Practical Course:
Vision-based Navigation
Winter Term 2016/2017

Lecture 1: Basics

Vladyslav Usenko, Benjamin Grzimek, Lukas von Stumberg,
Prof. Dr. Daniel Cremers

What we will cover today

- Linear algebra, notation

- 3D geometry

- Projective geometry
    - Camera intrinsics
    - Epipolar geometry

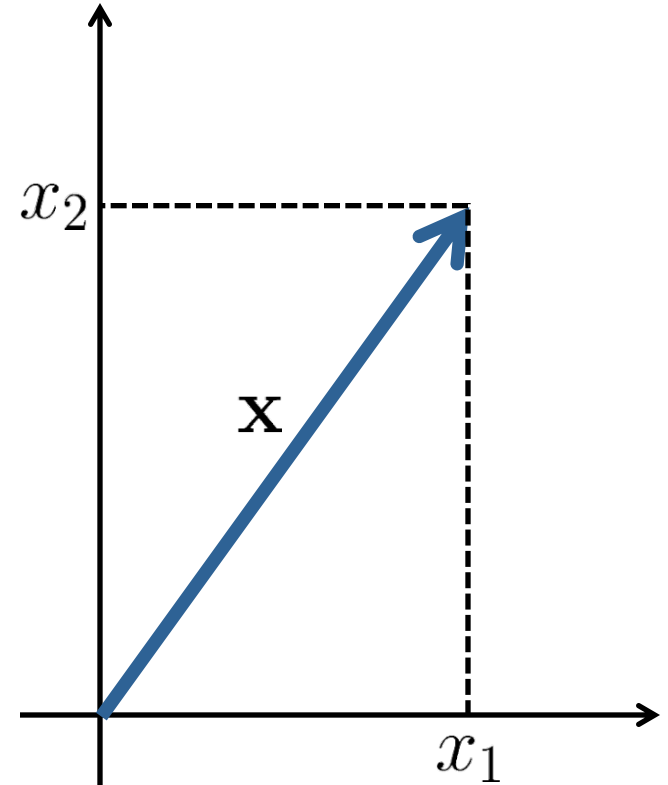- Robot Operating System (ROS)

What we will cover today

- **Linear algebra, notation**
- 3D geometry
- Projective geometry
  - Camera intrinsics
  - Epipolar geometry

- Robot Operating System (ROS)
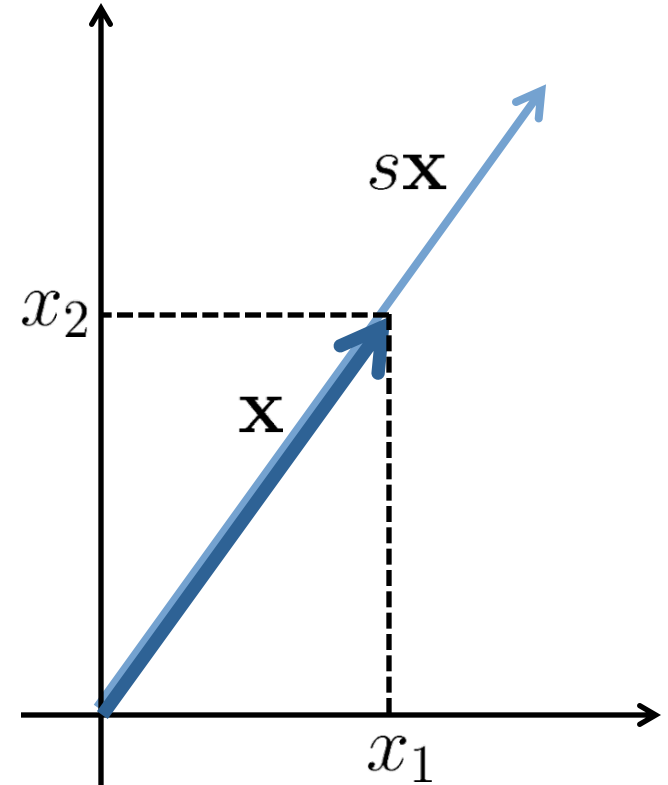
Vectors

- Vector and its coordinates

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$$

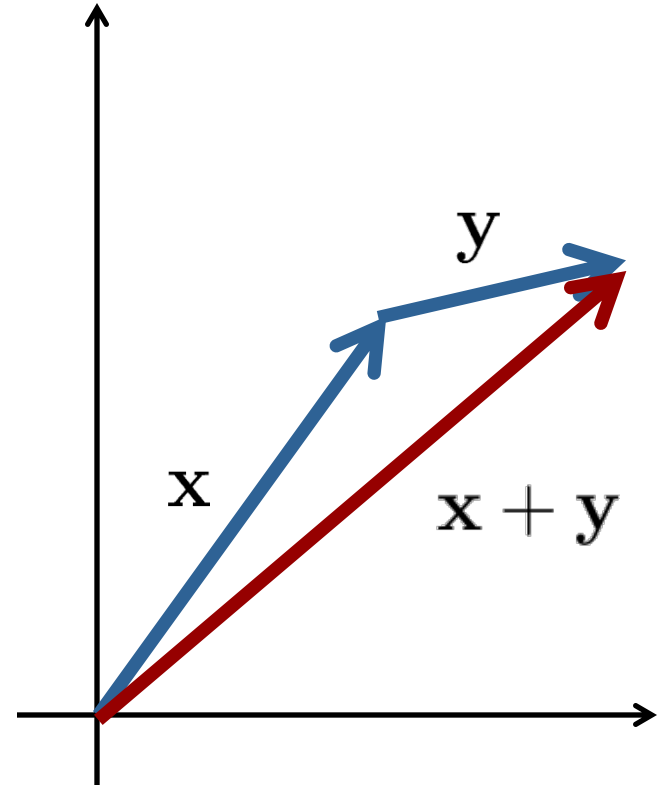- Vectors represent points in an n-dimensional space

Vector Operations

- **Scalar multiplication**
- Addition/subtraction
- Length
- Normalized vector
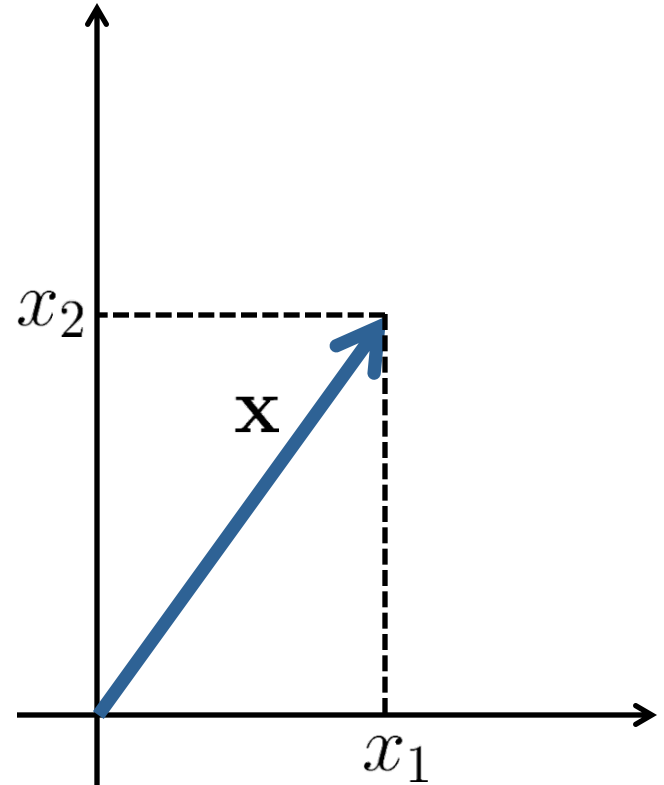- Dot product
- Cross product

Vector Operations

- Scalar multiplication
- **Addition/subtraction**
- Length
- Normalized vector
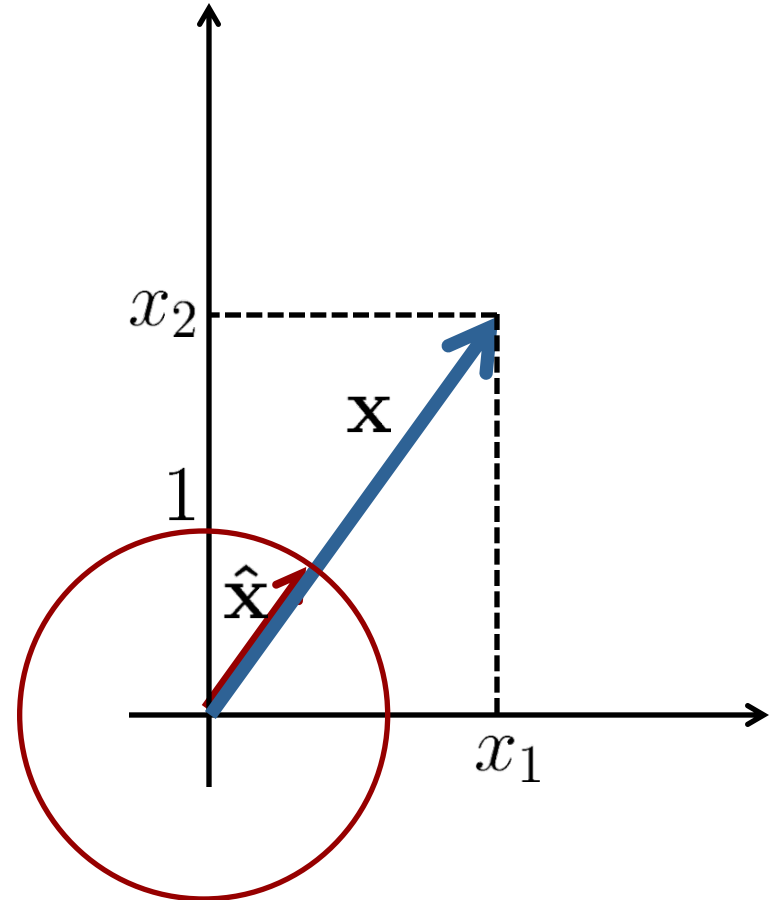- Dot product
- Cross product

Vector Operations

- Scalar multiplication
- Addition/subtraction
- **Length**
- Normalized vector
- Dot product
- Cross product

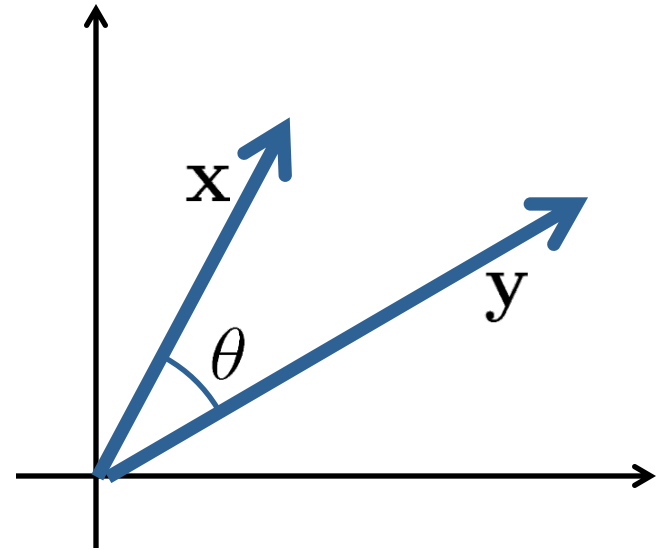$$\|x\|_2 = \|x\| = \sqrt{x_1^2 + x_2^2 + \ldots}$$

Vector Operations

- Scalar multiplication
- Addition/subtraction
- Length
- **Normalized vector**
- Dot product
- Cross product

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$$

Vector Operations

- Scalar multiplication
- Addition/subtraction
- Length
- Normalized vector
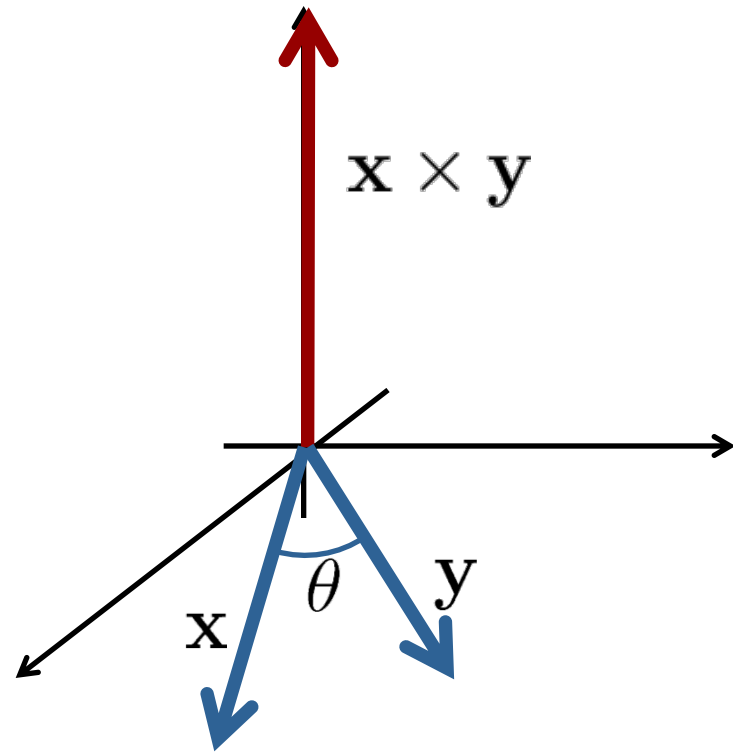- **Dot product**
- Cross product

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos \theta$$

$\mathbf{x}, \mathbf{y}$ are orthogonal if $\quad \mathbf{x} \cdot \mathbf{y} = 0$

$\mathbf{y}$ is lin. dependent from $\quad \{\mathbf{x}_1, \mathbf{x}_2, . \text{if.}\} \quad \mathbf{y} = \sum_i k_i \mathbf{x}_i$

Vector Operations

- Scalar multiplication
- Addition/subtraction
- Length
- Normalized vector
- Dot product
- **Cross product**

$$\mathbf{x} \times \mathbf{y}$$

$$\mathbf{x} \times \mathbf{y} = \|\mathbf{x}\|\|\mathbf{y}\| \sin(\theta)\mathbf{n}$$

Cross Product

- Definition
$$\mathbf{x} \times \mathbf{y} = \begin{pmatrix} x_2 y_3 - x_3 y_2 \\ x_3 y_1 - x_1 y_3 \\ x_1 y_2 - x_2 y_1 \end{pmatrix}$$

- Matrix notation for the cross product
$$[\mathbf{x}]_\times = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}$$

- Verify that
$$\mathbf{x} \times \mathbf{y} = [\mathbf{x}]_\times \mathbf{y}$$

Matrices

- Rectangular array of numbers

$$X = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1m} \\ x_{21} & x_{22} & \ldots & x_{2m} \\ \vdots & & & \\ x_{n1} & x_{n2} & \ldots & x_{nm} \end{pmatrix} \in \mathbb{R}^{n \times m}$$

rows   columns

- First index refers to row
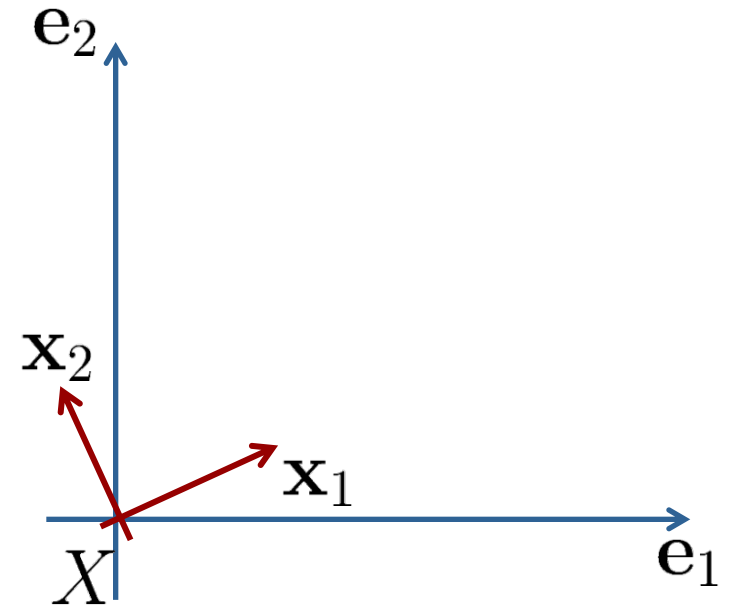- Second index refers to column

Matrices

- Column vectors of a matrix

$$X = \begin{pmatrix} \boxed{\begin{matrix} x_{11} \\ x_{21} \\ \vdots \\ x_{n1} \end{matrix}} & \boxed{\begin{matrix} x_{12} \\ x_{22} \\ \\ x_{n2} \end{matrix}} & \cdots & \boxed{\begin{matrix} x_{1m} \\ x_{2m} \\ \\ x_{nm} \end{matrix}} \end{pmatrix}$$

$$= \begin{pmatrix} \mathbf{x}_{*1} & \mathbf{x}_{*2} & \cdots & \mathbf{x}_{*m} \end{pmatrix}$$

- Geometric interpretation: for example, column vectors can form basis of a coordinate system

Matrices

- Row vectors of a matrix

$$X = \begin{pmatrix} \boxed{x_{11} \quad x_{12} \quad \ldots \quad x_{1m}} \\ \boxed{x_{21} \quad x_{22} \quad \ldots \quad x_{2m}} \\ \vdots \\ \boxed{x_{n1} \quad x_{n2} \quad \ldots \quad x_{nm}} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{1*}^\top \\ \mathbf{x}_{2*}^\top \\ \vdots \\ \mathbf{x}_{n*}^\top \end{pmatrix}$$

Matrices

- Square matrix
- Diagonal matrix
- Upper and lower triangular matrix
- Symmetric matrix
- Skew-symmetric matrix
- (Semi-)positive definite matrix
- Invertible matrix
- Orthonormal matrix
- Matrix rank

Matrices

- Square matrix
- Diagonal matrix
- Upper and lower triangular matrix
- Symmetric matrix $\quad X = X^\top$
- Skew-symmetric matrix
- (Semi-)positive definite matrix
- Invertible matrix
- Orthonormal matrix
- Matrix rank

$$X = -X^\top \left(= \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \right)$$

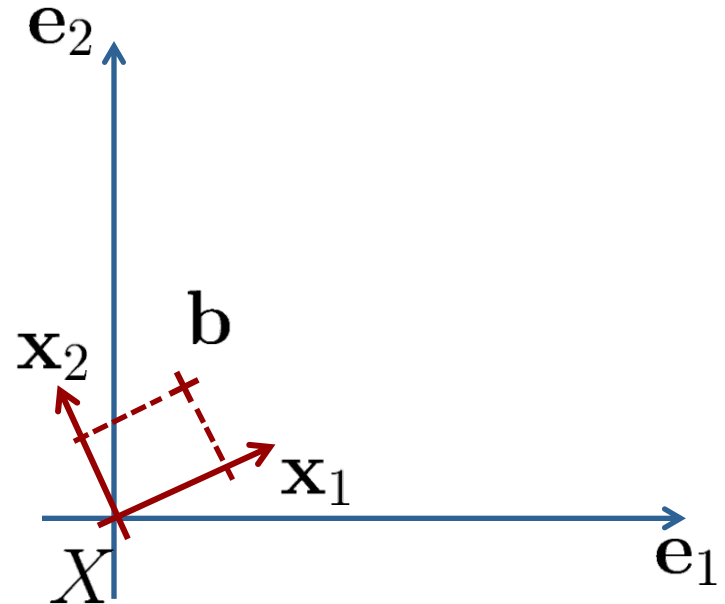$$\mathbf{a}^\top X \mathbf{a} \geq 0$$

Matrix Operations

- Scalar multiplication

- Addition/subtraction

- Transposition

- Matrix-vector multiplication

- Matrix-matrix multiplication

- Inversion

Matrix-Vector Multiplication

$$X \cdot \mathbf{b} = \sum_{k=1}^{n} \mathbf{x}_{*k} \cdot b_k$$

column vectors

- Geometric interpretation:
  A linear combination of the columns of X scaled by the coefficients of $\mathbf{b}$
  $\rightarrow$ coordinate transf. from local to global frame

Matrix-Matrix Multiplication

- Operator $\mathbb{R}^{n \times m} \times \mathbb{R}^{m \times p} \rightarrow \mathbb{R}^{n \times p}$

- Definition
$$C = AB$$
$$= A \begin{pmatrix} \mathbf{b}_{*1} & \mathbf{b}_{*2} & \cdots \mathbf{b}_{*p} \end{pmatrix}$$

- Interpretation: transformation of coordinate systems

- Can be used to concatenate transforms

Matrix-Matrix Multiplication

- Not commutative (in general) $\qquad AB \neq BA$

- Associative $\qquad A(BC) = (AB)C$

- Transpose $\qquad (AB)^\top = B^\top A^\top$

Matrix Inversion

- If A is a square matrix of full rank, then there is a unique matrix $B = A^{-1}$ such that $AB = I$

- Different ways to compute, e.g., Gauss-Jordan elimination, LU decomposition, …

- When A is orthonormal, then $A^{-1} = A^{\top}$

What we will cover today

- Linear algebra, notation
- **3D geometry**
- Projective geometry
    - Camera intrinsics
    - Epipolar geometry

- Robot Operating System (ROS)

Geometric Primitives in 3D

- 3D point
  (same as before)

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$$

- Augmented vector

$$\bar{\mathbf{x}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in \mathbb{R}^4$$

- Homogeneous coordinates

$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{pmatrix} \in \mathbb{P}^3$$
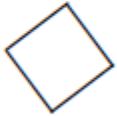
3D Transformations
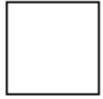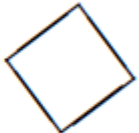
- Translation

$$\bar{\mathbf{x}}' = \underbrace{\begin{pmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix}}_{4\times4} \bar{\mathbf{x}}$$

- Euclidean transform (translation + rotation), (also called the Special Euclidean group SE(3))

$$\bar{\mathbf{x}}' = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \bar{\mathbf{x}}$$

- Scaled rotation, affine transform, projective transform…

3D Transformations

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} I & \vert & t \end{bmatrix}_{3\times 4}$	3	orientation	
rigid (Euclidean)	$\begin{bmatrix} R & \vert & t \end{bmatrix}_{3\times 4}$	6	lengths	
similarity	$\begin{bmatrix} sR & \vert & t \end{bmatrix}_{3\times 4}$	7	angles	
affine	$\begin{bmatrix} A \end{bmatrix}_{3\times 4}$	12	parallelism	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{4\times 4}$	15	straight lines	

3D Euclidean Transformtions

- Translation $\mathbf{t}$ has 3 degrees of freedom

- Rotation R has 3 degrees of freedom

$$X = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3D Rotations

- Rotation matrix R
(also called the special orientation group SO(3))

Alternative representations

- Euler angles
- Axis/angle
- Unit quaternion

Rotation Matrix

- Orthonormal 3x3 matrix

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

- Advantage: Can be easily concatenated and inverted (how?)
- Disadvantage: Over-parameterized
  (9 parameters instead of 3)

Euler Angles

- Product of 3 consecutive rotations (e.g., around X-Y-Z axes)
- Roll-pitch-yaw convention is very common in aerial navigation (DIN 9300)
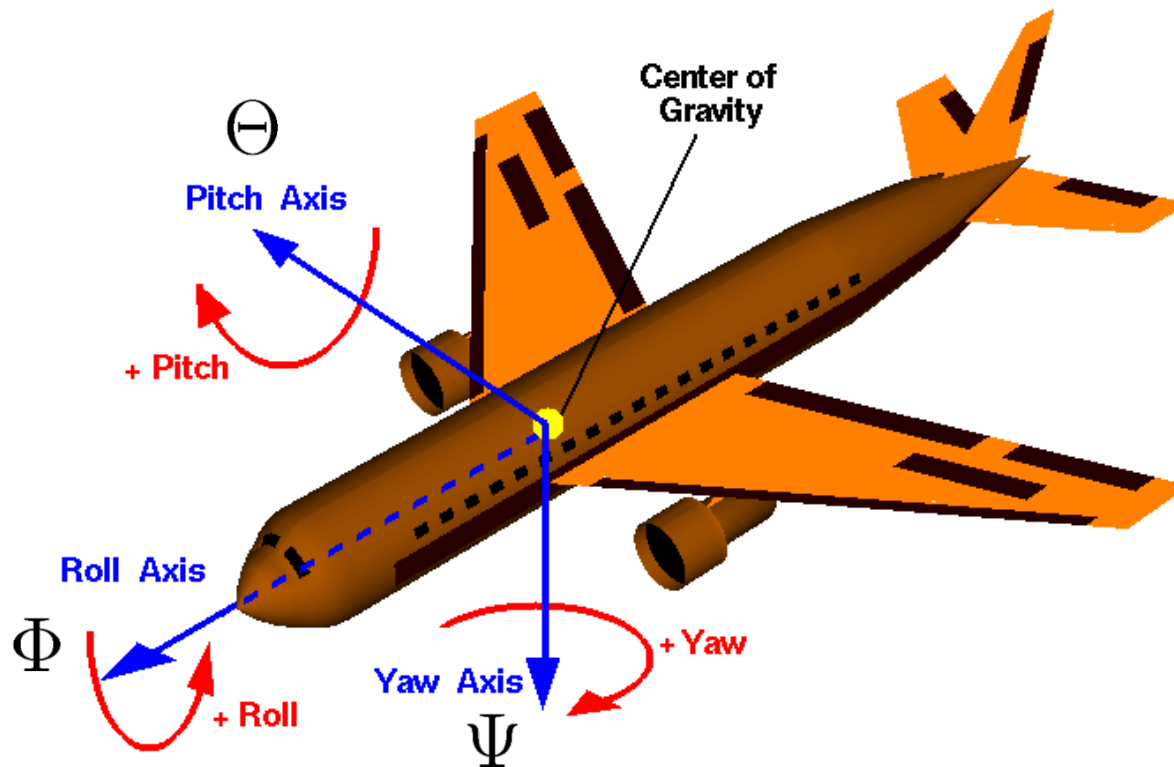
Roll-Pitch-Yaw Convention

- Yaw Ψ, Pitch Θ, Roll Φ to rotation matrix

$$R = R_Z(\Psi)R_Y(\Theta)R_X(\Phi)$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\Phi & \sin\Phi \\ 0 & -\sin\Phi & \cos\Phi \end{pmatrix} \begin{pmatrix} \cos\Theta & 0 & -\sin\Theta \\ 0 & 1 & 0 \\ \sin\Theta & 0 & \cos\Theta \end{pmatrix} \begin{pmatrix} \cos\Psi & \sin\Psi & 0 \\ -\sin\Psi & \cos\Psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\Theta\cos\Psi & \cos\Theta\sin\Psi & -\sin\Theta \\ \sin\Phi\sin\Theta\cos\Psi - \cos\Phi\sin\Psi & \sin\Phi\sin\Theta\sin\Psi + \cos\Phi\cos\Psi & \sin\Phi\cos\Theta \\ \cos\Phi\sin\Theta\cos\Psi + \sin\Phi\sin\Psi & \cos\Phi\sin\Theta\sin\Psi - \sin\Phi\cos\Psi & \cos\Phi\cos\Theta \end{pmatrix}$$

- Rotation matrix to Yaw-Pitch-Roll

$$\phi = \mathrm{Atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right)$$
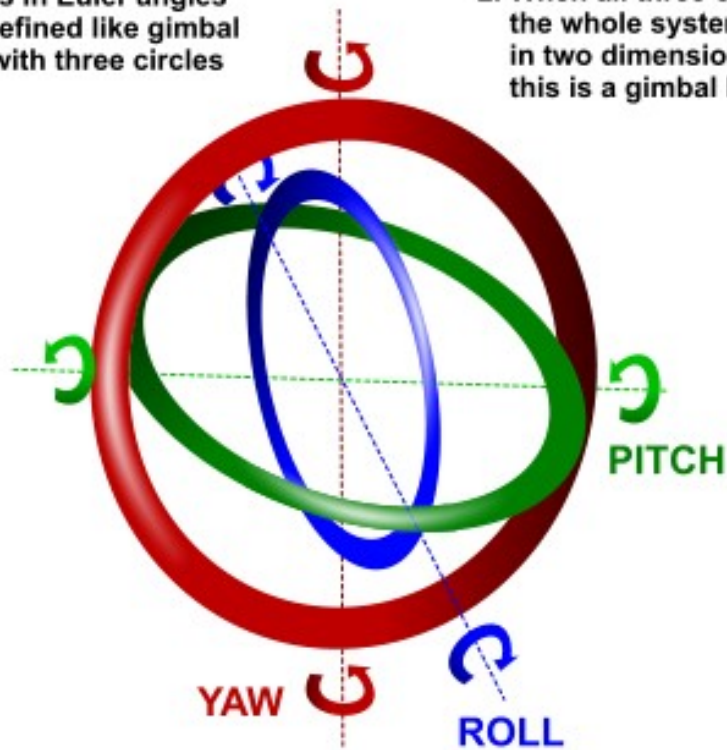
$$\psi = -\mathrm{Atan2}\left(\frac{r_{21}}{\cos(\phi)}, \frac{r_{11}}{\cos(\phi)}\right)$$

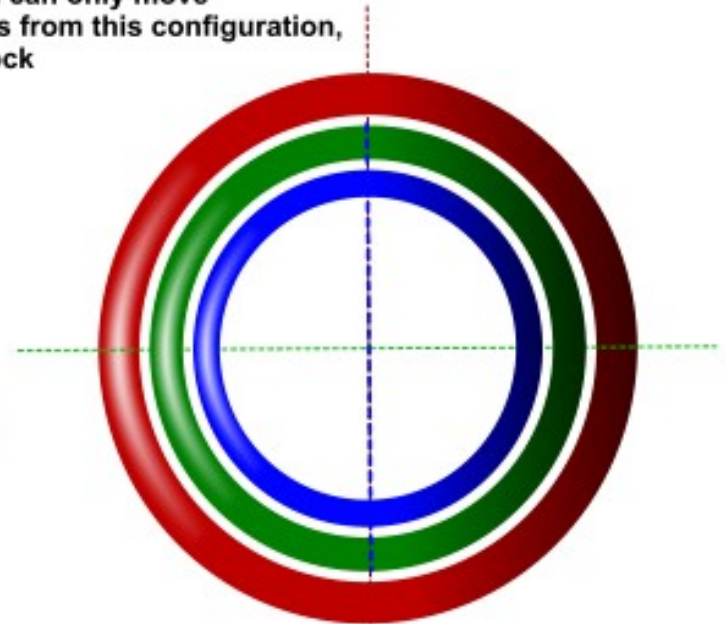$$\theta = \mathrm{Atan2}\left(\frac{r_{32}}{\cos(\phi)}, \frac{r_{33}}{\cos(\phi)}\right)$$

Gimbal Lock

- When the axes align, one degree-of-freedom (DOF) is lost…

1. Rotations in Euler angles can be defined like gimbal system with three circles

2. When all three circles are lined up, the whole system can only move in two dimensions from this configuration, this is a gimbal lock

3. Usage of quaternions can help to avoid such situations

PITCH

YAW

ROLL

Axis/Angle

- Represent rotation by
  - rotation axis $\hat{\mathbf{n}}$ and
  - rotation angle θ
- 4 parameters $(\hat{\mathbf{n}}, \theta)$
- 3 parameters $\boldsymbol{\omega} = \theta\hat{\mathbf{n}}$
  - length is rotation angle
  - also called the angular velocity
  - minimal but not unique (why?)
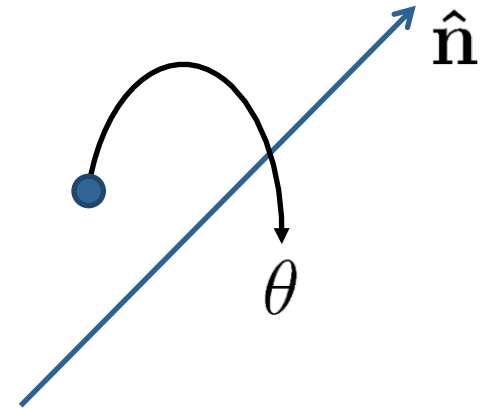
Conversion

- Rodriguez' formula

$$R(\hat{\mathbf{n}}, \theta) = I + \sin\theta [\hat{\mathbf{n}}]_\times + (1 - \cos\theta)[\hat{\mathbf{n}}]_\times^2$$
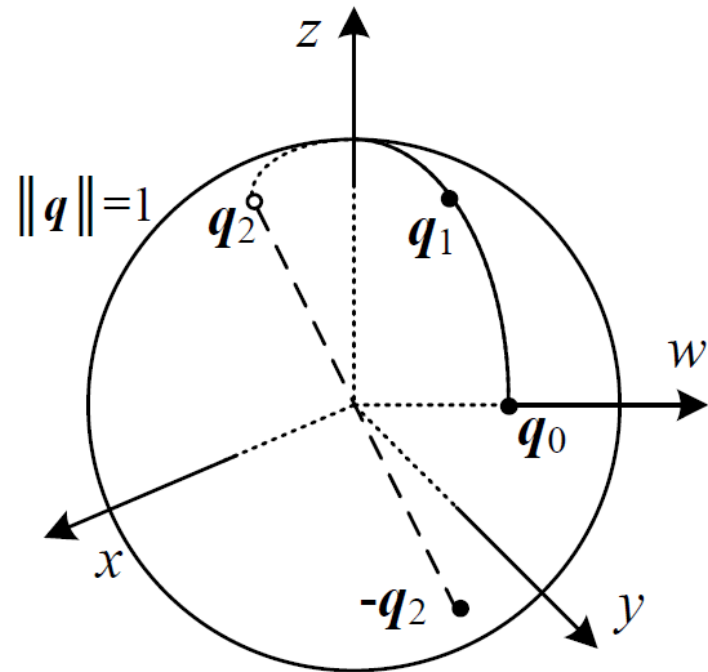
- Inverse

$$\theta = \cos^{-1}\left(\frac{\mathrm{trace}(R) - 1}{2}\right), \hat{\mathbf{n}} = \frac{1}{2\sin\theta}\begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}$$

- see: An Invitation to 3D Vision, Y. Ma, S. Soatto, J. Kosecka, S. Sastry, Chapter 2 (available online)

Unit Quaternions

- Quaternion $\mathbf{q} = (q_x, q_y, q_z, q_w)^\top \in \mathbb{R}^4$
- Unit quaternions have $\|\mathbf{q}\| = 1$
- Opposite sign quaternions represent the same rotation
- Otherwise unique $\mathbf{q} = -\mathbf{q}$

Unit Quaternions

- Advantage: multiplication and inversion operations are efficient
- Quaternion-Quaternion Multiplication

$$\mathbf{q}_0\mathbf{q}_1 = (\mathbf{v}_0, w_0)(\mathbf{v}_1, w_1)$$
$$= (\mathbf{v}_0 \times \mathbf{v}_1 + w_0\mathbf{v}_1 + w_1\mathbf{v}_0, w_0w_1 - \mathbf{v}_0\mathbf{v}_1)$$

- Inverse (flip sign of v or w)

$$\mathbf{q}^{-1} = (\mathbf{v}, w)^{-1}$$
$$= (\mathbf{v}, -w)$$

Unit Quaternions

- Quaternion-Vector multiplication (rotate point p with rotation q)

$$\mathbf{p}' = \mathbf{q}\bar{\mathbf{p}}\mathbf{q}^{-1}$$

with
$$\bar{\mathbf{p}} = (x, y, z, 0)^{\top}$$

- Relation to Axis/Angle representation

$$\mathbf{q} = (\mathbf{v}, w) = (\sin\frac{\theta}{2}\hat{\mathbf{n}}, \cos\frac{\theta}{2})$$

3D Orientations

- **Note:** In general, it is very hard to "read" 3D orientations/rotations, no matter in what representation

- **Observation:** They are usually easy to visualize and can then be intuitively interpreted

- **Advice:** Use 3D visualization tools for debugging (RVIZ, libqglviewer, …)

C++ Libraries for Lin. Alg./Geometry

- Many C++ libraries exist for linear algebra and 3D geometry
- Typically conversion necessary
- Examples:
  - C arrays, std::vector (no linear alg. functions)
  - gsl (gnu scientific library, many functions, plain C)
  - boost::array (used by ROS messages)
  - Bullet library (3D geometry, used by ROS tf)
  - **Eigen (both linear algebra and geometry)**
  - **Sophus (Eigen extensions, SE(3)/se(3), see Lecture 2)**

What we will cover today

- Linear algebra, notation
- 3D geometry
- **Projective geometry**
    - **Camera intrinsics**
    - **Epipolar geometry**

- Robot Operating System (ROS)

Pin-hole Camera

- Lit scene emits light
- Film/sensor is light sensitive
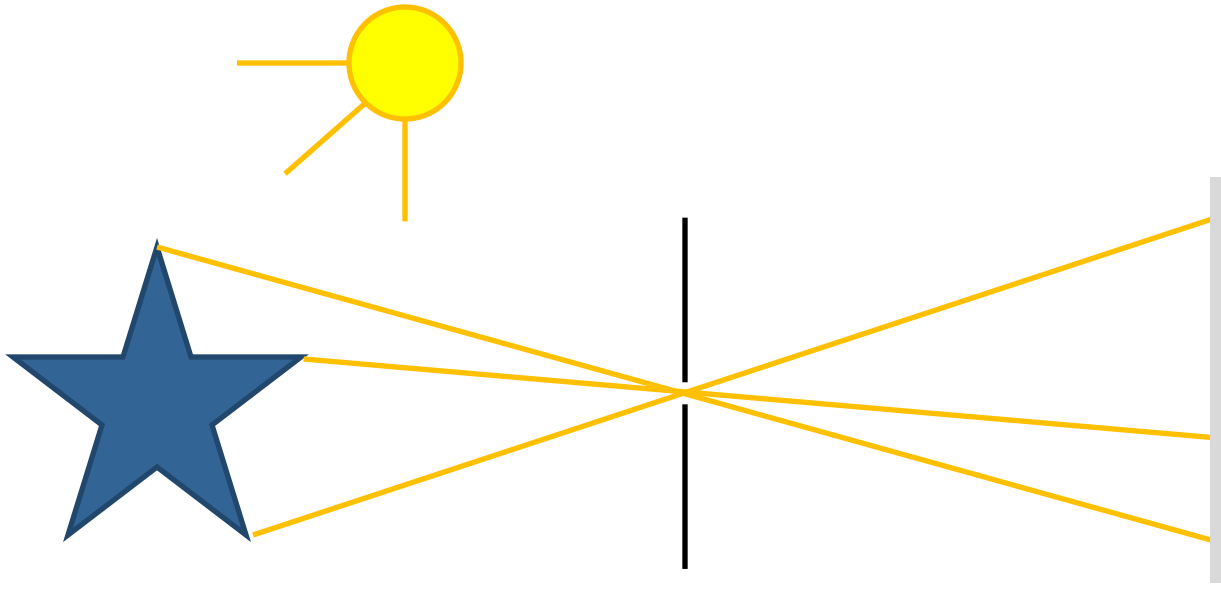
3D to 2D Perspective Projection
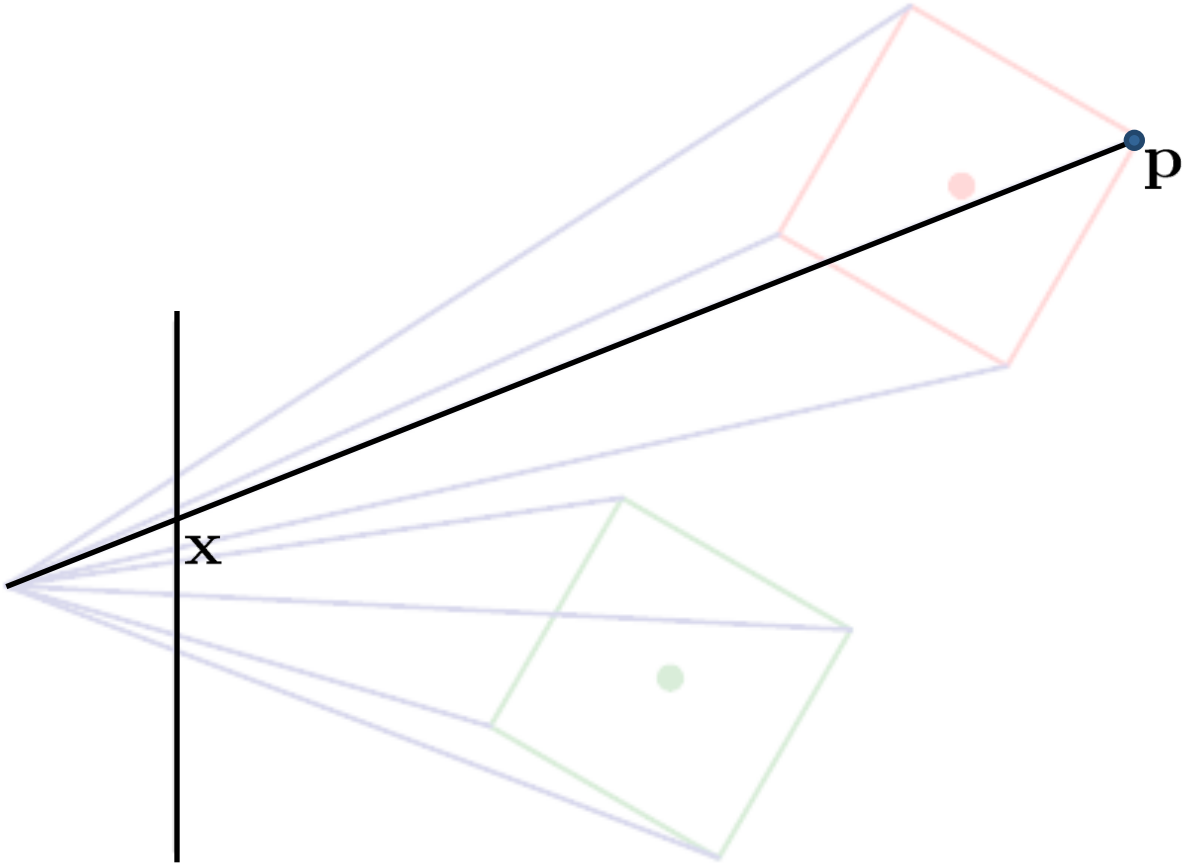
3D to 2D Perspective Projection

- 3D point $\mathbf{p}$ (in the camera frame)
- 2D point $\mathbf{x}$ (on the image plane)
- Pin-hole camera model

$$\tilde{\mathbf{x}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tilde{\mathbf{p}}$$

- Remember, $\tilde{\mathbf{x}}$ is homogeneous, need to normalize

$$\tilde{\mathbf{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} \quad \Rightarrow \quad \mathbf{x} = \begin{pmatrix} \tilde{x}/\tilde{z} \\ \tilde{y}/\tilde{z} \end{pmatrix}$$

Camera Intrinsics

- So far, 2D point is given in meters on image plane
- But: we want 2D point be measured in pixels (as the sensor does)
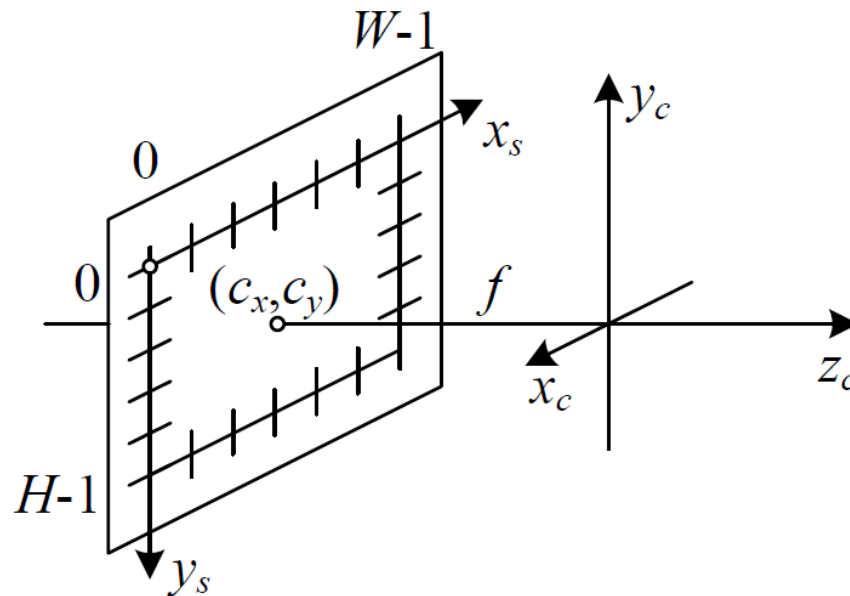
Camera Intrinsics

- Need to apply some scaling/offset

$$\tilde{\mathbf{x}} = \underbrace{\begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{\text{intrinsics } K} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\text{projection}} \tilde{\mathbf{p}}$$

- Focal length f_x, f_y
- Camera center c_x, c_y
- Skew s

Camera Extrinsics

- Assume $\tilde{\mathbf{p}}_w$ is given in world coordinates
- Transform from world to camera (also called the camera extrinsics)

$$\tilde{\mathbf{p}} = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \tilde{\mathbf{p}}_w$$

- Full camera matrix

$$\tilde{\mathbf{x}} = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & \mathbf{t} \end{pmatrix} \tilde{\mathbf{p}}_w$$

Real Camera Lenses

- Lit scene emits light
- Film/sensor is light sensitive
- A lens focuses rays onto the film/sensor
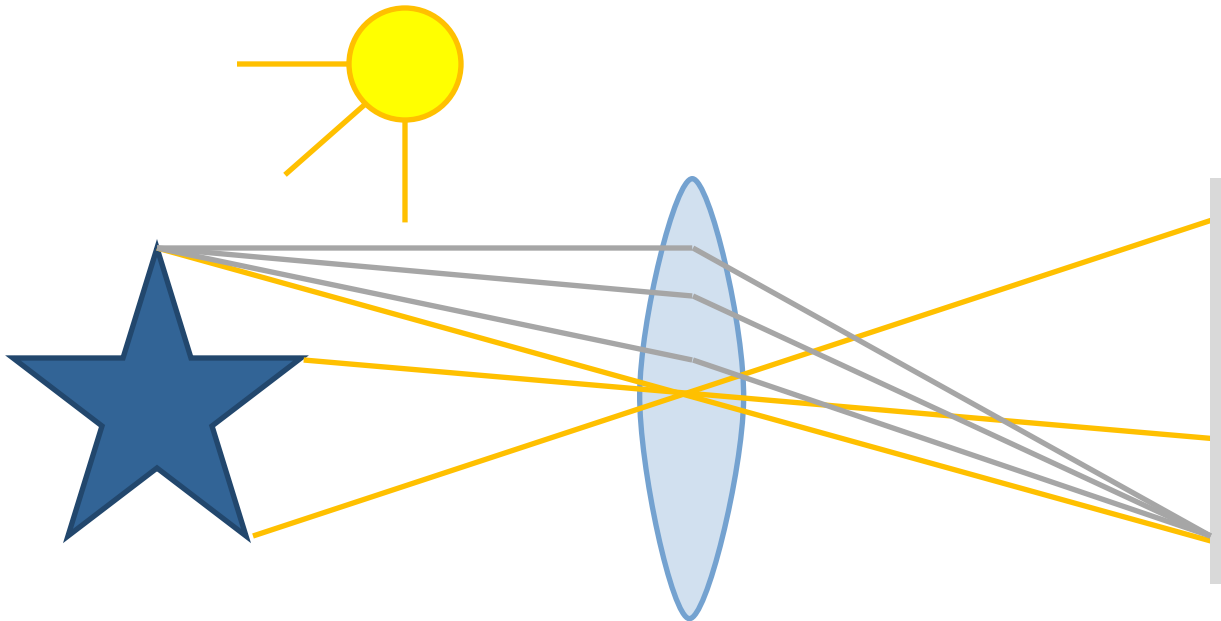
Real Cameras

- Radial distortion of the image
  - Caused by imperfect lenses
  - Deviations are most noticeable for rays that pass through the edge of the lens

Radial Distortion

- Radial distortion of the image
  - Caused by imperfect lenses
  - Deviations are most noticeable for rays that pass through the edge of the lens
- Typically compensated with a low-order polynomial

$$\hat{x}_c = x_c(1 + \kappa_1 r_c^2 + \kappa_2 r_c^4)$$
$$\hat{y}_c = y_c(1 + \kappa_1 r_c^2 + \kappa_2 r_c^4)$$

Digital Cameras

- Vignetting
- De-bayering
- Rolling shutter and motion blur
- Compression (JPG)
- Noise
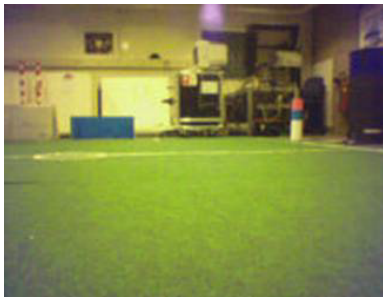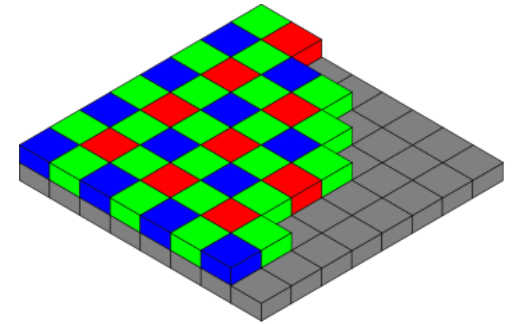
Epipolar Geometry

- Let's consider two cameras that observe a 3D world point
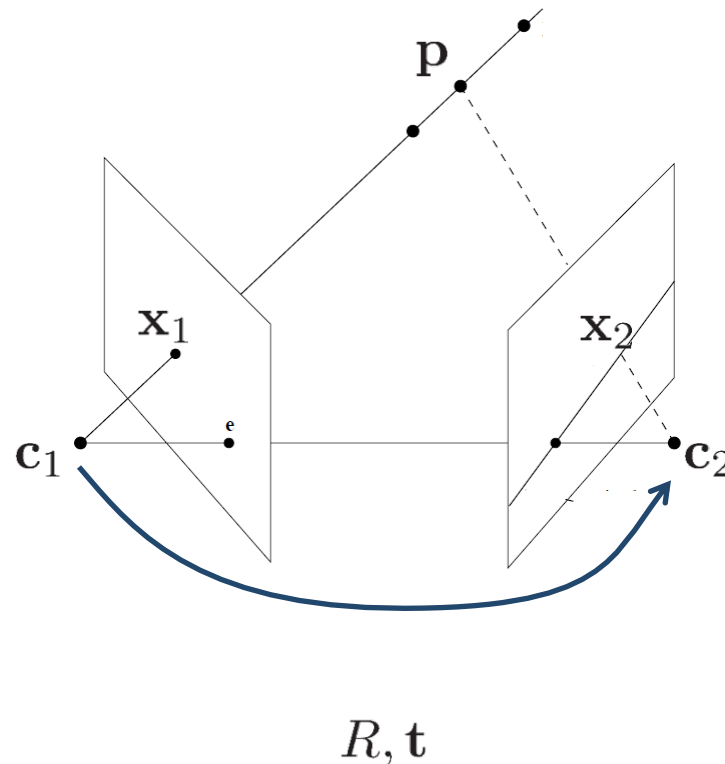
Epipolar Geometry

- The line connecting both camera centers is called the **baseline**

baseline
(line joining both camera centers)
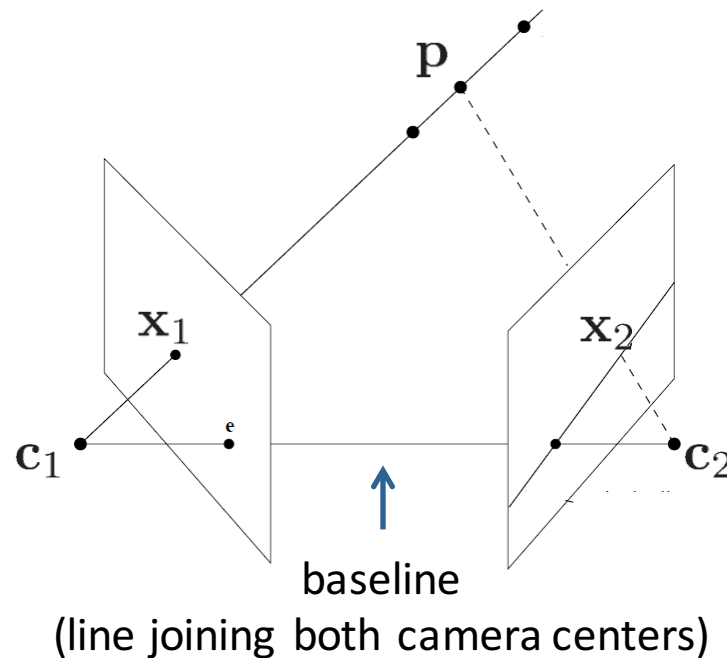
Epipolar Geometry

- Given the image of a point in one view, what can we say about its position in another?
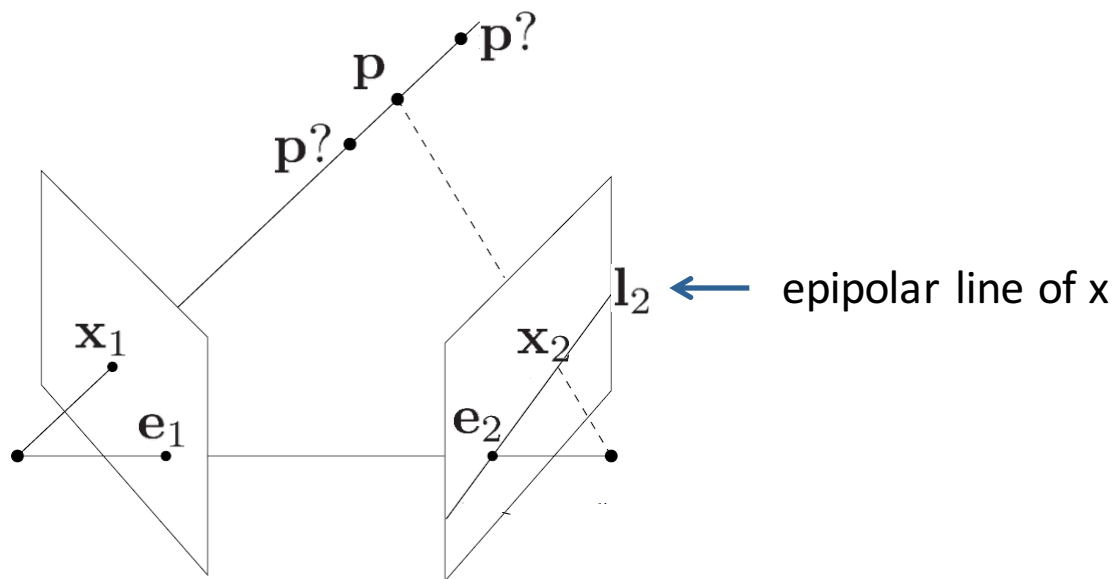
epipolar line of x

- A point in one image "generates" a line in another image (called the **epipolar line**)

What we will cover today

- Linear algebra, notation
- 3D geometry
- Projective geometry
  - Camera intrinsics
  - Epipolar geometry

- **Robot Operating System (ROS)**

Modern Robot Architectures

- Robots became rather complex systems
- Often, a large set of individual capabilities is needed
- Flexible composition of different capabilities for different tasks

Best Practices for Robot Architectures

- Modular
- Robust
- De-centralized
- Facilitate software re-use
- Hardware and software abstraction
- Provide introspection
- Data logging and playback
- Easy to learn and to extend

Robotic Middleware

- Provides infrastructure
- Communication between modules
- Data logging facilities
- Tools for visualization
- Several open-source systems available
    - **ROS (Robot Operating System)**,
    - Player/Stage,
    - CARMEN,
    - YARP,
    - OROCOS

Example Architecture for Navigation
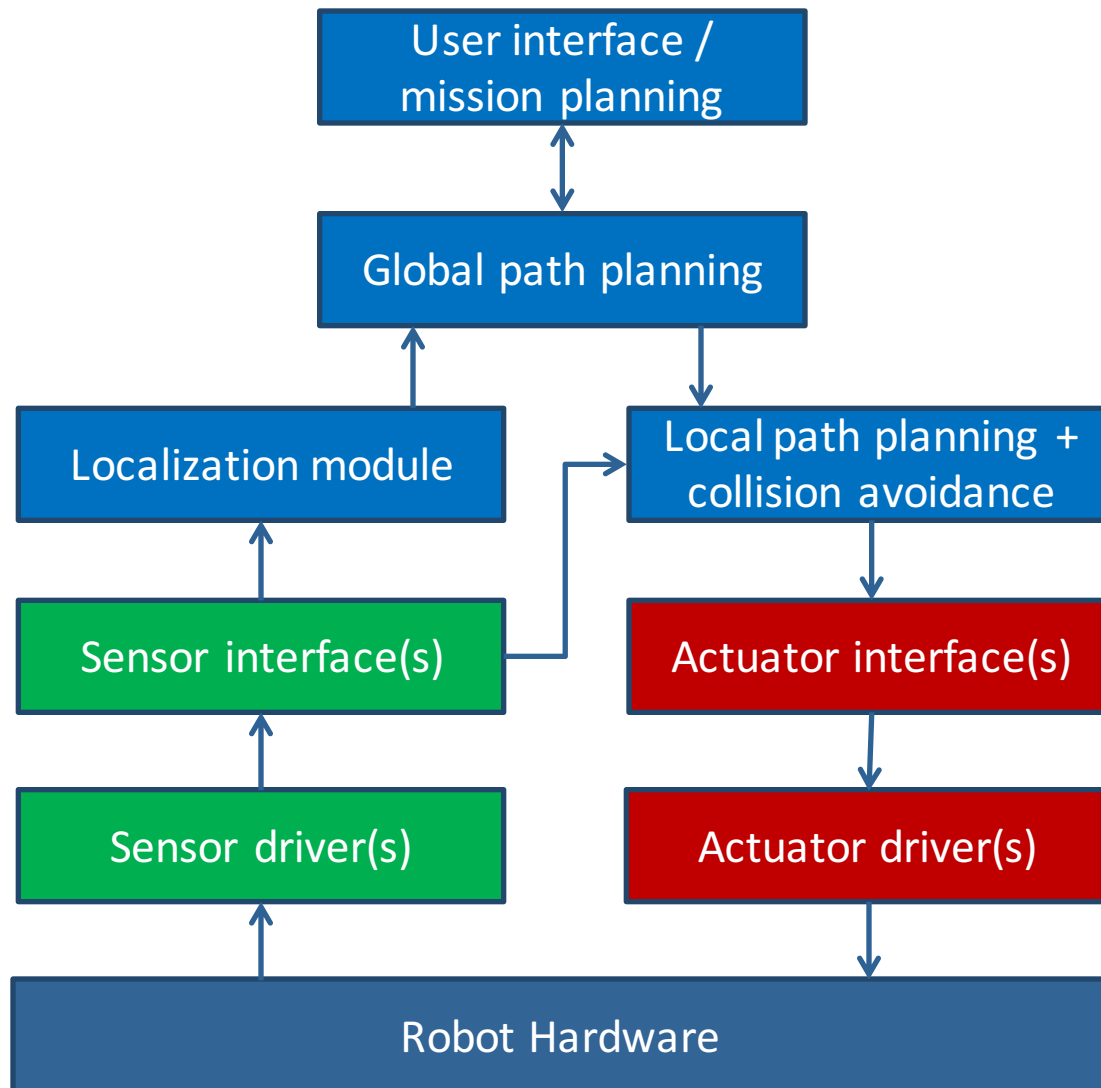
PR2 Software Architecture

- Two 7-DOF arms, grippers, torso, 2-DOF head
- 7 cameras, 2 laser scanners
- Two 8-core CPUs, 3 network switches
- 73 nodes, 328 message topics, 174 services
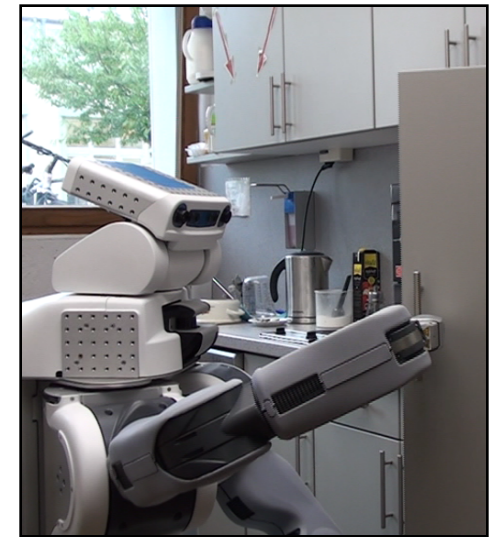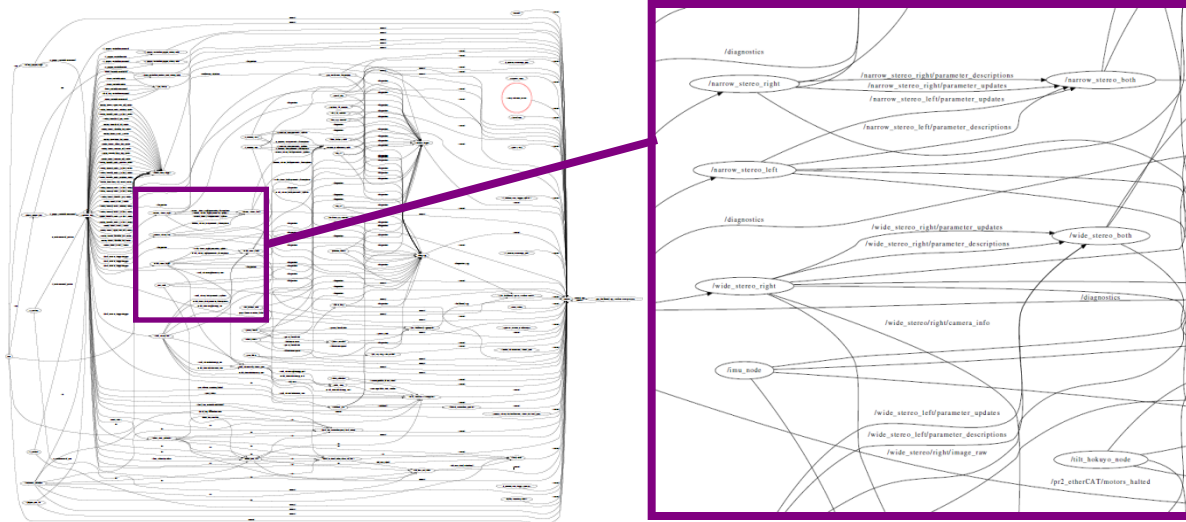
Communication Paradigms

- Message-based communication
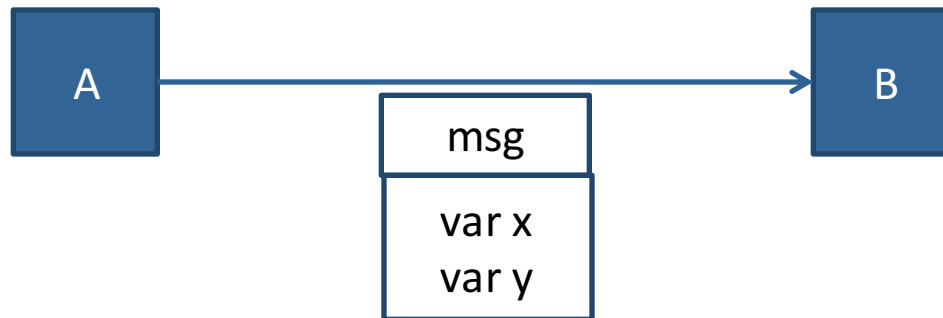
- Direct (shared) memory access
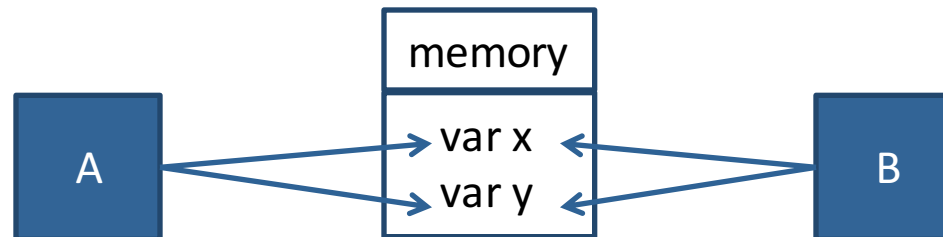
Forms of Communication

- Push

- Pull

- Publisher/subscriber

- Publish to blackboard

- Remote procedure calls / service calls

- Preemptive tasks / actions

Push

- Broadcast
- One-way communication
- Send as the information is generated by the producer P

Pull

- Data is delivered upon request by the consumer C (e.g., a map of the building)
- Useful if the consumer C controls the process and the data is not required (or available) at high frequency

data request

P ←————————————— C
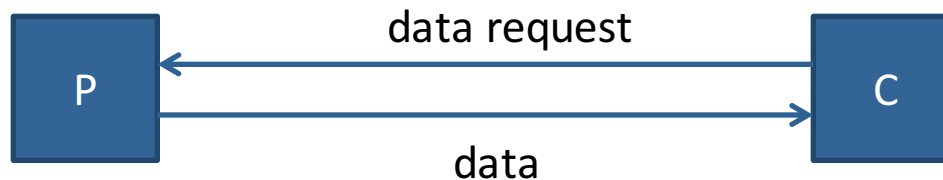
data

Publisher/Subscriber

- The consumer C requests a subscription for the data by the producer P (e.g., a camera or GPS)
- The producer P sends the subscribed data as it is generated to C
- Data generated according to a trigger (e.g., sensor data, computations, other messages, …)
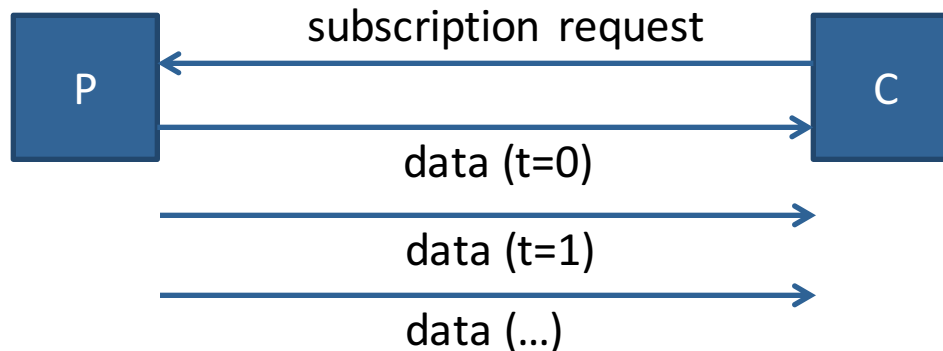
Publish to Blackboard

- The producer P sends data to the blackboard (e.g., parameter server)
- A consumer C pull data from the blackboard B
- Only the last instance of data is stored in the blackboard B
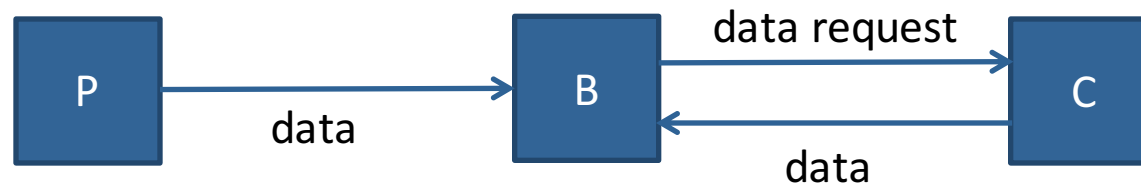
Service Calls

- The client C sends a request to the server S
- The server returns the result
- The client waits for the result (synchronous communication)
- Also called: Remote Procedure Call

C → request + input data → S

C ← result ← S
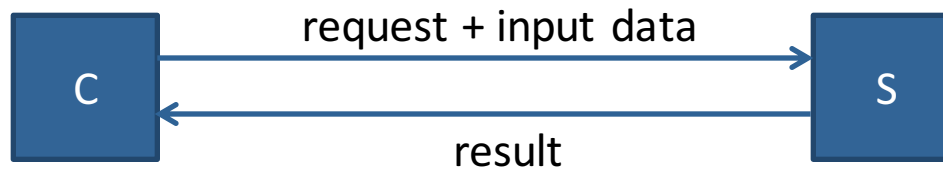
Actions (Preemptive Tasks)

- The client requests the execution of an enduring action (e.g., navigate to a goal location)
- The server executes this action and sends continuously status updates
- Task execution may be canceled from both sides (e.g., timeout, new navigation goal,…)

Robot Operating System (ROS)

- We will use ROS in the lab course
- http://www.ros.org/
- Installation instructions, tutorials, docs
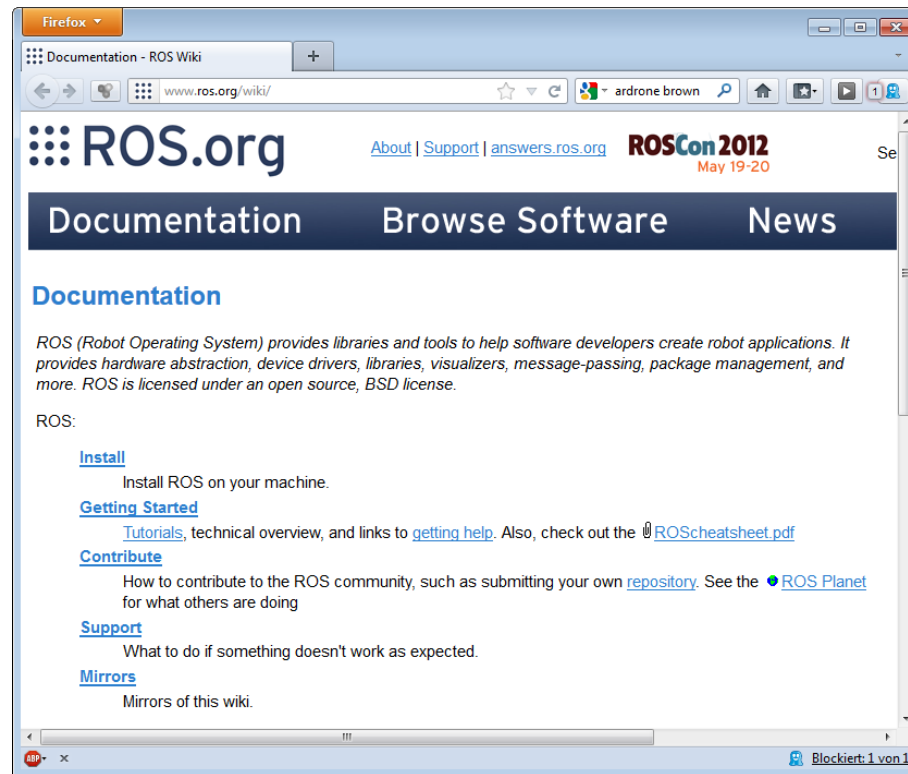
Concepts in ROS

- Nodes: programs that communicate with each other
- Messages: data structure (e.g., "Image")
- Topics: typed message channels to which nodes can publish/subscribe (e.g., "/camera1/image_color")
- Parameters: stored in a blackboard


```
┌──────────────────┐                      ┌──────────────────┐
│  camera_driver   │ ───────────────────> │  face_detector   │
└──────────────────┘        Image         └──────────────────┘
```
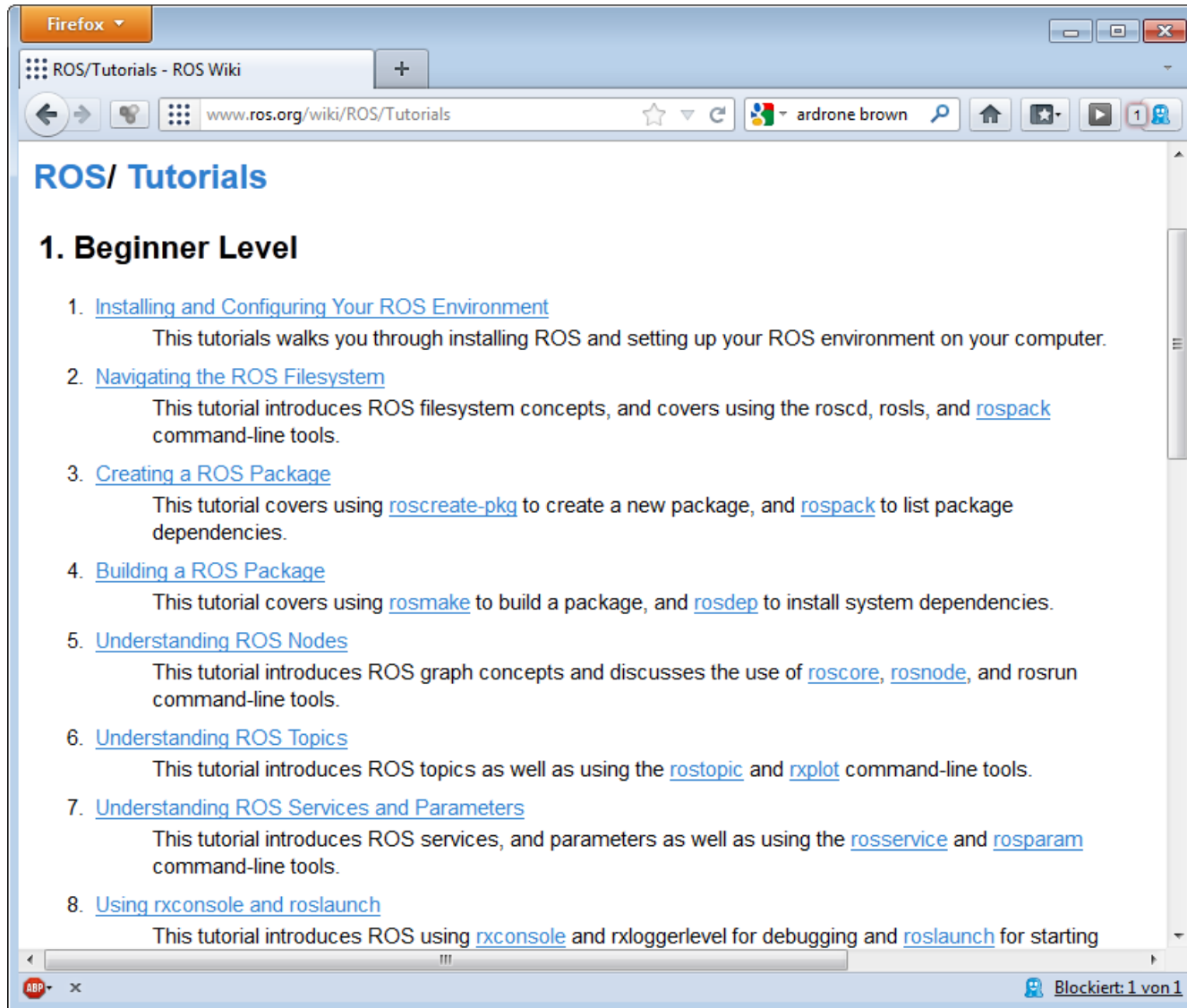
Software Management

- Package: atomic unit of building, contains one or more nodes and/or message definitions

- Build system "catkin" on top of CMake to ease building packages and ROS specific code, and to resolve dependencies between packages

- Repository: contains several packages, typically one repository per institution

Useful Tools

- catkin_init_workspace

- catkin_create_pkg

- catkin_make

- roscore

- rosnode list/info

- rostopic list/echo

- rosbag record/play

- rosrun

Tutorials in ROS

Questions ?