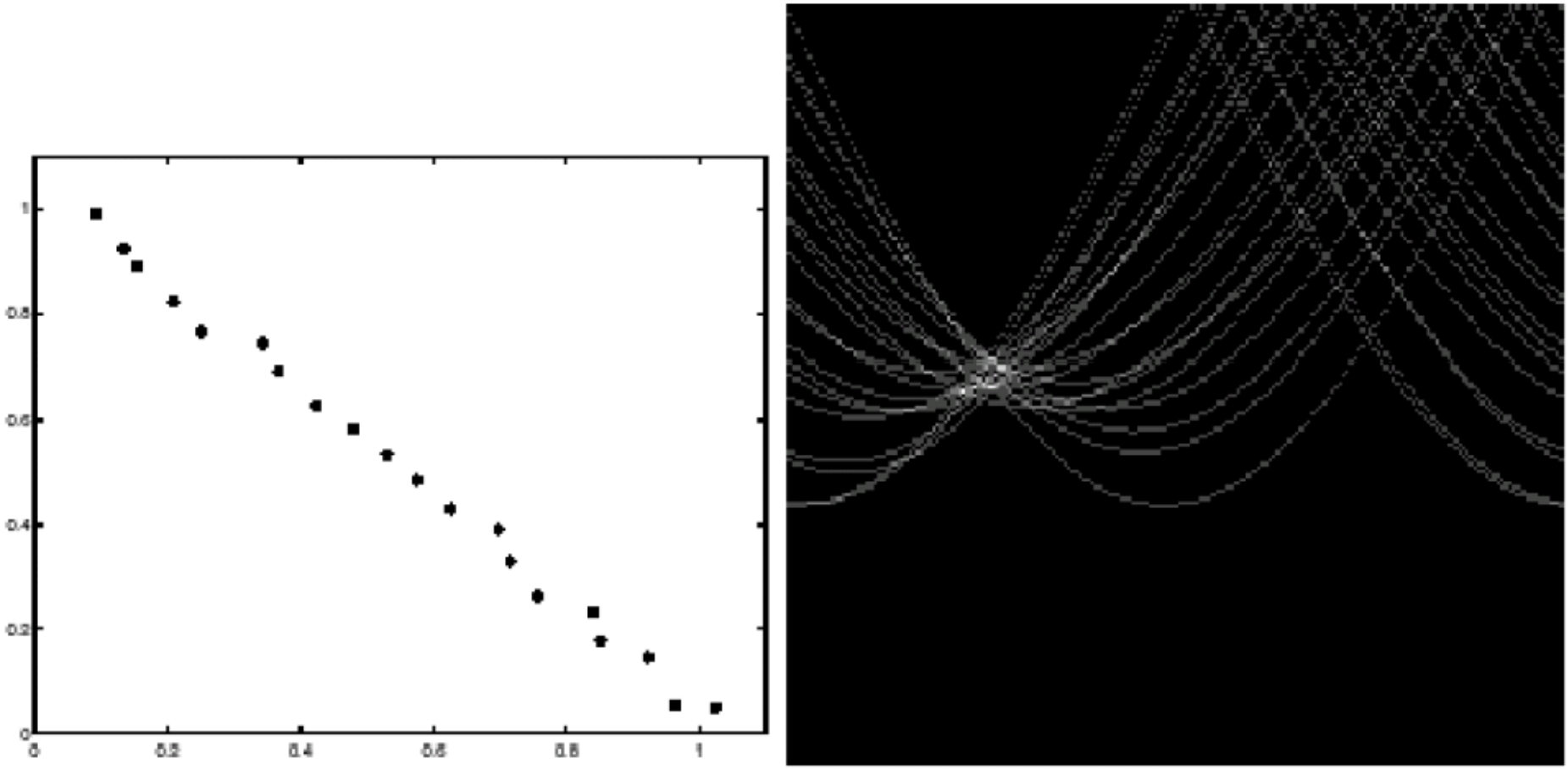


features



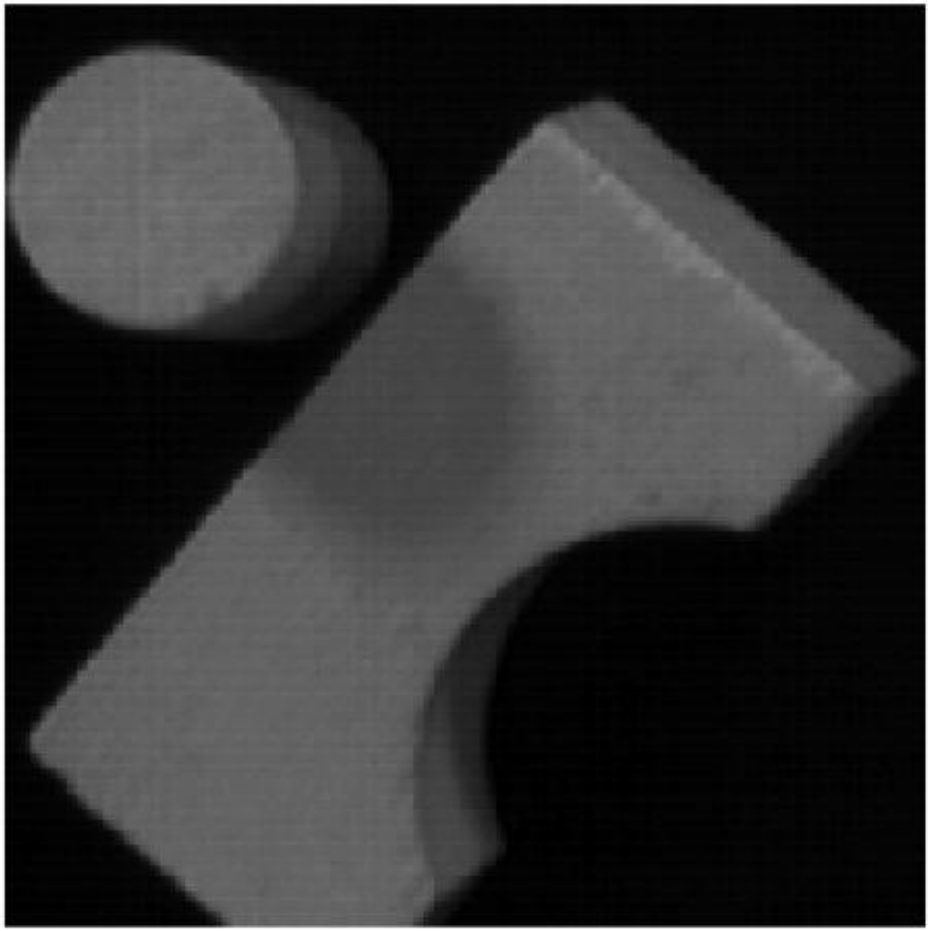
votes

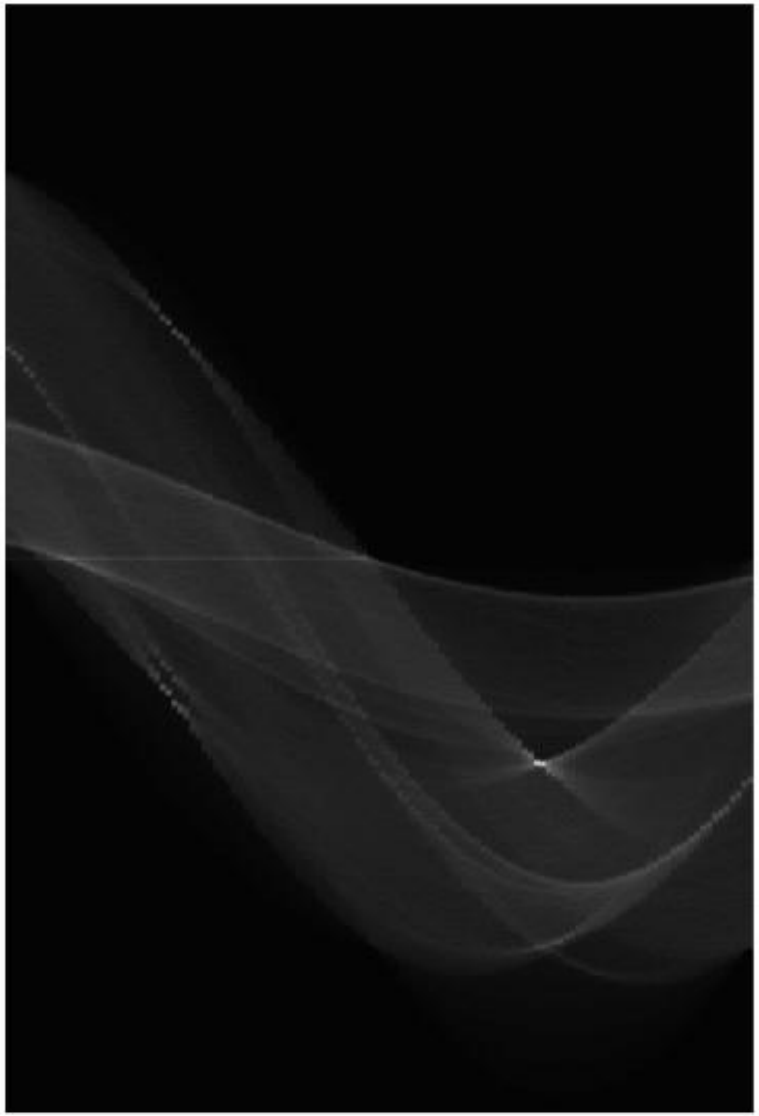
Hough Transform



- Noise makes peaks fuzzier and harder to locate
- In practice, choosing a good grid size is difficult

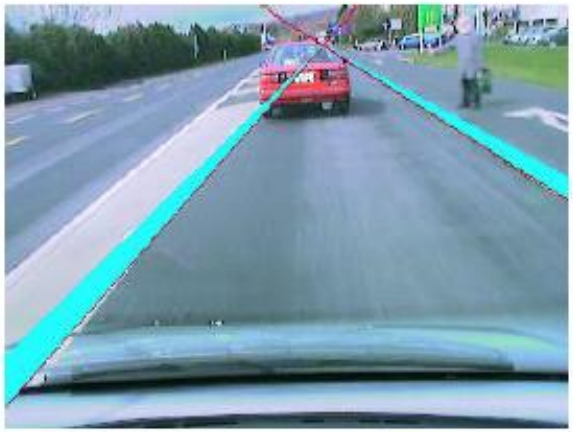
Hough Transform





Hough Transform

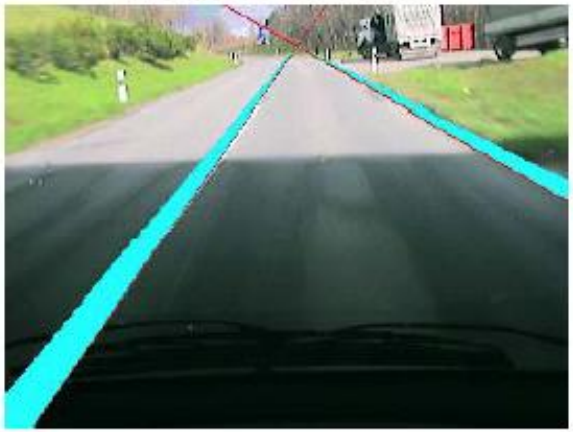
Inner city traffic



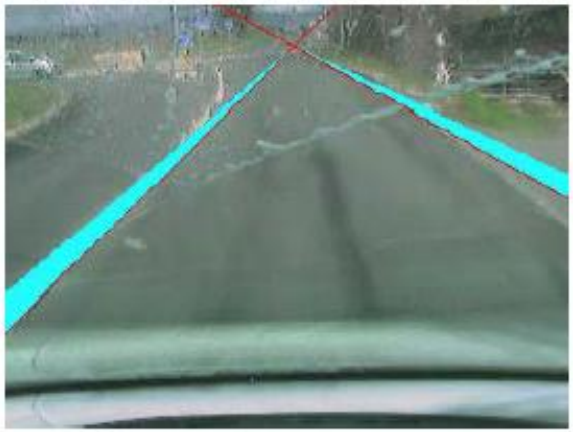
Ground signs

Country-side lane

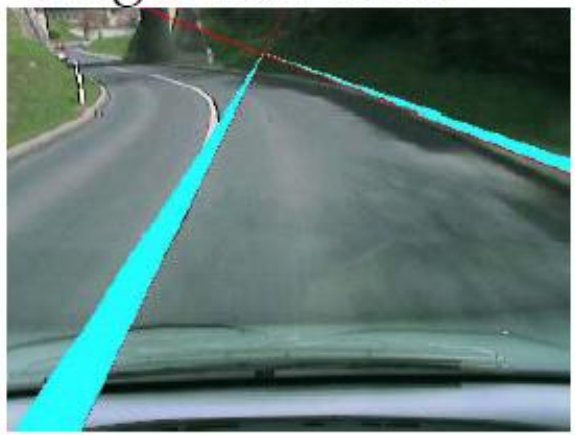
Tunnel exit



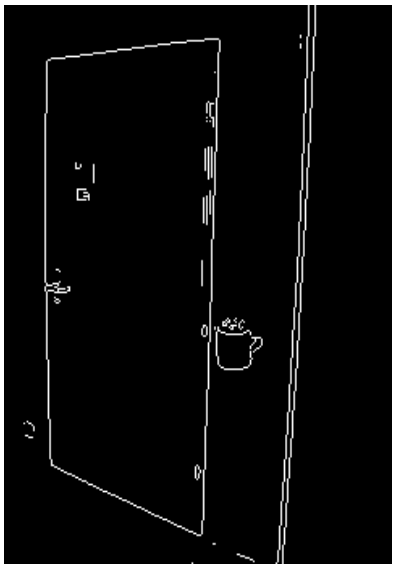
Obscured windscreen

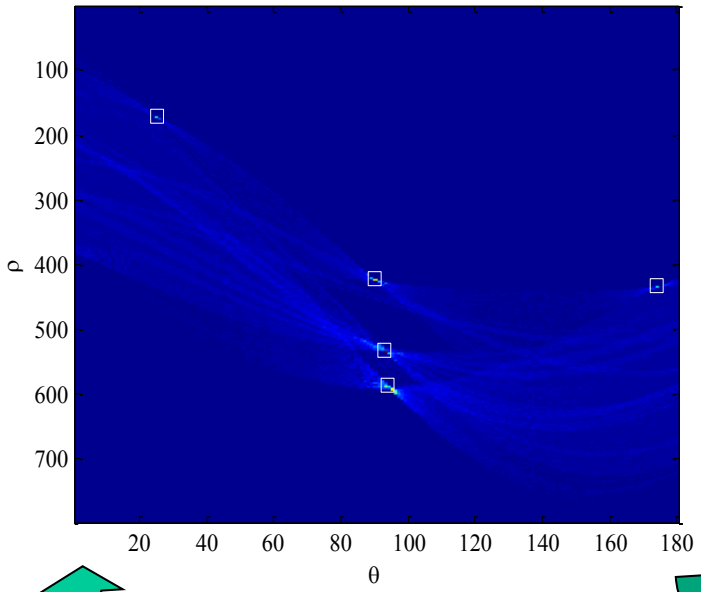


High curvature



Hough Transform





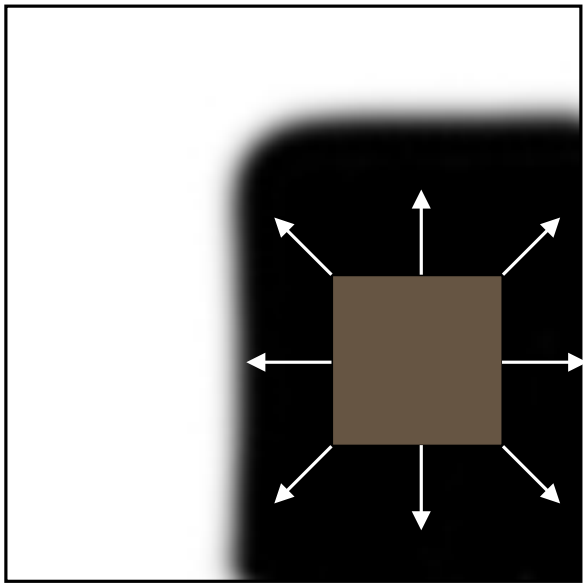
Hough Transform

https://www.youtube.com/watch?v=ebfi7qOFLuo

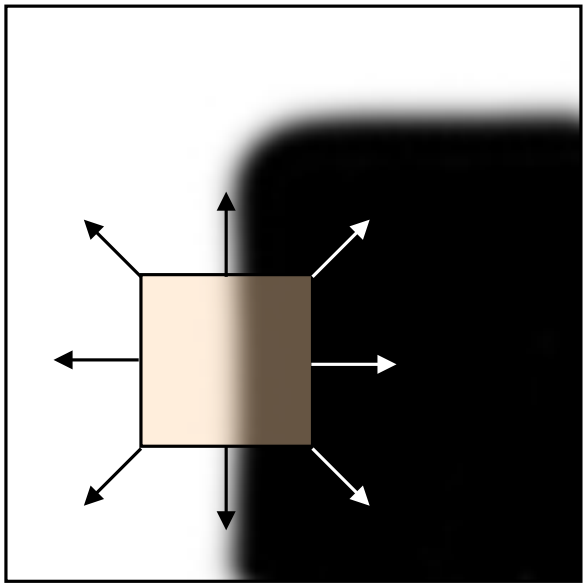
Harris Corner Detector

- The basic idea behind corner detection is that in the region around a corner, the image gradient has two or more dominant directions.
- Corners are repeatable and distinctive.

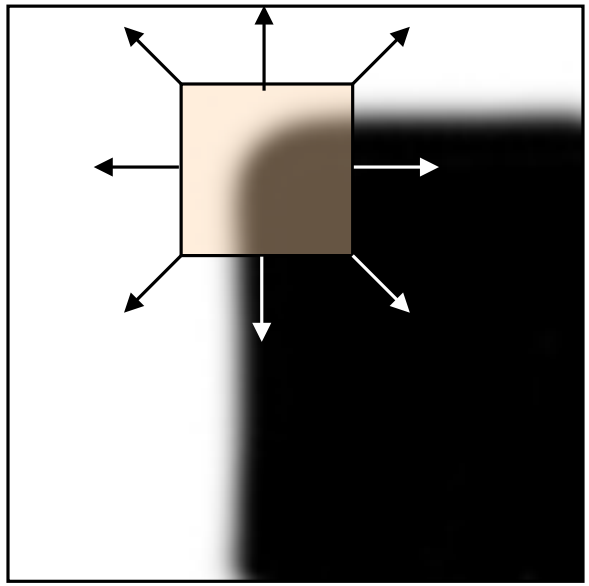
Detecting Corners



"flat" region:
no intensity change



"edge":
no change along the edge direction



"corner":
significant change in at least 2 directions

- Shifting a window in any direction should give a large intensity change in (at least) 2 directions.

Implementation

- Consider two image patches of size P, the first centred at (x, y), and the second centred at $(x + \Delta x, y + \Delta y)$

$$SSD(\Delta x, \Delta y) = \sum_{x,y \in P} (I(x, y) - I(x + \Delta x, y + \Delta y))^2$$

- If we write I_x to represent the partial derivative of $I(x, y)$ in the x direction, and $I_y = \frac{\partial I(x,y)}{\partial y}$, then we can approximate $I(x + \Delta x, y + \Delta y)$ via a Taylor expansion:

$$I(x + \Delta x, y + \Delta y) \approx I(x, y) + I_x(x, y)\Delta x + I_y \Delta y$$

- Substituting into the SSD yields

$$SSD(\Delta x, \Delta y) \approx \sum_{x,y \in P} (I_x(x, y)\Delta x + I_y \Delta y)^2$$

Matrix Form

- We can write this in matrix form as

$$\begin{bmatrix}\Delta x & \Delta y\end{bmatrix} M \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$$

$$M = \sum_{x,y \in P} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- M is the second moment matrix.
- Note that M is symmetric, so we can rewrite it as

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

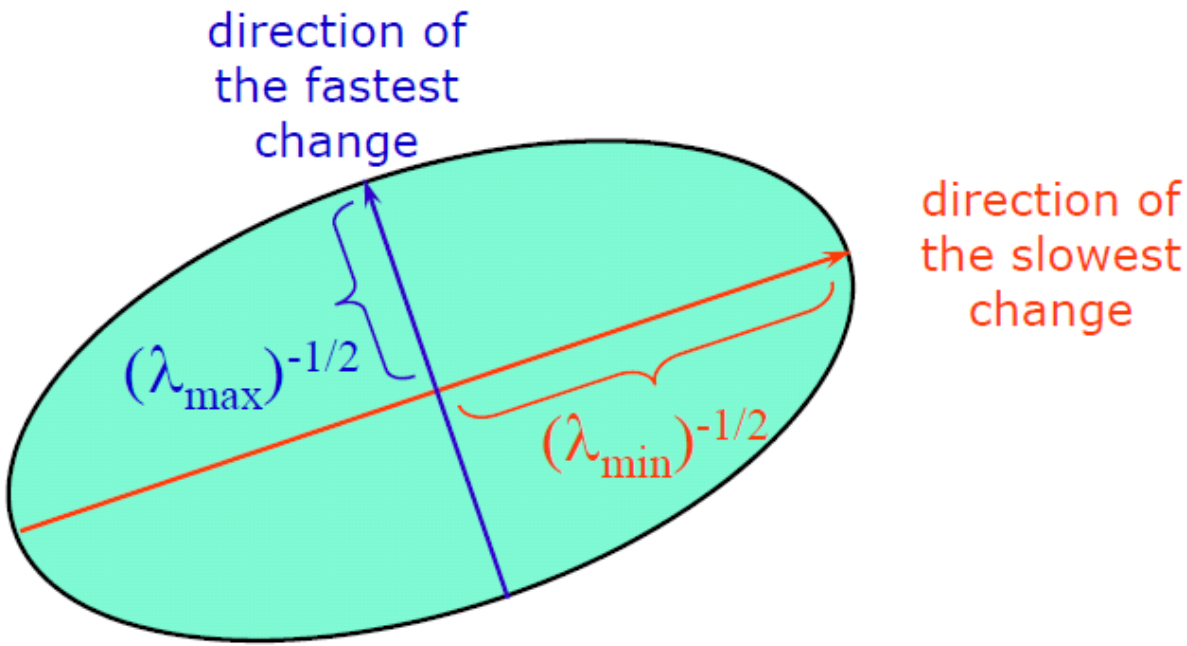
- Where λ_1, λ_2 are the eigenvalues of M (and R its eigenvectors).

Back to corner detection

- The Harris detector examines the eigenvalues to detect a corner by looking for large intensity changes in two (arbitrary) directions.
- M can be thought of as an ellipse with the axis lengths determined by the eigenvalues, and axis determined by R.

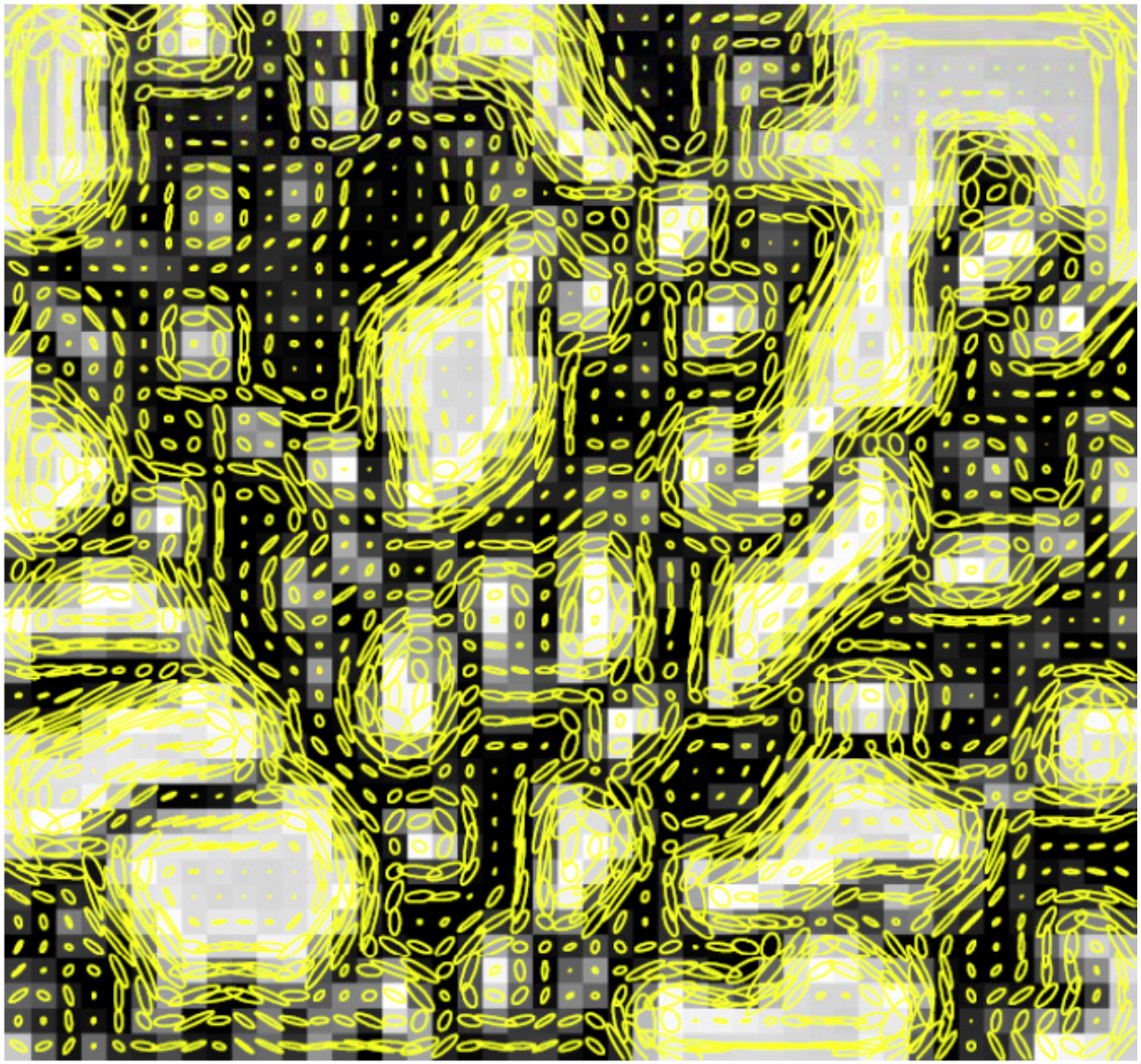
$$\begin{bmatrix}\Delta x & \Delta y\end{bmatrix} M \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix} = const.$$

$$M = R^{-1}\begin{bmatrix}\lambda_1 & 0 \\ 0 & \lambda_2\end{bmatrix}R$$



Visualisation

Visualisation



Corner detection

- We can determine whether P describes a patch by examining its eigenvalues:
  - $\lambda_1 \sim \lambda_2 \sim 0$: there is no structure; the SSD is almost constant in all directions, so it's a flat region.
  - $\lambda_i \sim 0, \lambda_j$ is large: edge; SSD has variation in one direction.
  - λ_1, λ_2 are both large: corner; SSD has large variations in both directions .
- Computing eigenvalues is computationally expensive, a cornerness function was proposed instead:

$$C = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = det(M) - \kappa trace(M)^2$$

- κ is empirically determined, typically between 0.04 and 0.15
- Extract local maxima of the cornerness function.
- Threshold on C and perform nonmaxima supression.

Algorithm

1. Compute x and y derivatives of the image

$$I_x = G_\sigma^x * I \qquad I_y = G_\sigma^y * I$$

2. Compute products of derivative for every pixel

$$I_x^2 = I_x I_x \qquad I_y^2 = I_y I_y \qquad I_{xy} = I_x I_y$$

3. Compute the sum of products of derivatives at each pixel

$$S_x^2 = G_{\sigma'} * I_x^2 \qquad S_y^2 = G_{\sigma'} * I_y^2 \qquad S_{xy} = G_{\sigma'} * I_{xy}$$

4. Compute the H matrix for every pixel

$$H_{xy} = \begin{bmatrix} S_x^2 & S_{xy} \\ S_{xy} & S_y^2 \end{bmatrix}$$

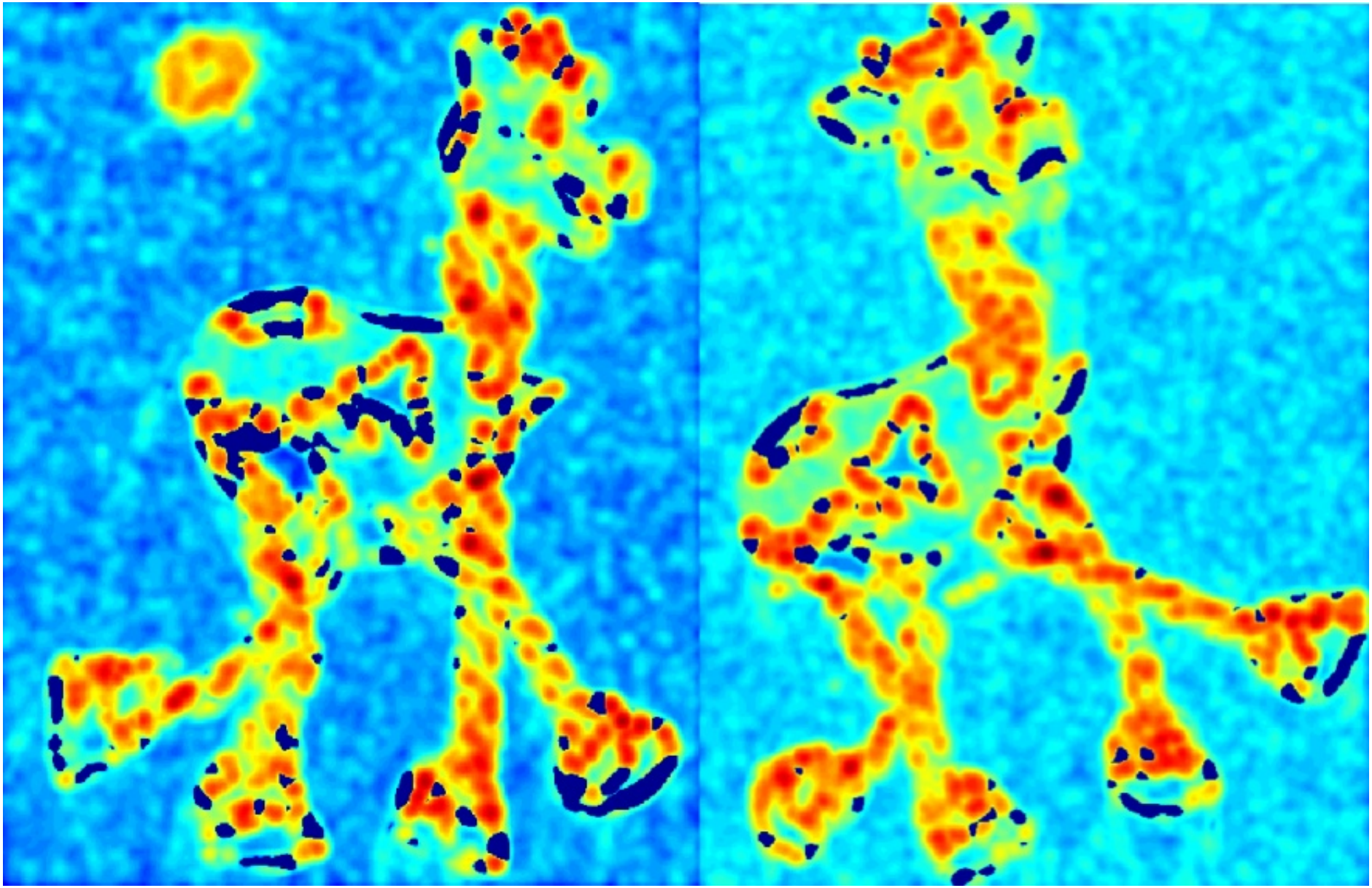
5. Compute the response of each pixel

$$R = det(H) - \kappa(Trace(H))^2$$

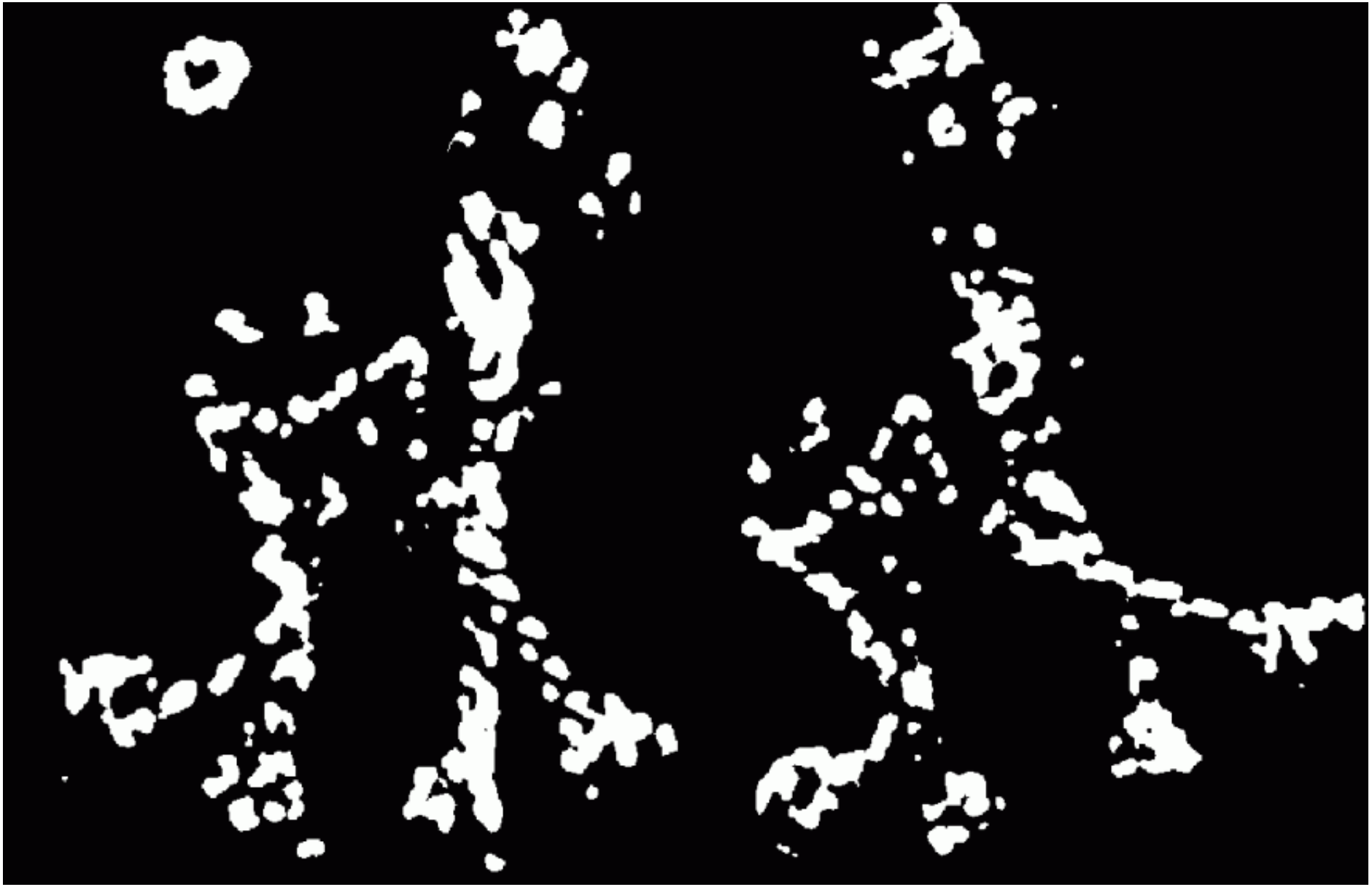
6. Threshold on R
7. Perform nonmaxima supression

Workflow

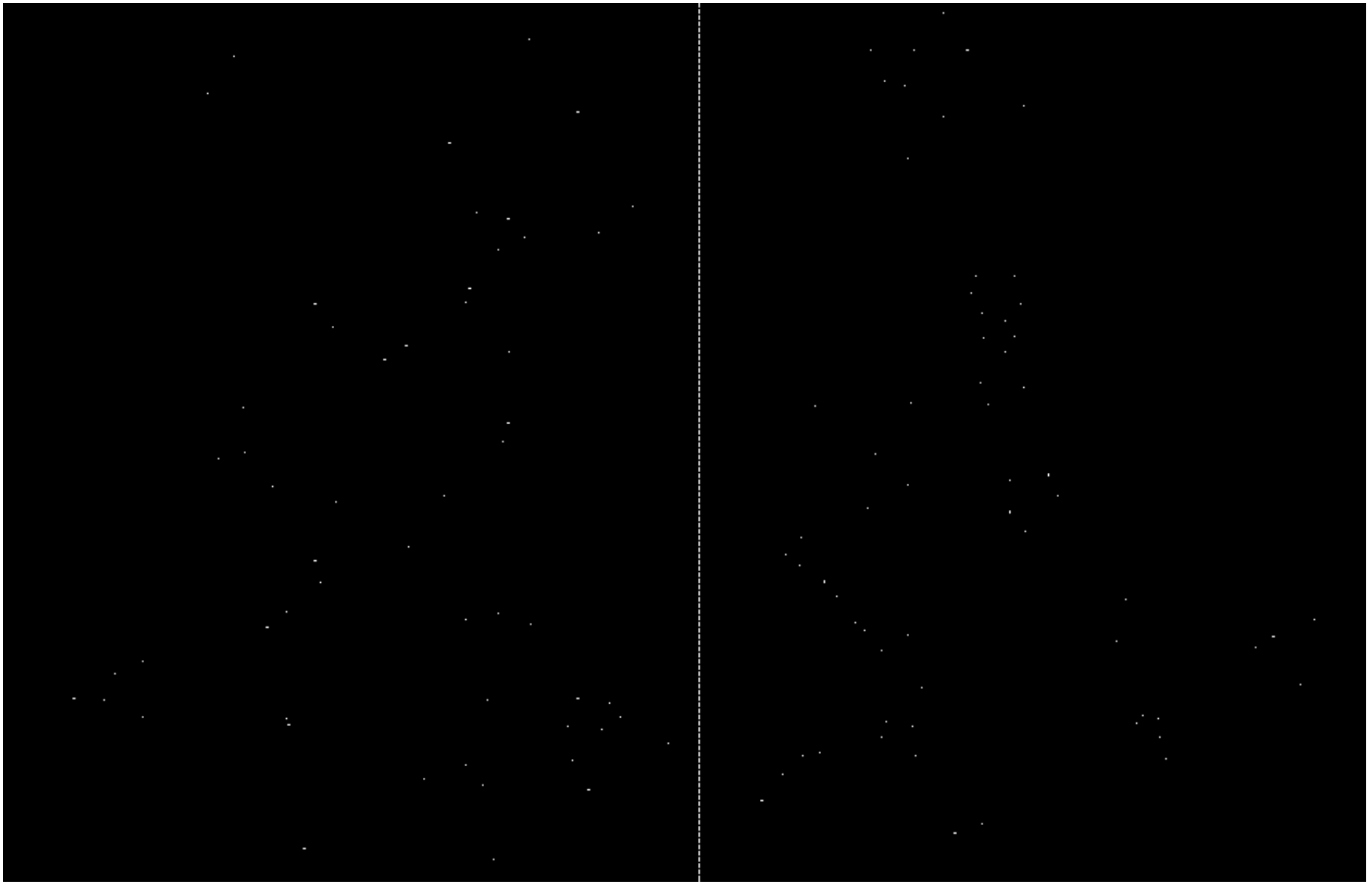
Workflow



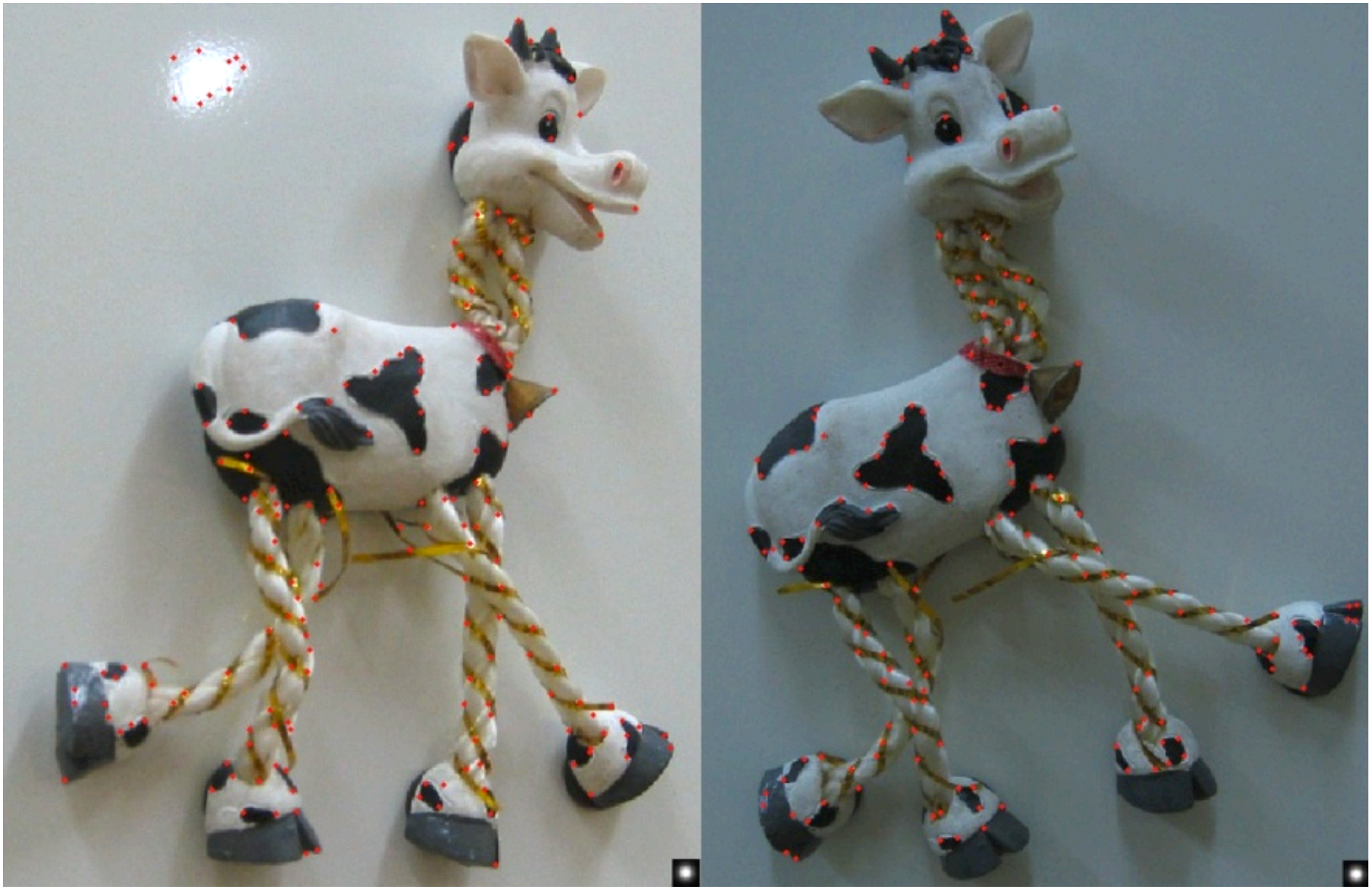
Workflow



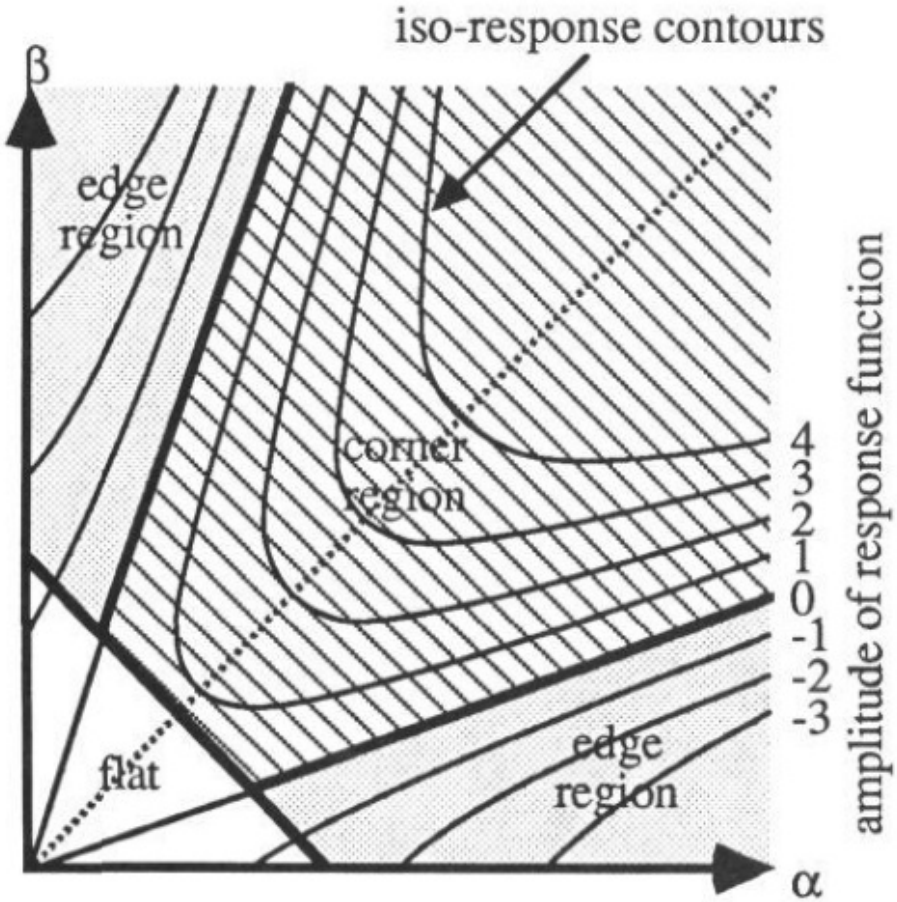
Workflow

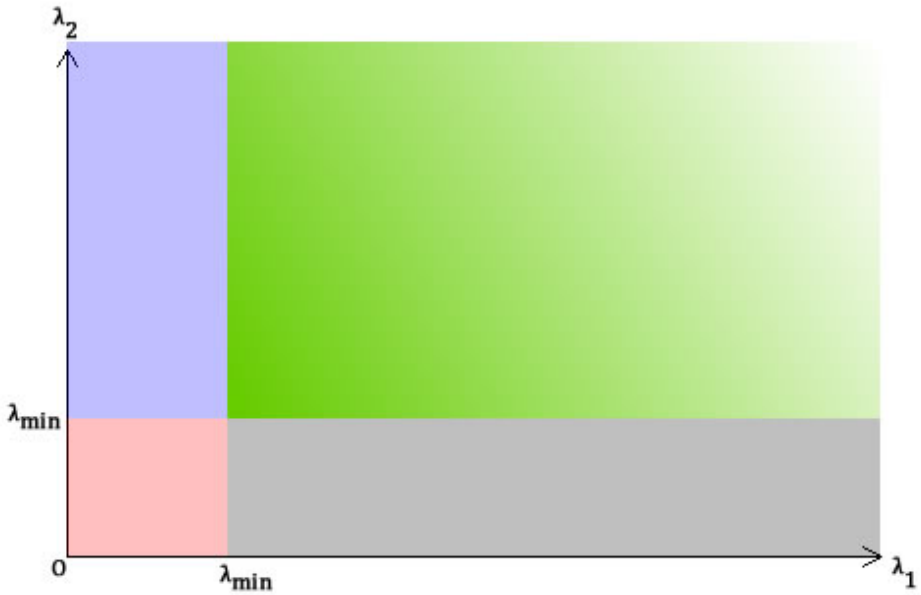


Workflow



Shi-Tomasi Corner Detection



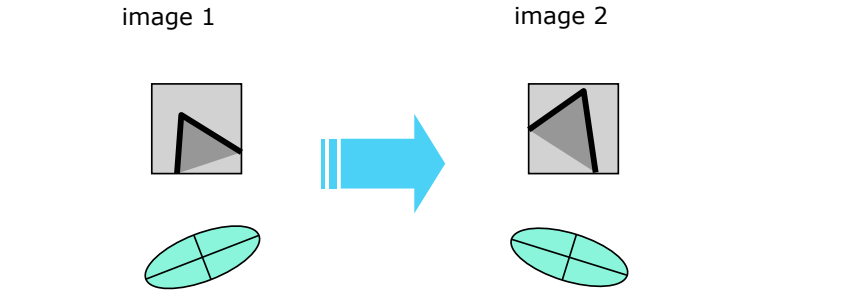


$$R = min(\lambda_1, \lambda_2)$$

$$R = det(H) - \kappa(Trace(H))^2$$

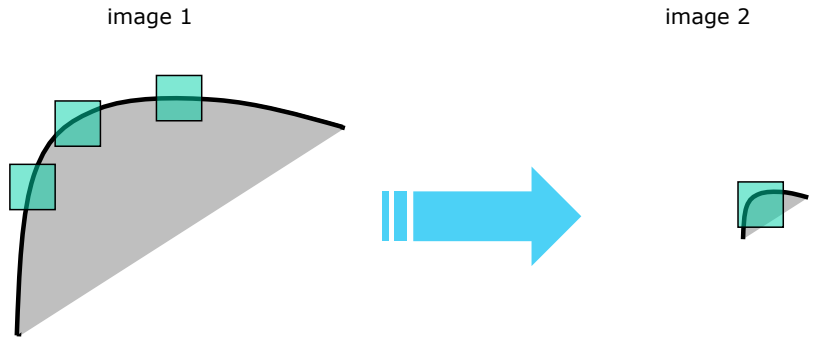
Properties

- Harris corner detector is invariant to linear intensity changes.
- Harris corner detector is invariant to image rotations.



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

- Not invariant to image scale



All points will be classified as **edges**

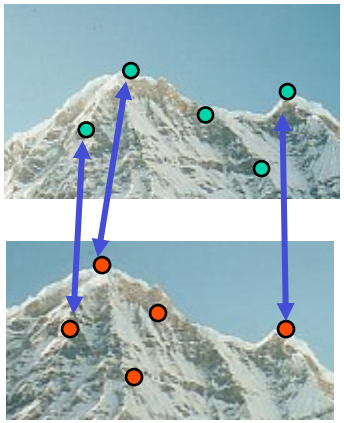
Corner!

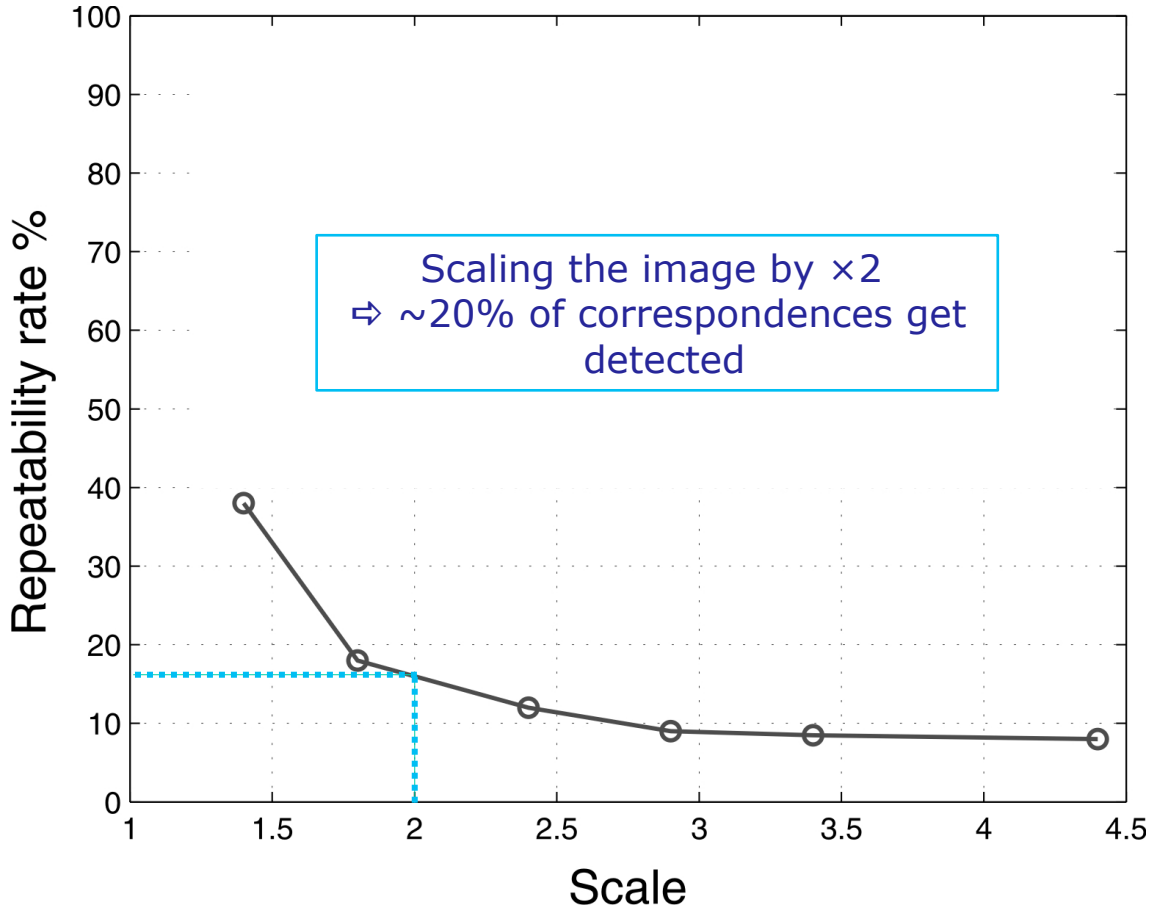
- Not invariant to geometric affine changes — ones that distort neighbourhood of the corner.

Properties

Repeatability:

$$\frac{\text{\# correspondences detected}}{\text{\# correspondences present}}$$



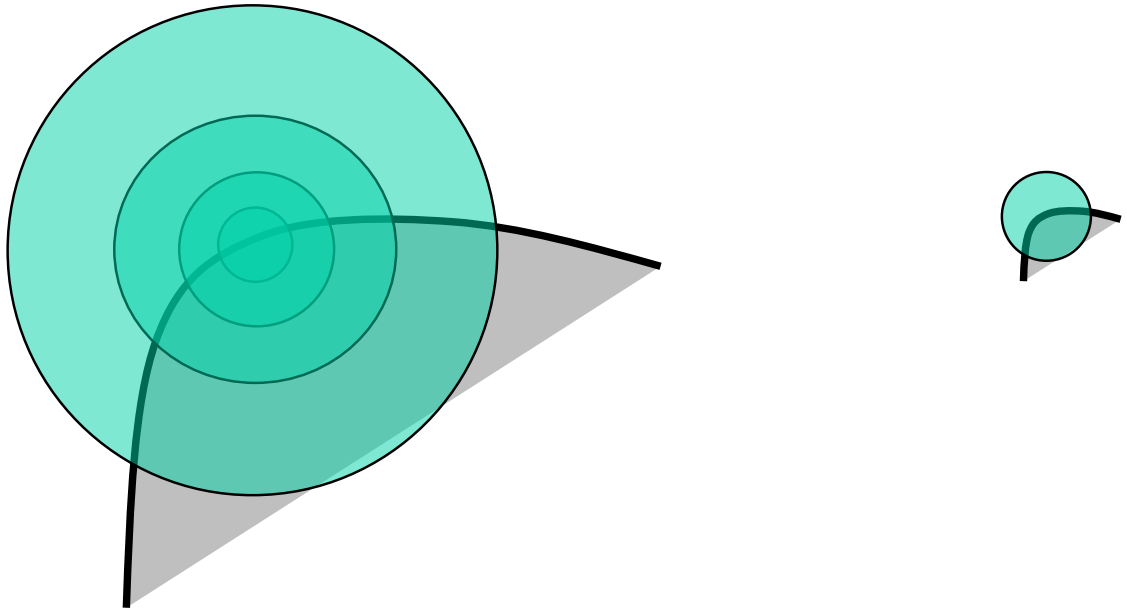


Applications

- https://www.youtube.com/watch?v=r99TJuJqJus

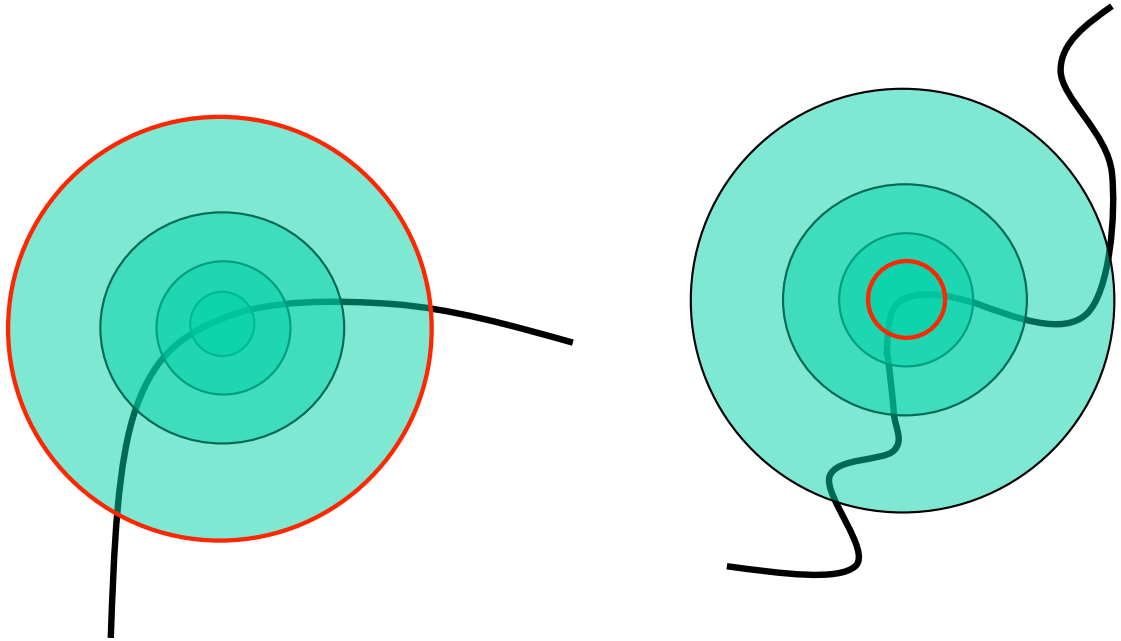
Scale Invariance

- When a Harris corner detector is used, the first step is normally to perform a Gaussian blur over the image.
- This makes our corner detector kernel circular rather than rectangular.
- To enable scale invariance, we seek a kernel size that makes corresponding regions look the same, irrespective of scaling.
- The problem revolves around identifying corresponding regions independently in each image.



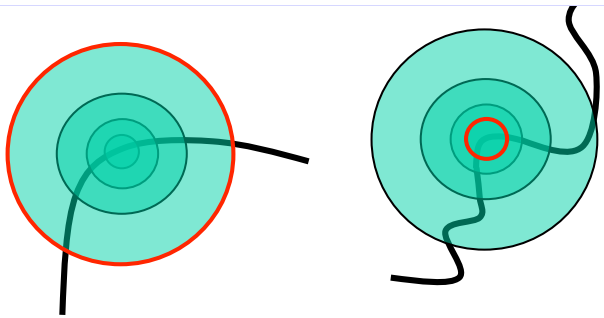
Scale Invariance

- When a Harris corner detector is used, the first step is normally to perform a Gaussian blur over the image.
- This makes our corner detector kernel circular rather than rectangular.
- To enable scale invariance, we seek a kernel size that makes corresponding regions look the same, irrespective of scaling.
- The problem revolves around identifying corresponding regions independently in each image.

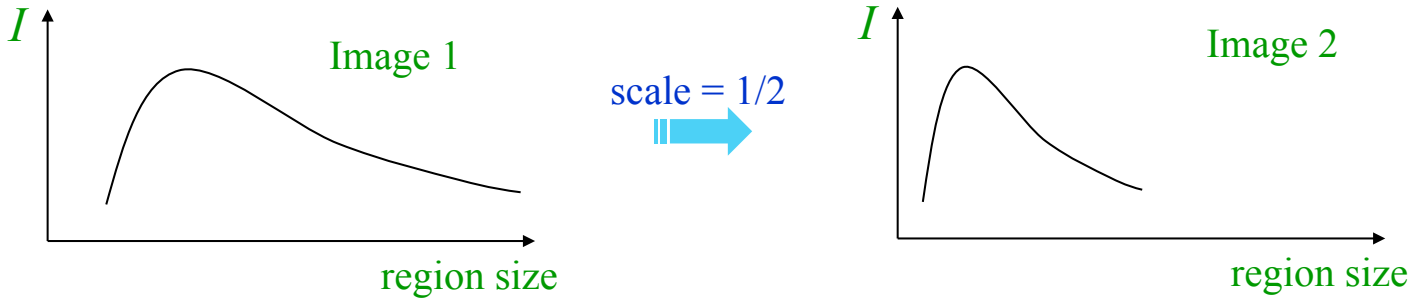


Approach

- Design a function to apply to the region which is scale invariant. In other words, it remains constant for corresponding regions, even if they are at different scales.
- Example: average image intensity over corresponding regions should remain constant, regardless of size.



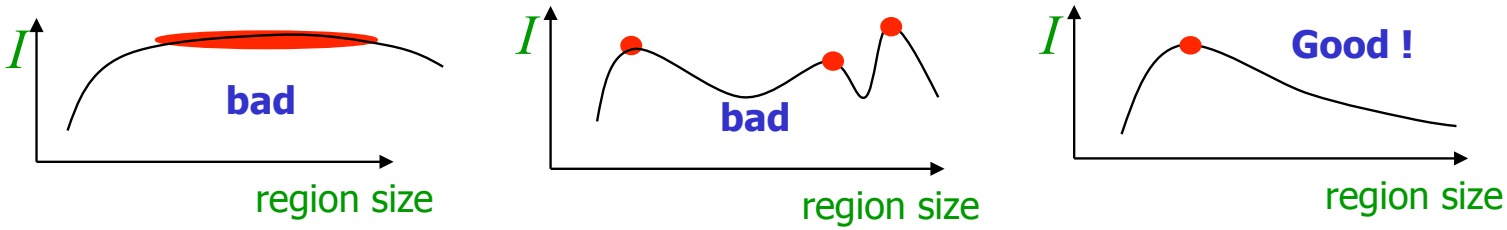
- Average intensity value enclosed in each disc, as a function of the disc-radius:



- The local maxima for each graph occur at corresponding region sizes, independently of other images.

Scale detection functions

- A good function for scale detection has a single clear sharp peak



- Intensity is a bad scale detection function.
- Sharp, local intensity changes in an image are good regions to monitor for identifying relative scale in typical images.

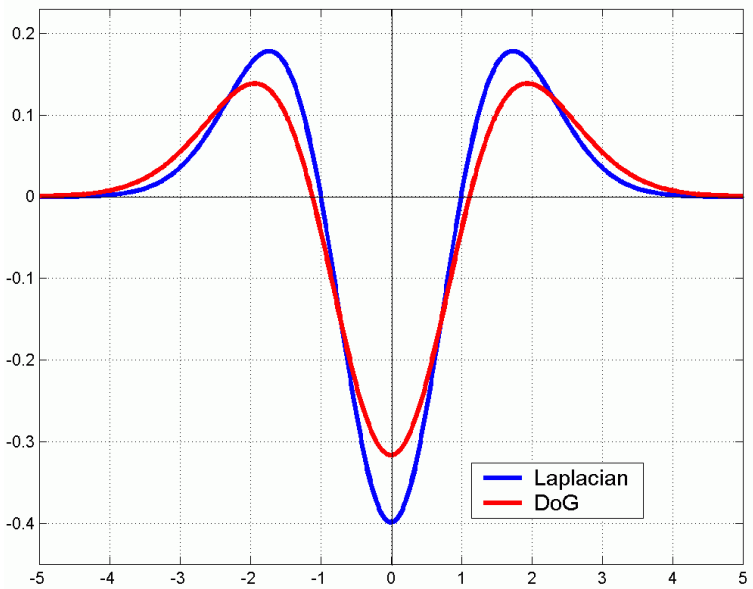
Scale determining functions

- Image is convolved with kernel to identify sharp intensity discontinuities.
- Typical kernels are the Laplacian of Gaussian

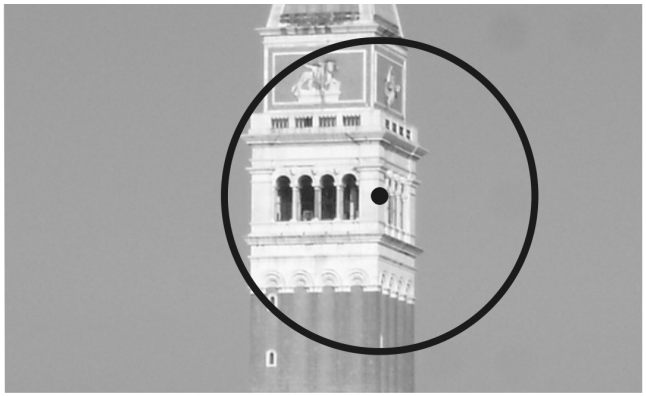
$$\nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}$$

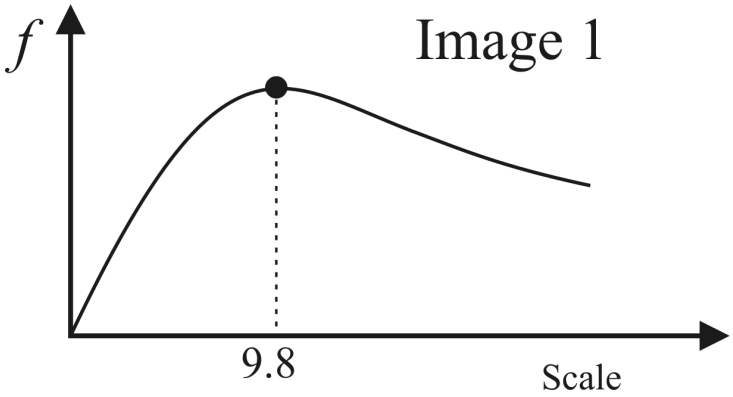
- Difference of Gaussians

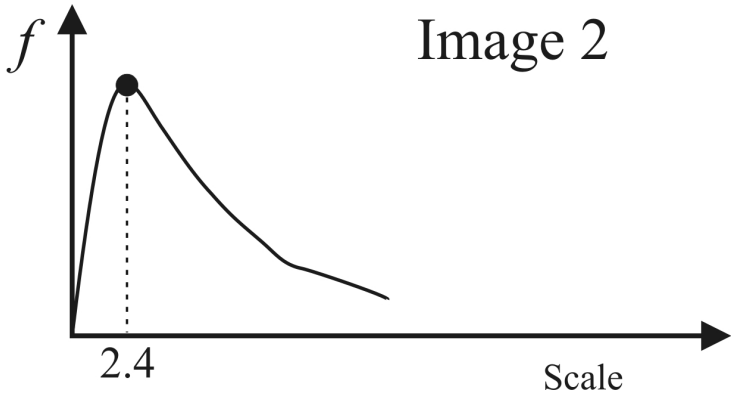
$$G_{k\sigma}(x,y) - G_{\sigma}(x,y)$$



Results







Harris-Laplace Detector

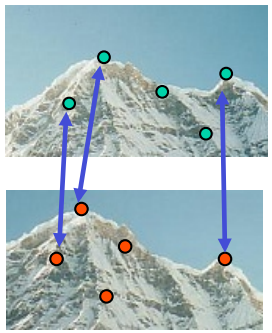
1. Let initial scale be s^0
2. Apply corner detection to identify a corner x^0
3. Find a new scale by running LoG or DoG at point x^0 to obtain a scale s^1
4. Reapply corner detection at this scale to obtain x^1
5. Apply corner detection
6. If scale or corner point changes, repeat algorithm

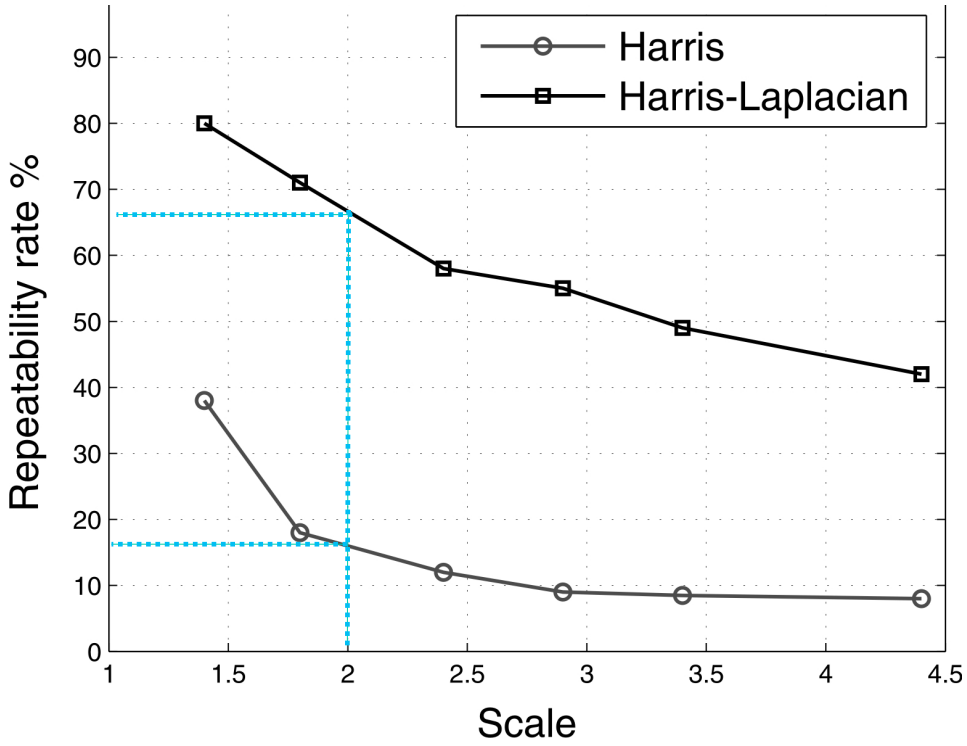
- Algorithm should converge to a unique "best" scale across multiple images.
- Note: rather than scaling kernel size, one normally scales the image itself.

Results

Repeatability:

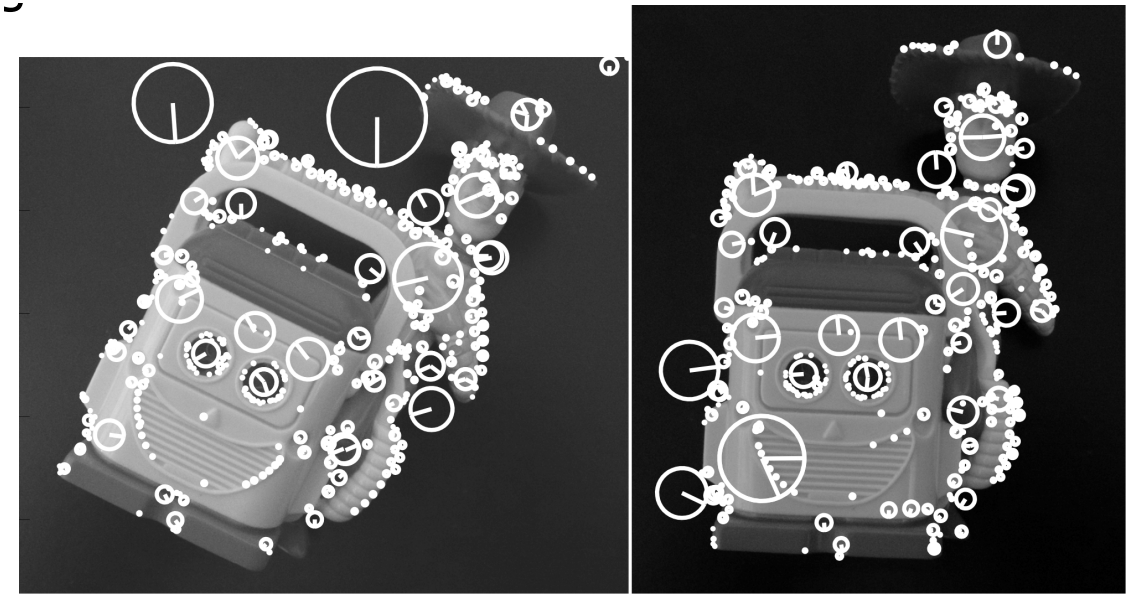
$$\frac{\text{\# correspondences detected}}{\text{\# correspondences present}}$$





SIFT features

- SIFT: Scale Invariant Feature Transform — aims to detect and describe regions of interest in an image.
- SIFT features attempt to be invariant to changes in rotation; scaling; small changes in viewpoint and illumination.
- Excellent in capturing distinctive structures, but computationally demanding.

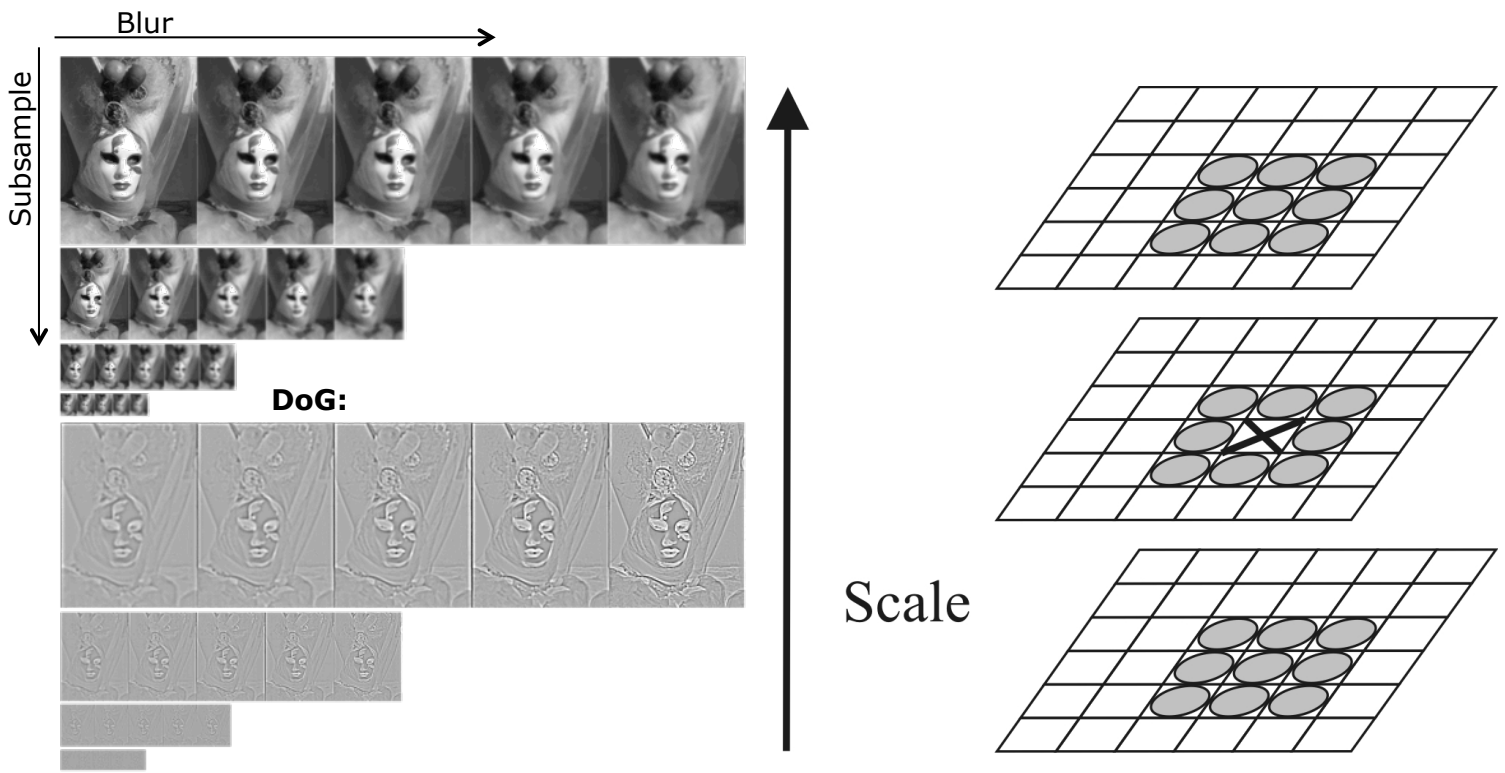


Lowe, D.G. "Distinctive Image Features from Scale-Invariant Keypoints", International Journal of Computer Vision 60, 2, pp. 91-110, 2004

SIFT stages

- Extract key points and scale
- Assign keypoint orientation
- Generate keypoint descriptors

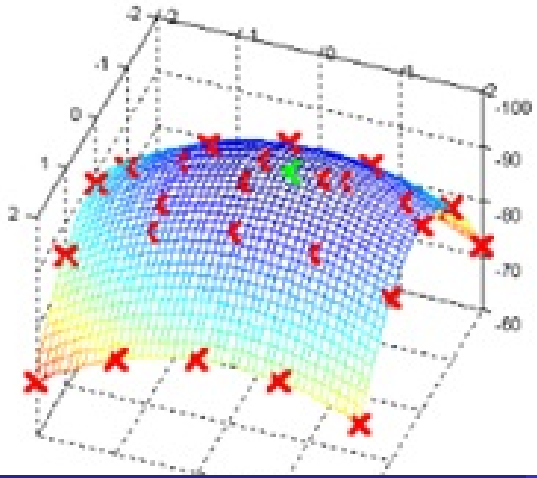
Keypoint extraction and scaling



- Typical use is 4 octaves and 5 blur levels.
- Blur increases by $\sqrt{(2)}$ between images. Octaves are double the size each time.
- Idea is to detect features in different octaves (image scalings) and blur levels (feature sizes)

DoG

- Laplacian detects sharp changes in the image.
- We utilise Difference of Gaussians (DoG) to approximate Laplacian.
- DoG is calculated between different blur levels.
- And for each octave.
- Maxima and minima are detected within a DoG, and between DoGs at different levels.
- First and last level are ignored.
- Maxima/minima is therefore computed among 26 neighbours.
- Exact position of a maximum/minimum should be determined via interpolation to sub-pixel precision.

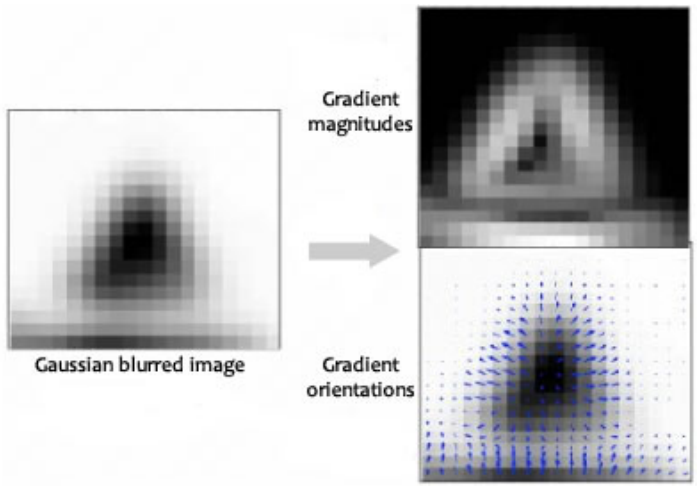


Edge and Low Contrast Elimination

- Features with low contrast or on edges are typically not invariant, and so should be removed.
- Low contrast features are eliminated if the value of the DoG pixel is less than a threshold.
- We run a corner detector to ensure that our features are "corner only".
- We now have a set of scale invariant keypoints.

Orientation

- For a given key point we compute gradient magnitudes and orientations around the keypoint (at the relevant scale and octave levels)



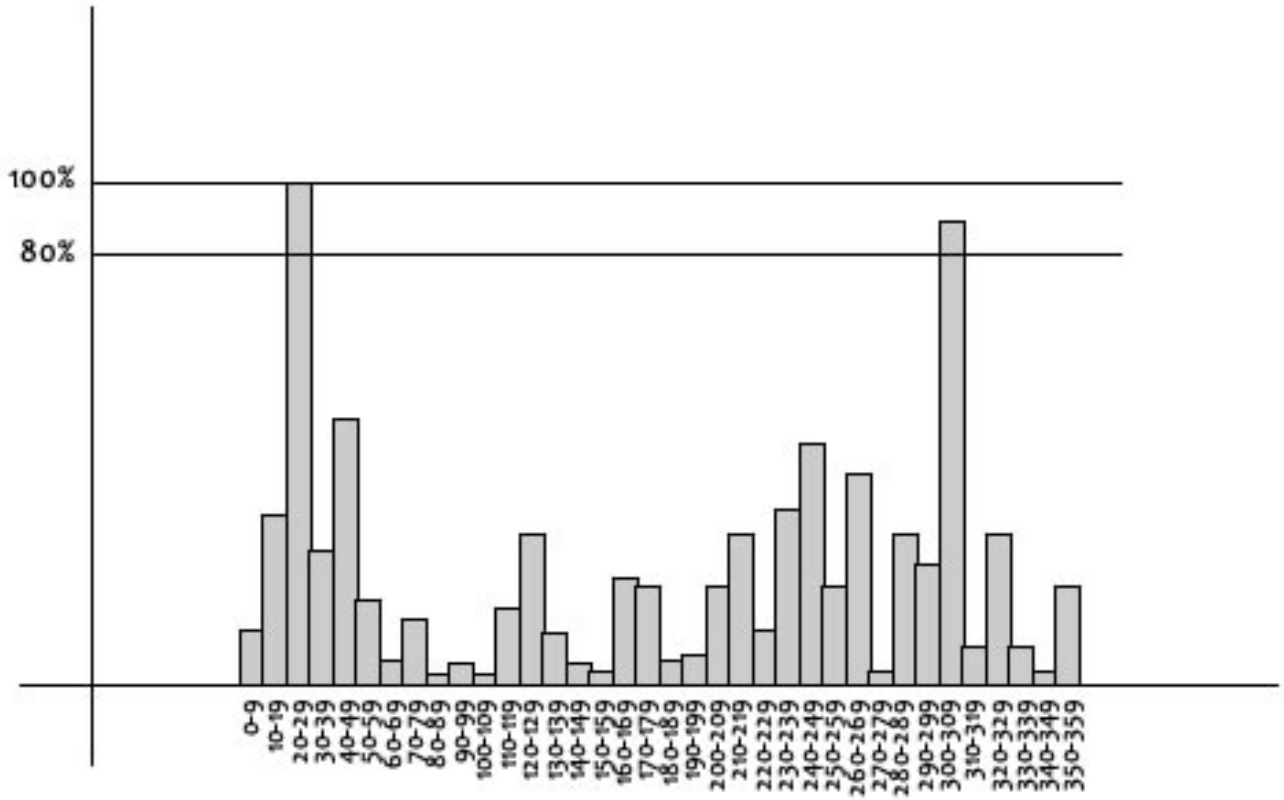
$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = arctan((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

- These are binned into a 36 bin histogram (1 bin for every 10 degrees).

Orientation and Histograms

- Peak of the histogram identifies feature direction.
- Large peaks are separated into a separate key point.



Fingerprint Generation

- We take a 16×16 pixel window around the key point. Split it into 16 4×4 windows.
- Calculate gradient magnitude and orientations for each such window, binning into an 8 bin histogram.
- Scale by distance from key point.
- We now have 8 numbers per 4×4 window, and we have 16 of these windows: 128 numbers.
- These are normalised, and orientation rotated with regards to the feature's orientation.
- The 128 numbers can be thresholded to provide illumination independence (typically using an upper limit of 0.2).
- Each feature is now associated with a rotation and illumination independent 128 entry vector.

So what?

- http://www.youtube.com/watch?v=zvHPloCxxfE
- http://www.youtube.com/watch?v=r9cgKC_-FF4
- http://www.youtube.com/watch?v=1GhNXHCQGsM
- We can recognise objects (even when partially hidden) and places.
- http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/

Location Detection

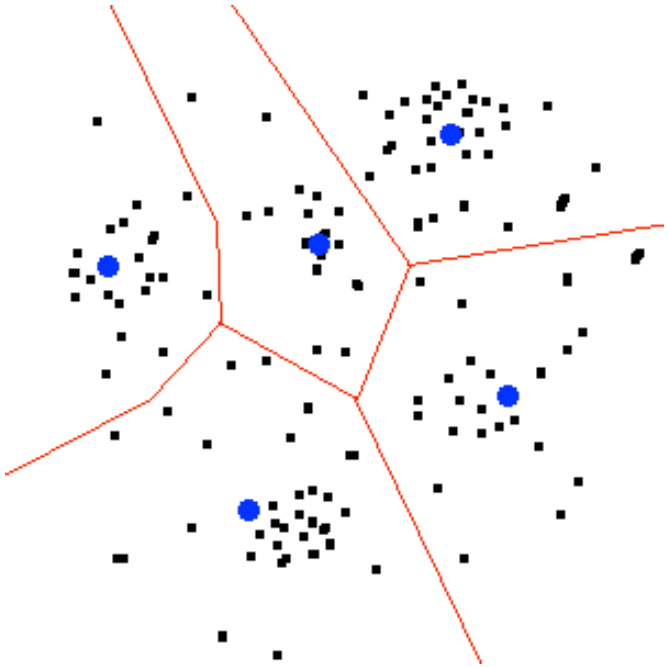
- We aim to identify discrete places in the world.
- SIFT features provide is with a bag of features — independent of spatial relation between interest points as only descriptors are provided.
- If we have two sets of descriptors (one from the current location, one from training data), we can compute similarity by counting the number of common feature descriptors.
- This requires a matching function.
- L_2 norm allows us to compute distance between high dimensional vectors.

Visual Words

- Visual words are 1-D representations of high dimensional feature descriptors.
- Conversion to visual words yields a bag of visual words.
- Basic approach is through k-means clustering.

k-means clustering

- Aims to partition a set of d-dimensional vectors into k sets.
- Minimises the within-cluster sum of squares.
- The visual word for a feature descriptor is the number of the cluster it belongs to.
- Observations are partitioned according to the Voronoi diagram generated by the means.



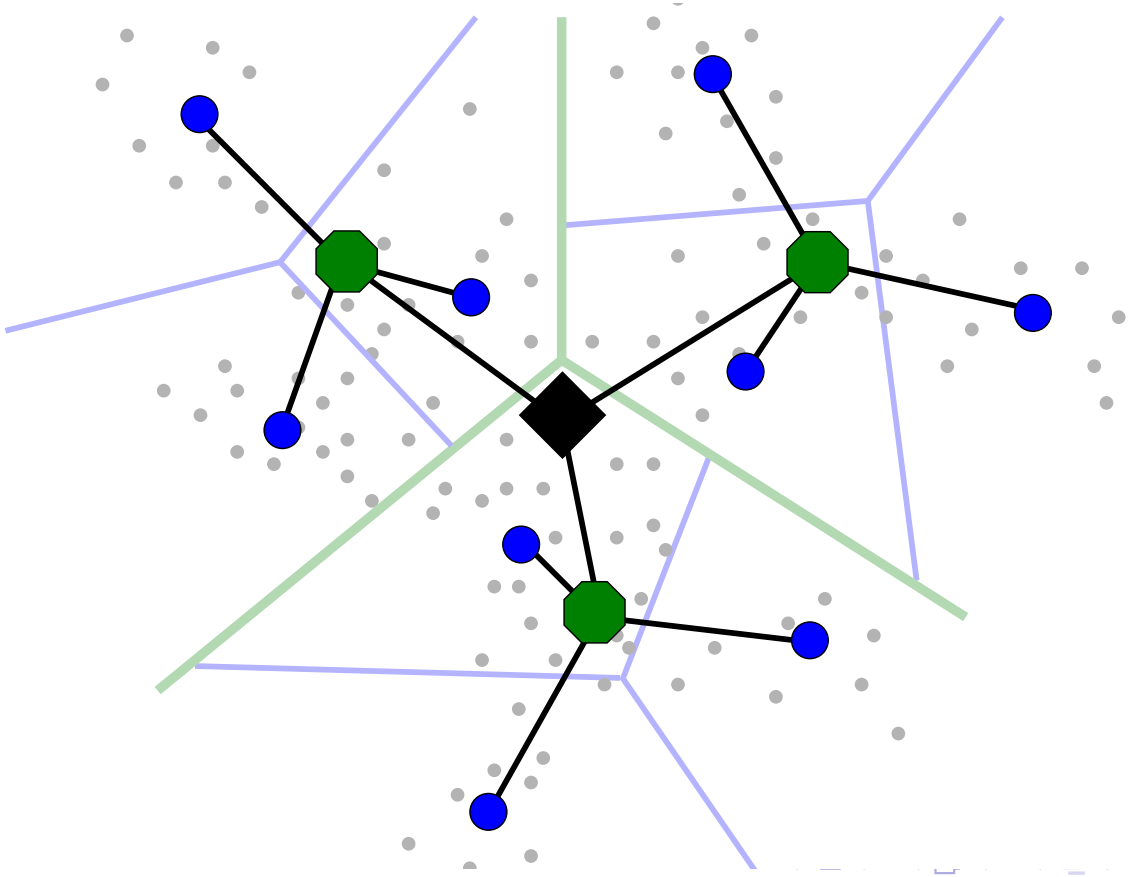
k-means clustering algorithm

Require: A set of vectors $X = \{x_1, \ldots, x_i\}$

let C be a list of clusters
Randomly pick k random unique elements from X as initial means $M = [m_1, \ldots, m_k]$
finished=false
while not **finished do**
    for all $x \in X$ **do**
        Assign x to cluster c_j where m_j is the nearest mean to x
    end for
    for all $c_i \in C$ **do**
        let m_i be the centroid of cluster c_i
    end for
    If no changes in cluster membership took place **then finished**=true
end while
return C

Retrieval

- Only the means need to be stored, rather than all feature vectors.
- A feature word is assigned by identifying the cluster it belongs to.
- This can still be expensive if a large number of clusters (unique features) exist; hierarchical partitioning can be used to reduce this cost.



Efficiency

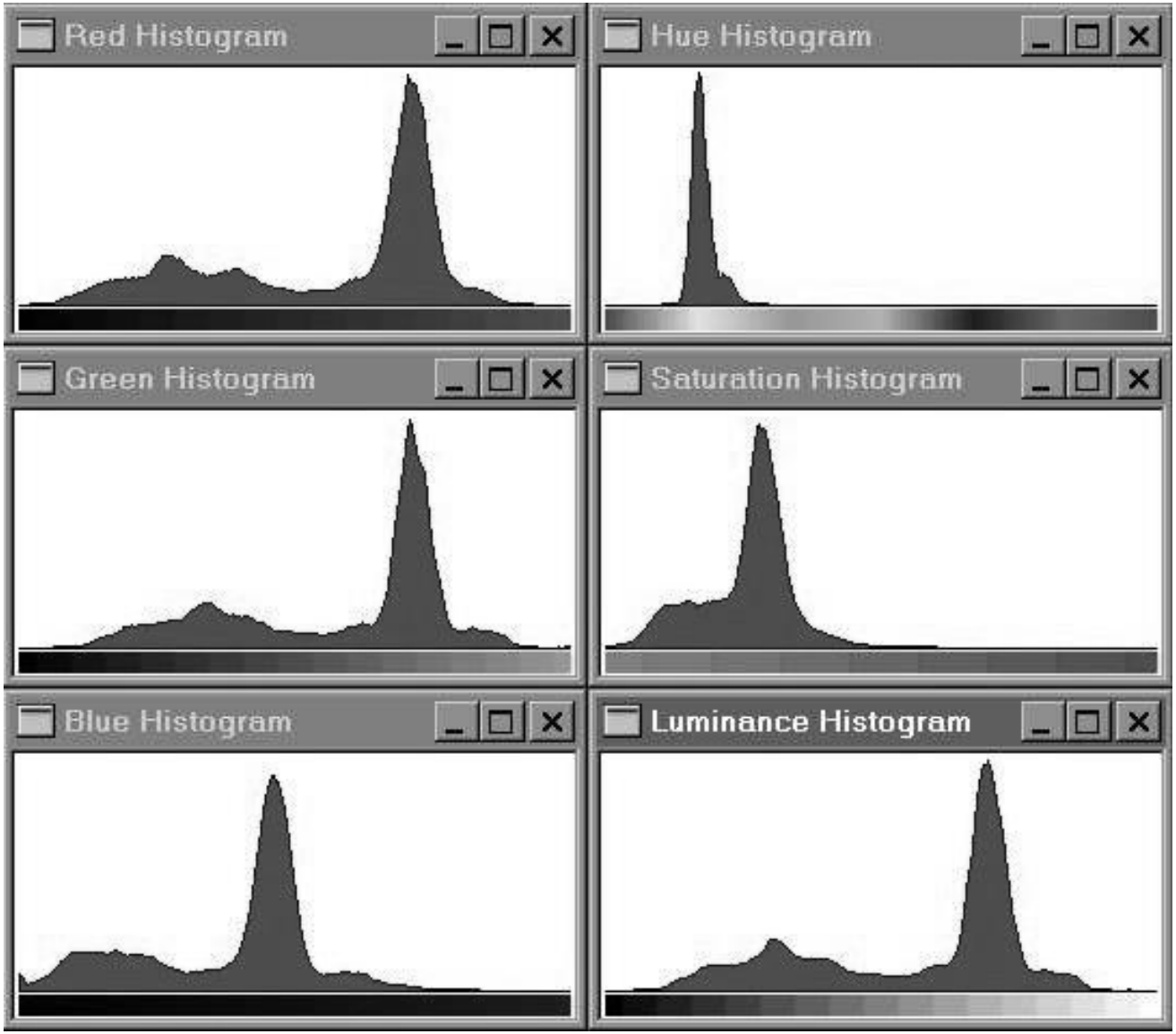
- We can still have a huge number of clusters (one per feature "type").
- An inverted file and voting scheme can help efficiency.
- Inverted file:
  - The index is a list of possible visual words
  - Key is a list of all image identifiers in which this visual word appears
- Voting scheme:
  - Initialise voting array with as many cells as images in the database.
  - Obtain visual word from query, add 1 to the list of image identifiers attached to this visual word.
- Algorithm returns most similar image (with most votes), and also ranking of all images in the database at no additional cost.
- N most similar images can be tested for geometric consistency using x and y coordinates of matching visual words.

Global Place Recognition

- The approach above used local features.
- A image histogram based approach identifies similar locations by seeking a similar histogram.
- Under the same lighting conditions, after smoothing, small changes in robot position should yield similar histograms.
- If a 360 degree image is used, then this approach is rotation invariant.
- Typically, separate histograms are built for r, g, b and hue, saturation and luminance values.
- Histograms are preprocessed using a Guassian smoothing operator.
- Each pixel is then put into the appropriate bin for each band.
- Jeffery divergence is used to compute difference between histograms H and K

$$d(H,K) = \sum_i (h_i \log \frac{2h_i}{h_i + h_i} + h_i \log \frac{2k_i}{h_i + k_i})$$

Histograms


Red Histogram
Hue Histogram
Green Histogram
Saturation Histogram
Blue Histogram
Luminance Histogram

Why is this useful?

- Object detection
- Location detection is useful in SLAM — knowing if you have visited a place before
- Identifying specific features of an area