# Localization

# Localization

**global** aka **absolute**: environment based

- sensed features
  - landmarks (passive)
  - beacons (active)
- with known locations
- geometric calculations
  - e.g., triangulation
- with absolute error
  - time invariant

**relative**: self-referenced

- based on ego-motion
  - e.g.,
  - odometry = cumulative kinematic motion estimates
  - inertial sensors
    - gyros, accelerometers
    - double integration
- with relative error
  - accumulates over time
  - bump noise

# Absolute Localization: Triangulation, Tri/Multilateration

# Triangulation

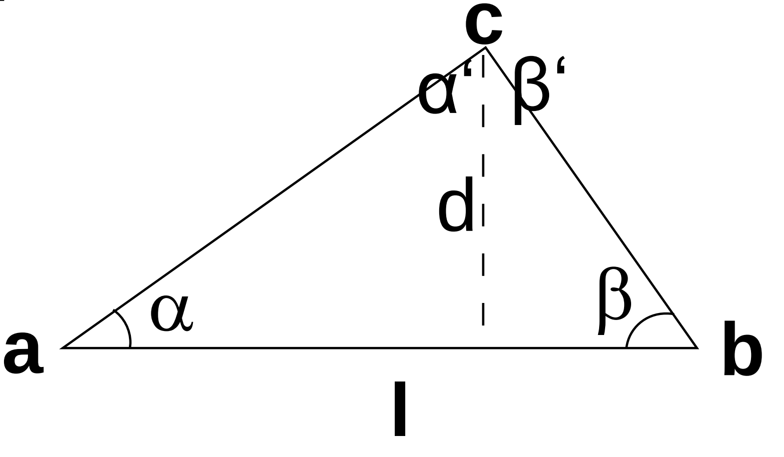
given

- 2 points a, b with known locations
  - aka beacons
  - forming a baseline l (line segment)
- 2 angles α, β
  - aka bearings (of c from the baseline)
  - or α‘, β‘ (bearings of beacons from c)
  - note: α‘ = 90º – α, β‘ = 90º – β

find location of c

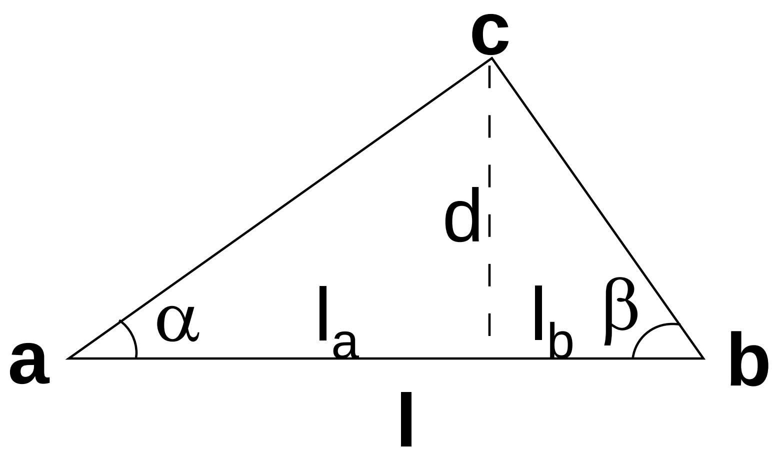
$$l = \overline{ab}$$

$$(\overline{ab} = \{(x,y) | a + cb, c \in [0,1]\})$$



# Triangulation

- baseline l: $l = l_a + l_b$
- angles α, β




location of c:

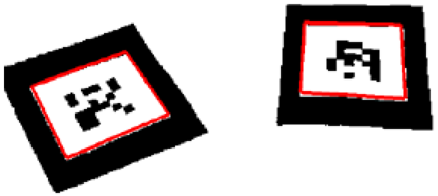
$$d = \frac{1}{\left(\frac{1}{\tan\alpha} + \frac{1}{\tan\beta}\right)} = \frac{l \sin\alpha \sin\beta}{\sin(\alpha+\beta)}$$

$$l_a = \frac{d}{\tan\alpha}, l_b = \frac{d}{\tan\beta}$$

Note: using two alternative ways for d

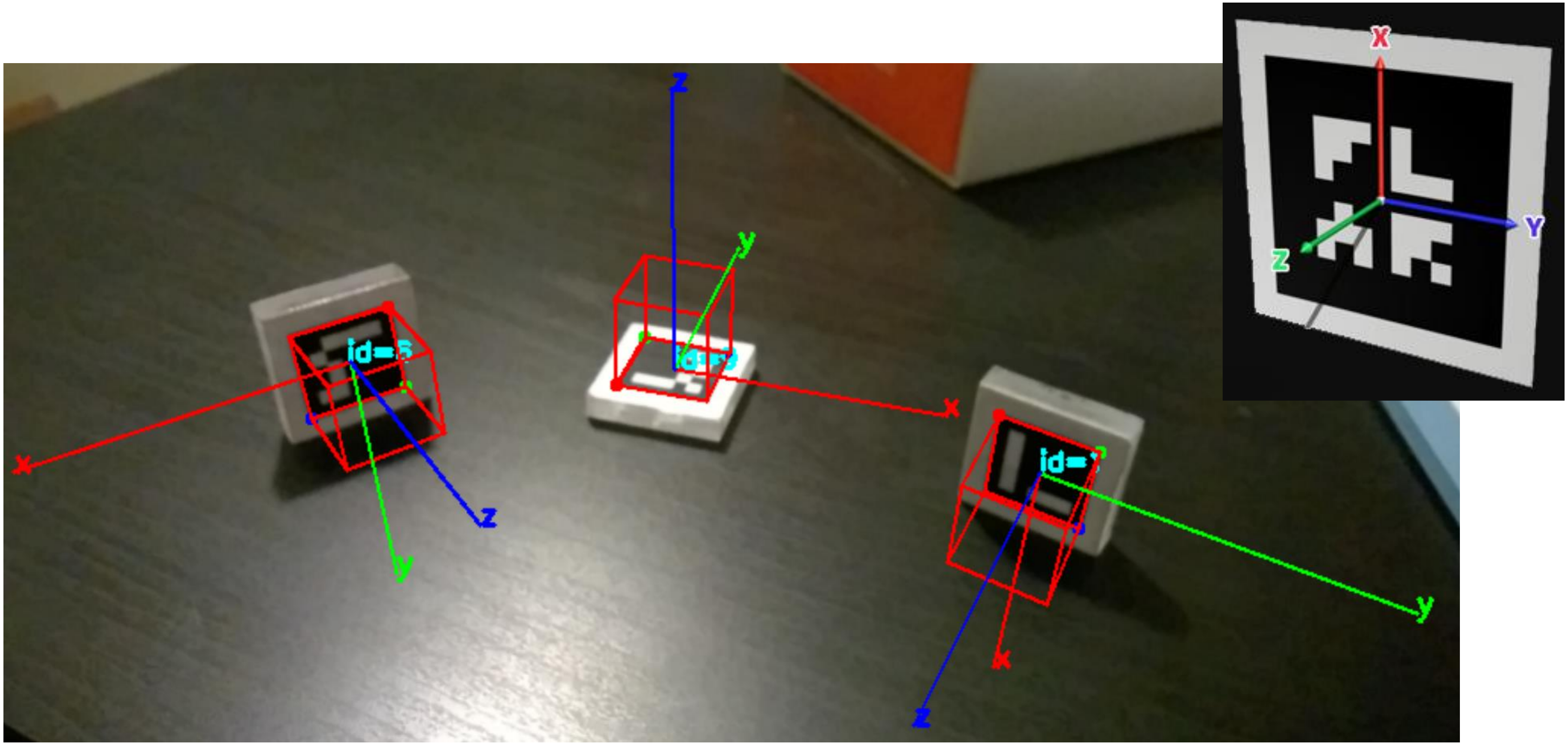
- tangens
- law of sines $\frac{\sin\alpha}{\overline{bc}} = \frac{\sin\beta}{\overline{ac}} = \frac{\sin\gamma}{\overline{ab}}$

# Bearings



e.g., via computer vision

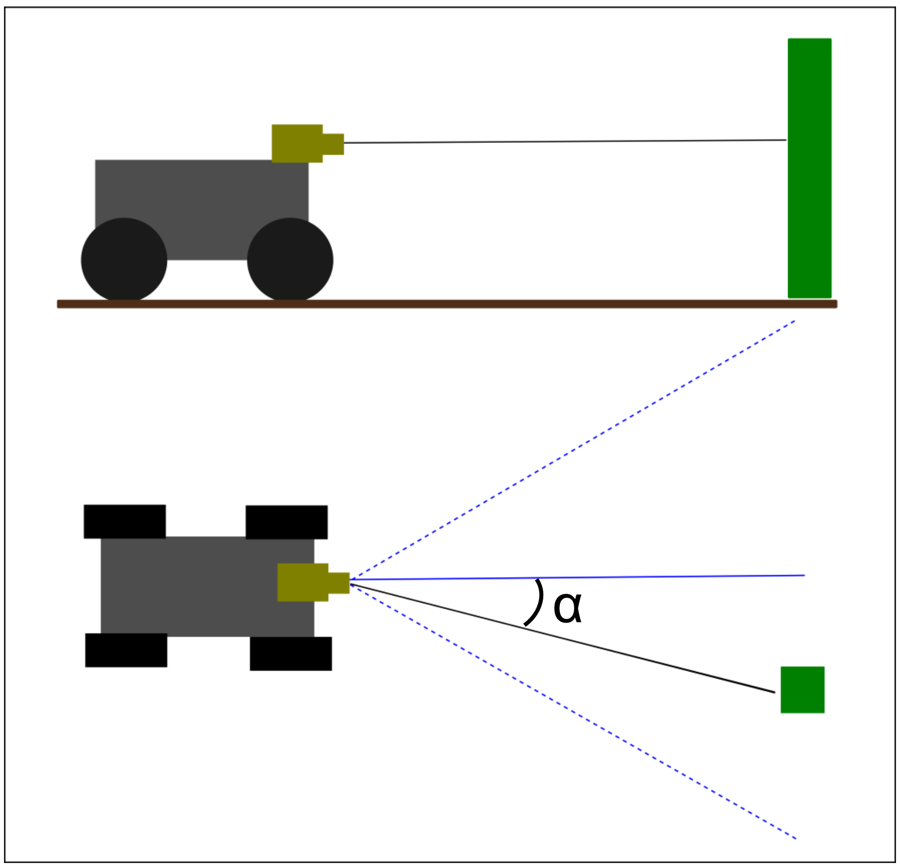
- artificial markers: e.g., Augmented Reality (AR) markers



# Bearings

e.g., via computer vision

- natural landmarks:
  e.g., flat world assumption & camera pin-hole/lens model



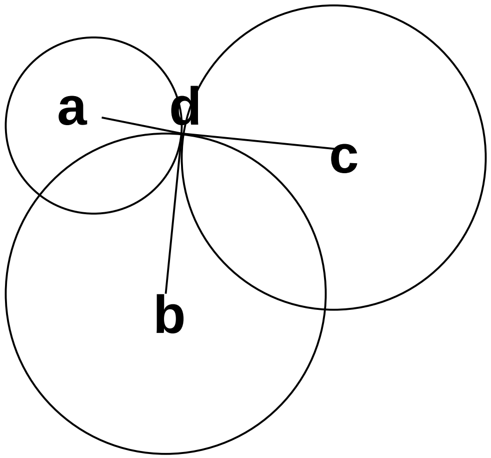

$$\sin(\alpha) = \frac{w_p x_o - \frac{1}{2} w_I}{v}$$

- α: bearing
- $(x_o, y_o)$ : pixel location of object
- $h_I$ x $w_I$ : dimensions of the imager
- $h_p$ x $w_p$ : dimensions of a pixel
- v : distance of imager from lens
  (= f for pin-hole model)

# Trilateration

given

- 3 points a, b, c with known locations $$i = \left(x_i, y_i\right)^T, i \in \{a, b, c\}$$
- 3 ranges $d_i$ from a, b, c to d $$d_i = \sqrt{\left(x_d - x_i\right)^2 + \left(y_d - y_i\right)^2}, i \in \{a, b, c\}$$




find location of d

$$d = \begin{pmatrix} x_d \\ y_d \end{pmatrix}$$

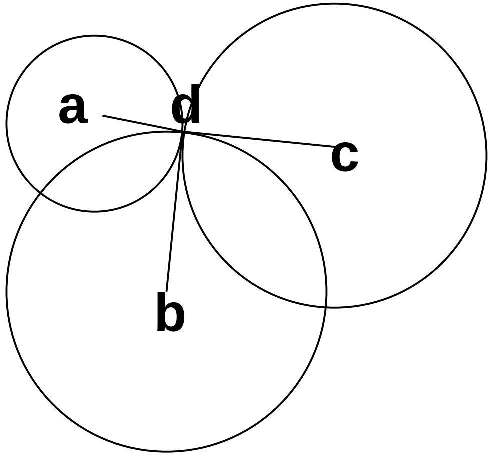
$$= \begin{pmatrix} 2(x_a - x_c) & 2(y_a - y_c) \\ 2(x_b - x_c) & 2(y_b - y_c) \end{pmatrix}^{-1} \begin{pmatrix} x_a{}^2 - x_c{}^2 + y_a{}^2 - y_c{}^2 + d_a{}^2 - d_c{}^2 \\ x_b{}^2 - x_c{}^2 + y_b{}^2 - y_c{}^2 + d_b{}^2 - d_c{}^2 \end{pmatrix}$$

# Trilateration

reality

- (range) measurements are noisy
- but possibly more than 3 beacons & measurements

=> multilateration



# Multilateration

- one target, e.g., robot, to be localized *(x,y)*
- *n>3* beacons *i* with known locations $(x_i, y_i)$
- and *n* noisy ranges $d_i$ with error $e_i$

⇒find (x,y) such that $e_i$ minimal

⇒use Least-Squares optimization

(short) excursus

# Least Squares Optimization

# Least Squares Optimization

- given
  - $n$ noisy data points $(x_i,y_i)$ with error e (aka residual r)
  - according to a model function $f_a$ ,
  - i.e., $y_i = f_a(x_i) + e$
- find
  - the $m$ parameters $a_j$
  - of the model function $f_a()$
  - (note: $m < n$ , i.e., overdetermined)
- such that the model “fits the data well”
- i.e., a **regression** problem

# Least Squares Optimization

- given: $n$ data points $(x_i,y_i)$, $y_i = f_a(x_i) + e_i$
- find: $m$ parameters $a_j$ $(m < n)$

such that the model "fits the data well", e.g.,

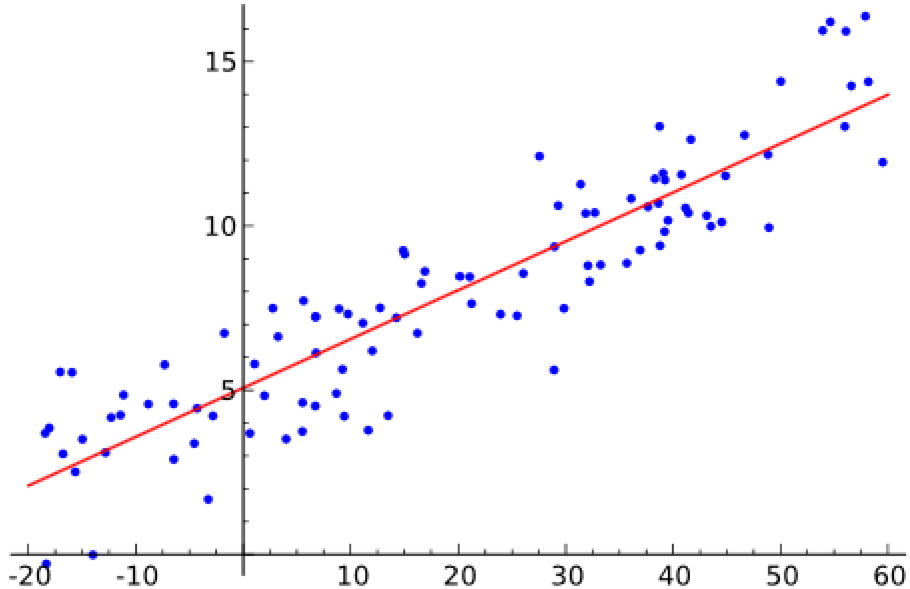
- **minimize** the **sum of the squared errors** (hence **least squares**)
- i.e., find

$$\min_a S : S = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left( y_i - f_a(x_i) \right)^2$$

# Least Squares Optimization

simple example

- fit 2D line fct $y = a_1 x + a_0$
- to $n$ data points $(x_i, y_i)$



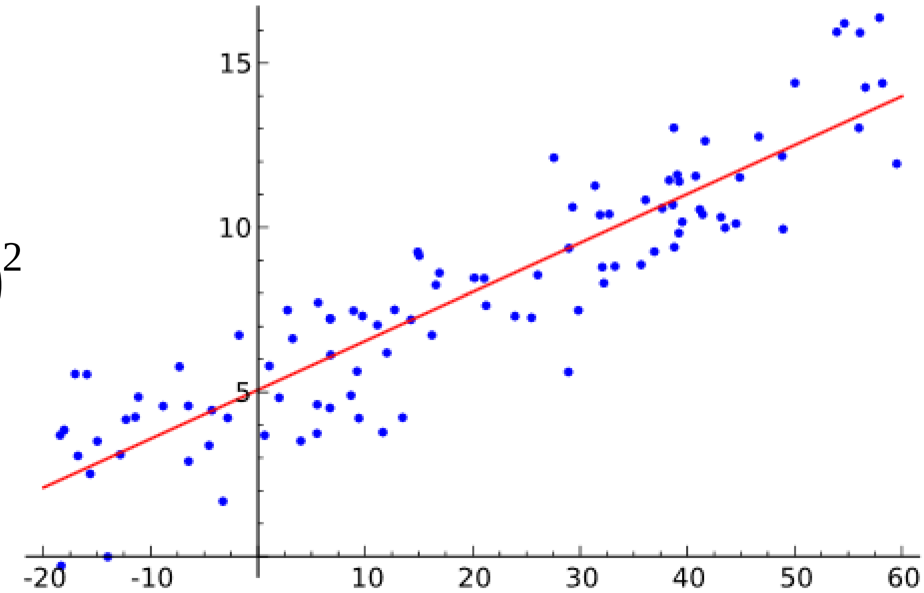
# Least Squares Optimization

solution

- consider $S$ as a function in $a_1$ and $a_0$
- find minimum of $S$, i.e.,
- compute partial derivatives & find zero crossing

$$\min_{a_0, a_0} S :$$

$$S = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left( y_i - a_1 x_i + a_0 ) \right)^2$$



# Least Squares Optimization

(re-arranged)

partial derivatives:

$$\left(\sum_{i=1}^{n} x_i\right) a_1 + n a_0 = \sum_{i=1}^{n} y_i$$

$$\left(\sum_{i=1}^{n} x_i^{\,2}\right) a_1 + \left(\sum_{i=1}^{n} x_i\right) a_0 = \sum_{i=1}^{n} x_i y_i$$

solution: $$a_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$a_0 = \bar{y} - a_1 \bar{x}$$

with

arithmetic means of $x_i$ ,$y_i$

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \quad \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

# Least Squares Optimization

fortunately, things can be generalized:

- Linear Least Squares (LLS)
  - aka Ordinary Least Squares (OLS)
  - has closed form solution

- Non-Linear Least Squares
  - requires iterative optimization
  - based on Linear Least Squares

# Linear Least Squares (LLS)

given:

- $n$ measurements of a linear $f_a()$ with $m$ parameters
- i.e., $n$ linear equations with $m$ parameters $a_j$

$$i \in \{1, \ldots, n\} : y_i = \sum_{j=1}^{m} a_j x_{ij} = \sum_{j=1}^{m} x_{ij} a_j \qquad\qquad y = Xa$$

find: best LS parameter fit, i.e.,

$$a^* = \underset{a}{\operatorname{argmin}} S(a) : S(a) = \sum_{i=1}^{n} \left( y_i - \sum_{j=1}^{m} x_{ij} a_j \right)^2 = \|y - Xa\|^2$$

# Linear Least Squares (LLS)

given: $y = Xa$

find: $a^* = \underset{a}{\arg\min} \|y - Xa\|^2$

general closed form solution:

$$a^* = (X^T X)^{-1} X^T y$$

looks familiar???

# Linear Least Squares (LLS)

given: $y = Xa$

find: $a^* = \underset{a}{\arg\min} \|y - Xa\|^2$

general closed form solution:

$$a^* = (X^T X)^{-1} X^T y$$

$$= X^+ y$$

pseudo-inverse
(and option to use SVD)

# Linear Least Squares (LLS)

linear algebra view:

(note: “swapped” A/X x/a in common notations)

$$A: n \times n \quad x, b: n \times 1 \qquad Ax = b \Rightarrow x = A^{-1}b$$

$$A: m \times n, \; x: n \times 1, \; b: m \times 1 \qquad Ax = b \Rightarrow x = A^{+}b$$

**Least Squares fit**

noisy $a_{ij}$: $\tilde{a}_{ij} \cong a_{ij}$, i.e., $\tilde{A} \cong A: m \times n$

$\tilde{A}x = b \Rightarrow x^* = \tilde{A}^{+}b$ with x* is best (LS) fit

# Least Squares Optimization

in general:

- Linear Least Squares (LLS)
  - closed form with pseudo-inverse
- **Non-Linear Least Squares**
  - iterative optimization
  - **Gauß-Newton-Method**

# Non-Linear Least Squares

**Gauß-Newton-Method**

- f non-linear fct with *m* parameters *a*: y = $f_a$(x):=f(x,a)
- residual (error): fct r(a) in *a*
- find LS fit for *a* ,i.e.,

$$a^* = \arg\min_a S(a) : S(a) = \sum_{i=1}^{n} \left( y_i - f(x_i, a) \right)^2 := \sum_{i=1}^{n} r(a)^2$$

# Non-Linear Least Squares

**Gauß-Newton-Method**

find $a^* = \underset{a}{\arg\min} \sum_{i=1}^{n} r(a)^2$

iterative algorithm

- take initial guess $a_0$
- iterate in the spirit of Newton's method
- with Jacobian $J_r$ of r(a), resp. Jacobian $J_f$ of f(x,a)

$$a_{t+1} = a_t - (J_r{}^T J_r)^{-1} J_r{}^T \cdot r(a_t) \qquad \Leftrightarrow \qquad a_{t+1} = a_t + (J_f{}^T J_f)^{-1} J_f{}^T \cdot r(a_t)$$

$$= a_t - J_r{}^+ r(a_t) \qquad\qquad = a_t + J_f{}^+ r(a_t)$$

# back to Multilateration

# Multilateration

- one target, e.g., robot, to be localized *(x,y)*
- *n>3* beacons *i* with known locations $(x_i, y_i)$
- and *n* noisy ranges $d_i$ with error $e_i$

$$d_1^{\ 2} = (x_1 - x)^2 + (y_1 - y)^2$$

$$d_2^{\ 2} = (x_2 - x)^2 + (y_2 - y)^2$$

$$\ldots$$

$$d_n^{\ 2} = (x_n - x)^2 + (y_n - y)^2$$

*hmm, non-linear*

*use Gauss-Newton???*

*there is a better option*

find LS fit for *(x,y)*

# Multilateration

trick: subtract the last equation from the other n-1

$$d_1^{\,2} - d_n^{\,2} = x_1^{\,2} - x_n^{\,2} - 2(x_1 - x_n)\underline{x} + y_1^{\,2} - y_n^{\,2} - 2(y_1 - y_n)\underline{y}$$

$$d_2^{\,2} - d_n^{\,2} = x_2^{\,2} - x_n^{\,2} - 2(x_2 - x_n)\underline{x} + y_2^{\,2} - y_n^{\,2} - 2(y_2 - y_n)\underline{y}$$

$$\ldots$$

$$d_{n-1}^{\,2} - d_n^{\,2} = x_{n-1}^{\,2} - x_n^{\,2} - 2(x_{n-1} - x_n)\underline{x} + y_{n-1}^{\,2} - y_n^{\,2} - 2(y_{n-1} - y_n)\underline{y}$$

=> linear in the unknowns x,y

# Multilateration

Multilateration as Linear Least Squares:

$$A = \begin{pmatrix} 2(x_1 - x_n) & 2(y_1 - y_n) \\ 2(x_2 - x_n) & 2(y_2 - y_n) \\ \multicolumn{2}{c}{\cdots} \\ 2(x_{n-1} - x_n) & 2(y_{n-1} - y_n) \end{pmatrix} \quad b = \begin{pmatrix} x_1{}^2 - x_n{}^2 + y_1{}^2 - y_n{}^2 - d_1{}^2 + d_n{}^2 \\ x_2{}^2 - x_n{}^2 + y_2{}^2 - y_n{}^2 - d_2{}^2 + d_n{}^2 \\ \vdots \\ x_{n-1}{}^2 - x_n{}^2 + y_{n-1}{}^2 - y_n{}^2 - d_{n-1}{}^2 + d_n{}^2 \end{pmatrix}$$

find LLS fit **x***=(x,y)$^T$

$$Ax = b$$

$$x^* = (A^T A)^{-1} A^T b = A^+ b$$

# Ranging

how to measure

distance estimate $\boldsymbol{d_i}$ from $\boldsymbol{x}$ (e.g., robot) to $\boldsymbol{x_i}$ (e.g., beacon)?

- computer vision, e.g.
  - monocular camera: scale of a known object
  - stereo vision: 2 cameras, known baseline, triangulation for range
  - structured light aka RGBD (Kinect, Xtion): depth from known pattern
  - all typically very coarse and noisy
- received signal strength (RSS)
  - typically RF based (WLAN)
  - very coarse and unreliable; multipath/damping/reflection/scattering
  - successful WLAN localization typically uses *fingerprinting*
- time of flight

# Ranging

**time of flight** aka time of arrival (ToA)

- one way
  - given: velocity of signal, (global) time of send, (global) time of arrival
  - challenge: clocks need to be perfectly synchronized
  - alternative: 2 signals, 1 “instantaneous” for sync, see also TDoA
- two way
  - aka round trip: receiver sends signal back to sender
  - including passive reflection (obstacle sensors; see mapping later)
  - given: velocity of signal, (local) time of send, (local) time of arrival
  - note: option to use phase-shift from, e.g., passive reflection
- time difference of arrival (TDoA)
  - use 2 different signal velocities, both send at the same time
  - given: velocities, (local) times of arrival of both signals

# Global Localization, e.g.
# Global Navigation Satellite Systems (GNSS)

- NAVSTAR GPS
  - Navigation Signal Timing and Ranging Global Positioning System
  - “mother of all GNSS” (almost synonym for it)
- GLONASS
  - Global Navigation Satellite System
  - Russian Aerospace Defence Forces
  - 2nd most used GNSS
  - especially good at high latitudes (North, South)
- Galileo
  - EU development
  - European Space Agency (ESA) & European GNSS Agency (GSA)
  - early operation since Dec. 2016, full operation expected in 2020
- BDS aka BeiDou (Big Dipper)
  - BeiDou Navigation Satellite System
  - Chinese development
  - currently in set-up phase, operation expected in 2020

## Global Localization, e.g.

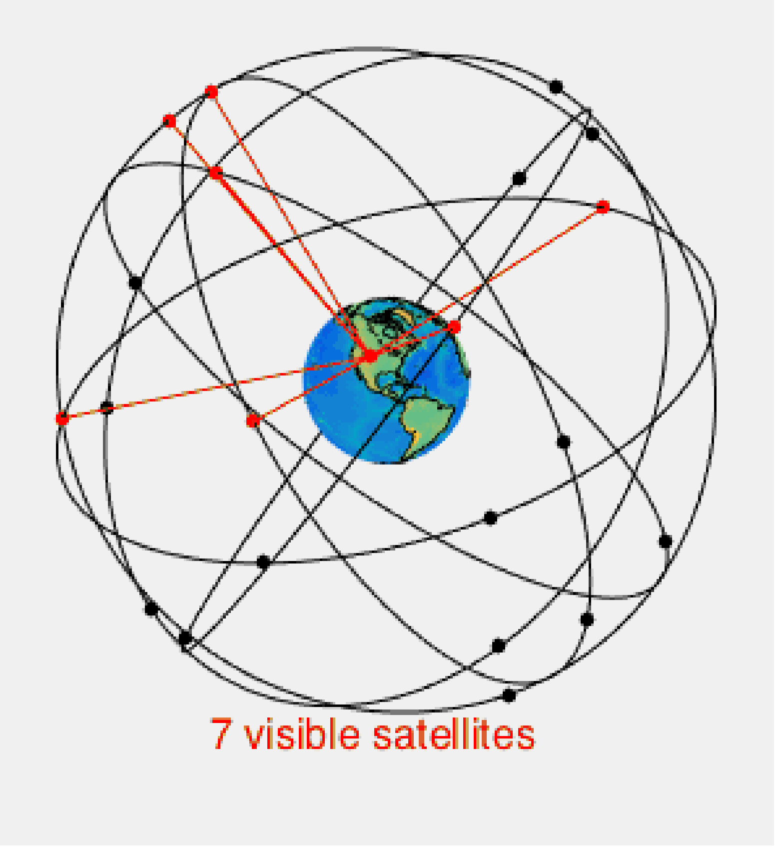
# Global Navigation Satellite Systems (GNSS)

- NAVSTAR GPS
  - Navigation Signal Timing and Ranging Global Positioning System
  - “mother of all GNSS” (almost synonym for it)
- GLONASS
  - Global Navigation Satellite System
  - Russian Aerospace Defence Forces
  - $2^{nd}$ most used GNSS
  - especially good … (North, South)
- Galileo
  - …elopment
  - European Space Agency (ESA) & European GNSS Agency (GSA)
  - early operation since Dec. 2016, full operation expected in 2020
- BDS aka BeiDou (Big Dipper)
  - BeiDou Navigation Satellite System
  - Chinese development
  - currently in set-up phase, operation expected in 2020

all: multi-lateration with satellites as beacons and RF 1-way ToA

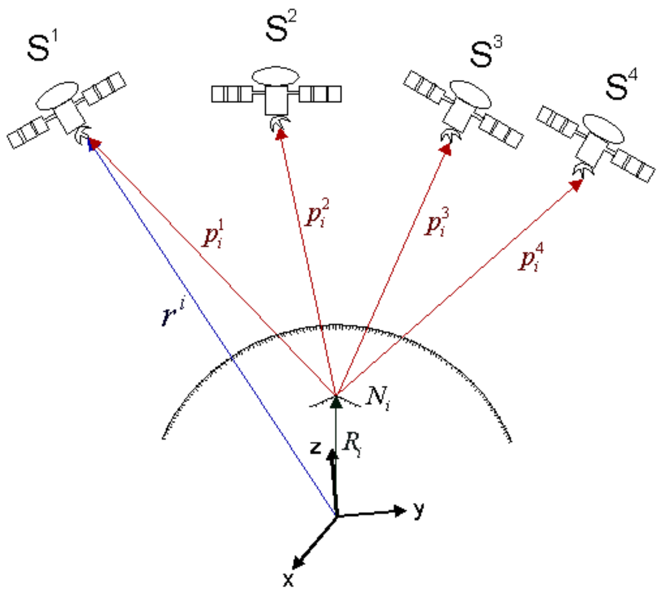
# NAVSTAR GPS

- start 1973, operational 1995
- satellites as beacons
  - multilateration
  - need to “see” min 4 satellites (for longitude, latitude, altitude)
  - 21 can cover the whole earth
  - but the more, the better (smaller LS error)
  - #satellites increasing over time (1994: 24, 2017: 31)

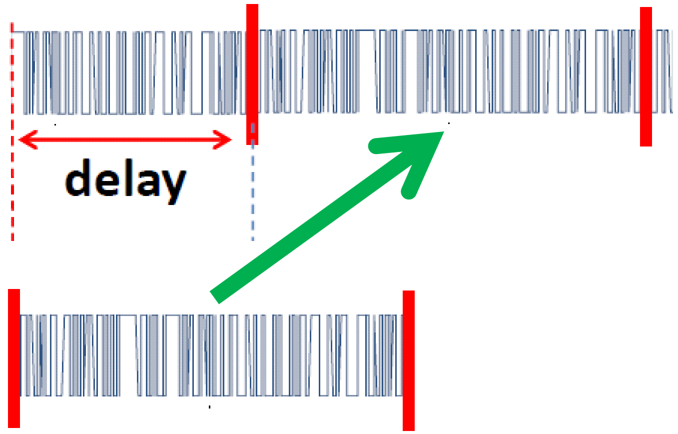


# NAVSTAR GPS

- satellites as beacons
  - they are not stationary
  - hence localized by ground stations
  - satellites broadcast this data to GPS receiver




- ranging
  - 1-way time of flight (ToF)
  - by phase-shift of PN-code
  - challenge: precise clocks
  - satellites: atomic clocks (compensation for relativistic effects)
  - receiver: init from satellites (optimization by error minimization)

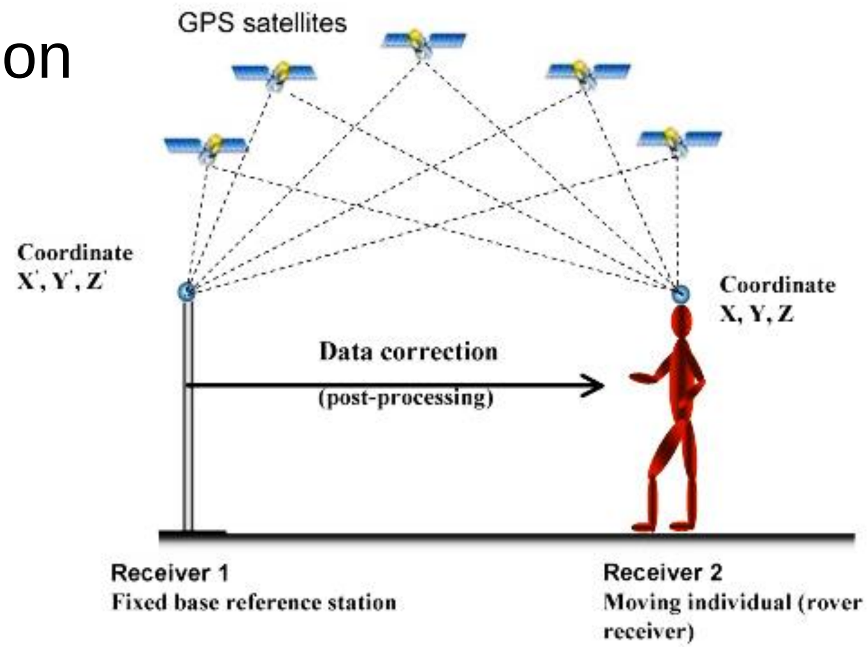



best case accuracy: ~3 meters

# Differential GPS (DGPS)

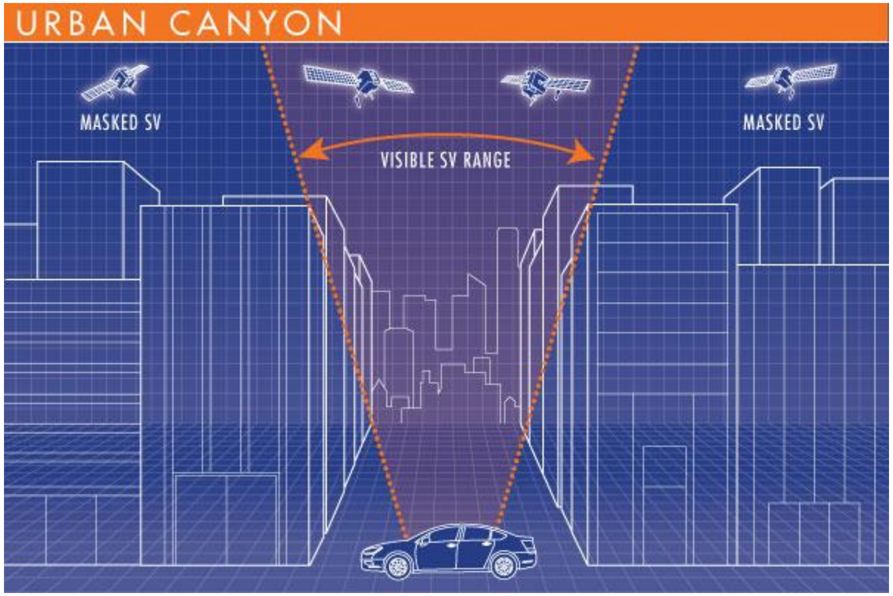
fixed ground stations

- to compute compensation
- for slow changing error sources
  - atmospheric effects
  - imprecise satellite localization
  - clock errors



# Limitations of GPS

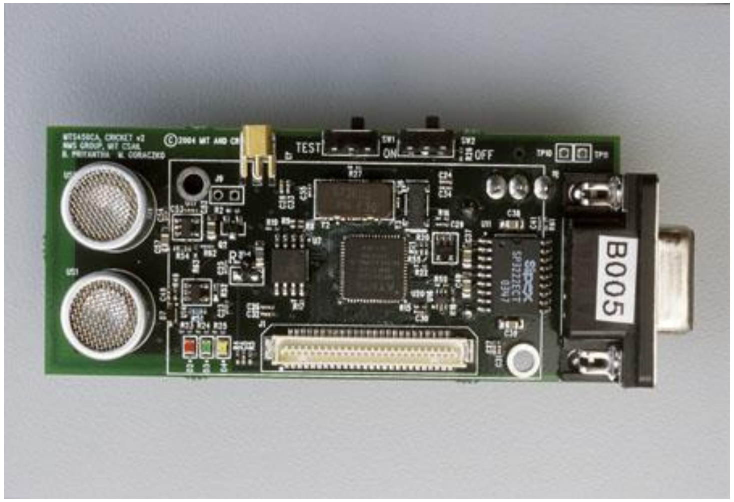
- even DGPS is still quite coarse
- GPS does not work indoor
- limitations outdoor
  - vegetation (under / near trees)
  - urban canyons (near (high) buildings)



# Example: Global Indoor Localization

## MIT Cricket System

- fixed beacons
- with TDoA range
  - RF (sync)
  - ultrasound
- performance
  - 1-3 cm accuracy
  - but speed of sound
    - is „slow“ (in air: ~343 m/sec), hence “slow” updates
    - is not constant, i.e., depends on temperature, humidity, etc.

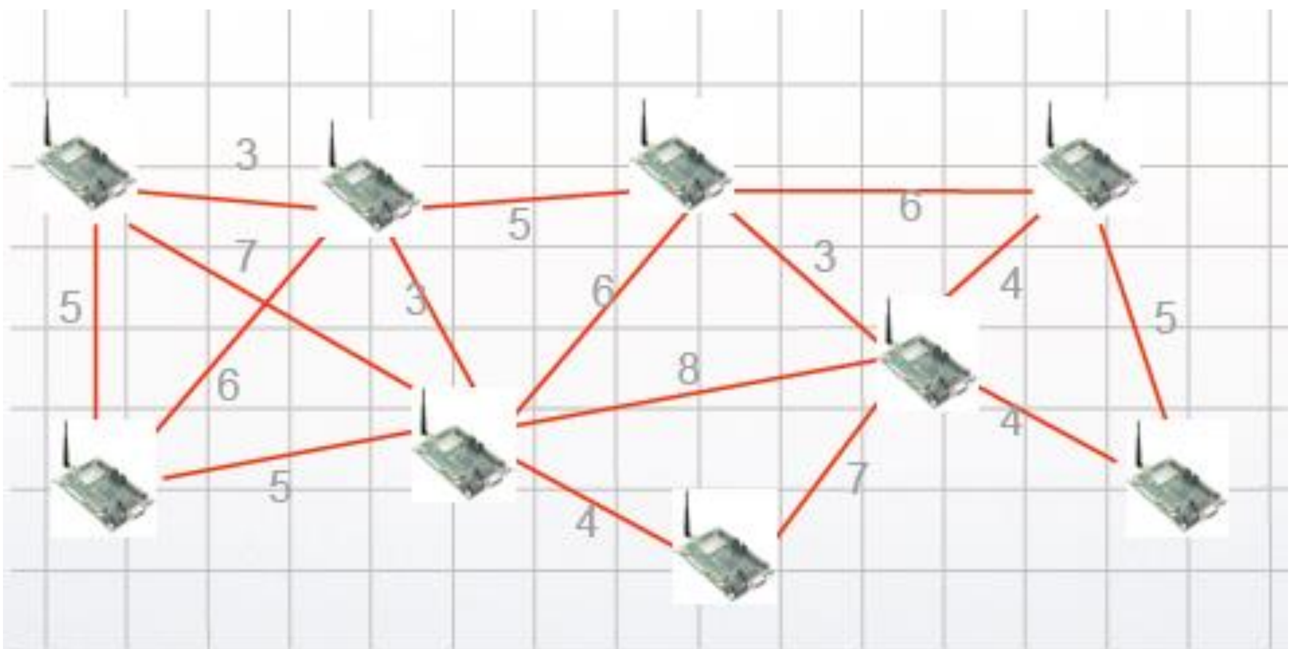



# Beacon-Free
# (Global) Localization

# Beacon-Free Localization

- lack of infrastructure,
- i.e., no beacons with known poses
- e.g., sensor networks

given ranges, find topology (graph layout)



# Multidimensional Scaling (MDS)

- origin in social sciences, psychology
- given: m-dimensional relations between n entities
- i.e., matrix D with noisy distances $d_{ij}$
- find: coordinates matrix X

localization:

- m = 2 or 3
- relation = Euclidean distance (aka metric MDS)

# Multidimensional Scaling (MDS)

- n (m-dim) locations:

$$x_k = \left(x_{1k}, \cdots, x_{mk}\right)$$

- coordinates matrix X

$$X = \left(x_{ij}\right) = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

- noisy metric distances $d_{ij}$:

$$d_{ij} = \sqrt{\sum_{k=1}^{m} \left(x_{ik} - x_{jk}\right)^2}$$

note: $d_{ij} = d_{ij}$ , i.e., D is symmetric (non-symmetric $d_{ij} \neq d_{ji}$ : use average)

# Multidimensional Scaling (MDS)

**3D**

- n (m=3) locations:

x y z

$$x_k = (x_{1k}, x_{2k}, x_{3k})$$

coordinate

- coordinates matrix X

$$X = \left(x_{ij}\right) = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

- noisy metric distances $d_{ij}$:

$$d_{ij} = \sqrt{\sum_{k=1}^{3} \left(x_{ik} - x_{jk}\right)^2}$$

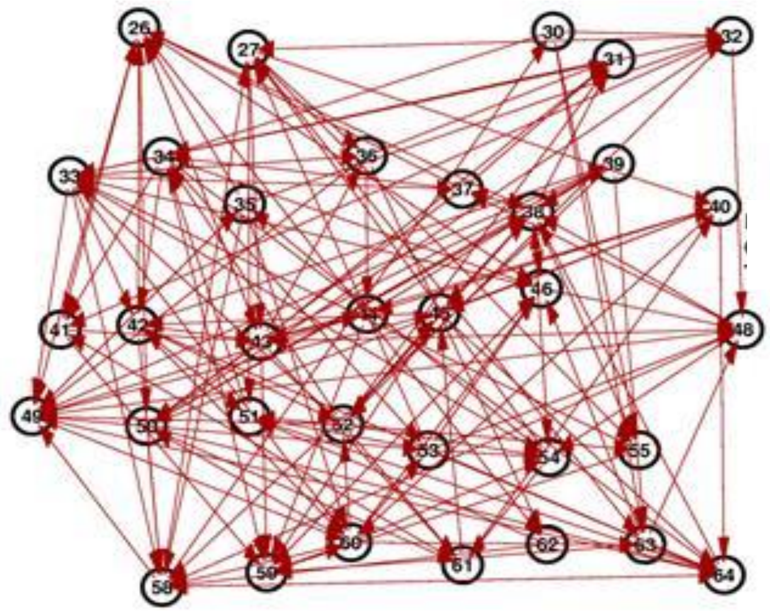
note: $d_{ij} = d_{ij}$ , i.e., D is symmetric (non-symmetric $d_{ij} \neq d_{ji}$ : use average)

# Multidimensional Scaling (MDS)

so, we want to find X with

$$XX^T = \left(d_{ij}{}^2\right) := D^{(2)}$$

(note: $D^{(2)} \neq D^2$)



- noisy $d_{ij}$ : like springs between locations
- MDS minimizes the stress (LS error of locations)
- important trick
  - centroid of the $x_k$ as origin
  - via method known as **double-centering**

# Multidimensional Scaling (MDS)

nxn **centering** matrix C:

$$C_{(n)} = I - \tfrac{1}{n}\left(1_{ij}\right) = I - \tfrac{1}{n}11^T$$

$$C_1 = (1) - (1) = (0)$$

$$C_2 = I - \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

$\ldots$

mxn matrix A: removing the means

- from all n columns of A: $A' = C_m A$
- from all m rows of A : $A' = AC_n$

# Multidimensional Scaling (MDS)

**double centering**

(row & column vectors sum up to 0)

$$A' = CAC$$

or: $a'_{ij} = a_{ij} - a_{*j} - a_{i*} + a_{**}$

- row average $\quad a_{*j} = \frac{1}{n} \sum_{i=1}^{n} a_{ij}$
- column average $\quad a_{i*} = \frac{1}{n} \sum_{j=1}^{n} a_{ij}$
- total average $\quad a_{**} = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}$

# Multidimensional Scaling (MDS)

double centering & MDS

- we want to double center X, but we have only D
- but it can be shown for the double-centered $D^{(2)}$'

$$D^{(2)}{}' = \left( d_{ij}{}^{2}{}' \right) \qquad d_{ij}{}^{2}{}' = d_{ij} - d_{*j} - d_{i*} + d_{**} = -2 \sum_{a=1}^{m} x_{ia} x_{ja}$$

so, $-\frac{1}{2} D^{(2)}{}' := B = XX^{T}$

# Multidimensional Scaling (MDS)

D is symmetric,

- hence $D^{(2)}$ symmetric
- hence double-centered $D^{(2)}$' symmetric
- hence B (= -1/2 $D^{(2)}$') symmetric

**SVD** of ***symmetric*** matrix **A** = $\mathbf{VLV}^T$

$$XX^T = -\tfrac{1}{2} D^{(2)}{}' = B = VLV^T$$

$$\Rightarrow X = VL^{(1/2)}$$

# Multidimensional Scaling (MDS)

MDS algorithm

1. set matrix A $\qquad A = -\frac{1}{2} D^{(2)}$
2. set matrix B (double-centering) $\qquad B = CAC$
3. compute SVD of B $\qquad B = VLV^{T}$
4. get X (sqrt of diagonal L) $\qquad X = VL^{(1/2)}$

# Multidimensional Scaling

Implementation

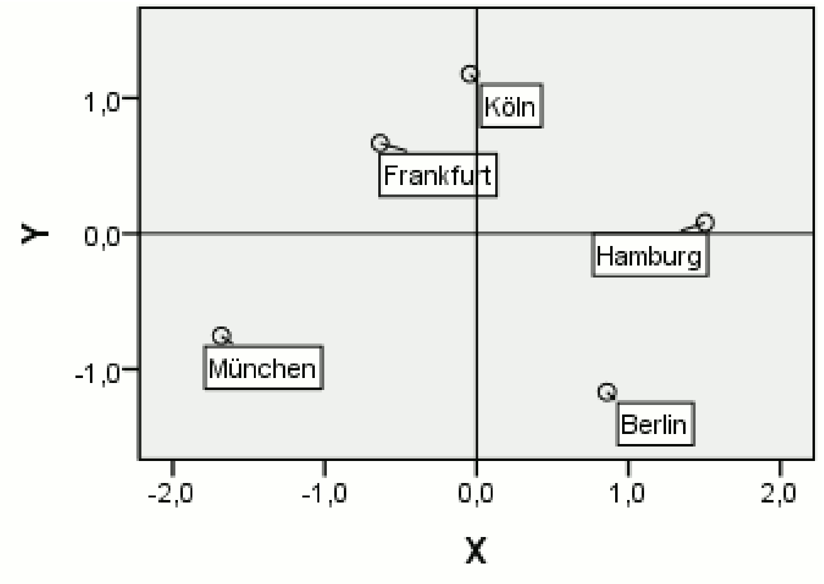
- MATLAB
  - Matlab Toolbox for Dimensionality Reduction
  - http://ict.ewi.tudelft.nl/~lvandermaaten/Matlab_Toolbox_for_Dimensionality_Reduction.html
- Orange
  - C++ data mining toolbox; Python scripting
  - http://www.ailab.si/orange/...
  - orngMDS module: .../doc/modules/orngMDS.htm
- DIY
  - mainly GSL for SVD; that‘s all... ☺

# MDS & missing ranges

MDS, best case:
all ranges given

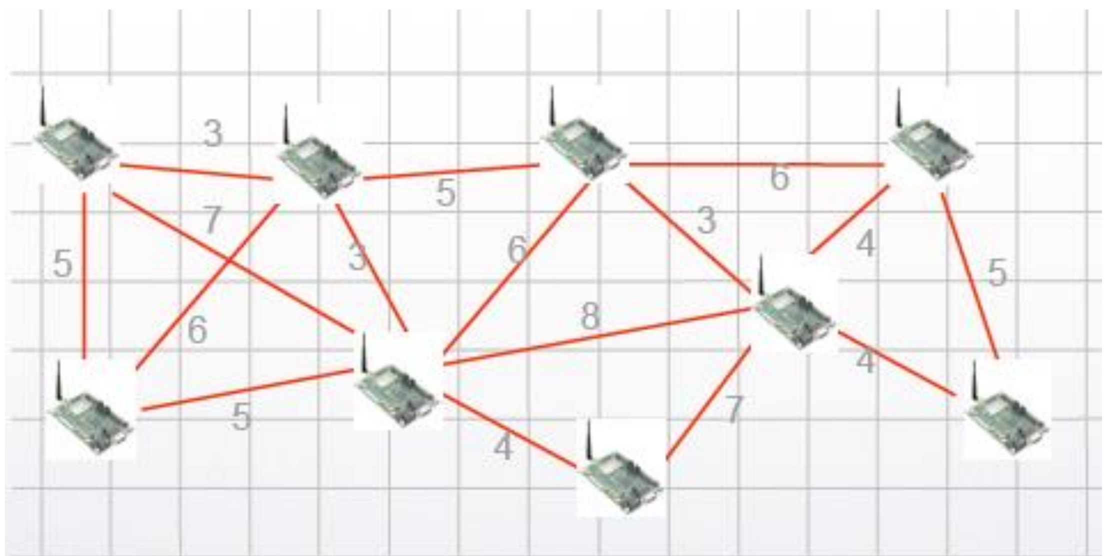
| *D=($d_{ij}$)* | Berlin | Frankfurt | Hamburg | Köln | München |
|---|---|---|---|---|---|
| **Berlin** | 0 | 548 | 289 | 576 | 586 |
| **Frankfurt** | 548 | 0 | 493 | 195 | 392 |
| **Hamburg** | 289 | 493 | 0 | 427 | 776 |
| **Köln** | 576 | 195 | 427 | 0 | 577 |
| **München** | 586 | 392 | 776 | 577 | 0 |

| *City* | X | Y |
|---|---|---|
| **Berlin** | 0,8585 | -11,679 |
| **Frankfurt** | -0,6363 | 0,6660 |
| **Hamburg** | 15,036 | 0,0800 |
| **Köln** | -0,0438 | 11,760 |
| **München** | -16,821 | -0,7542 |



# MDS & missing ranges

- MDS assumes all ranges given
- i.e., fully connected graph
- what if some $d_{ij}$ are missing?



# MDS & missing ranges

option: try to estimate the unknown ranges $d_{ij}$

e.g., MDS-MAP

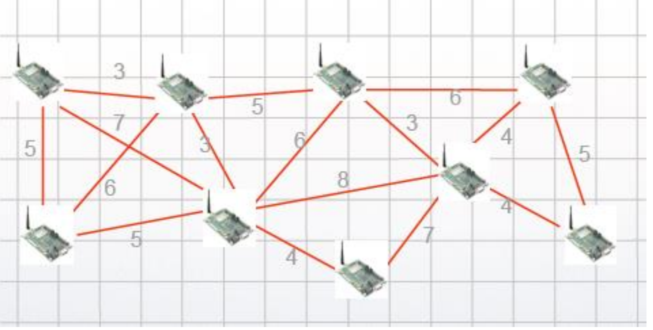
- compute shortest path between all pairs
- upper bound for missing distances
- especially popular in sensor networks
  - network distances often via connectivity,
  - i.e., #hops anyway

# MDS & missing ranges

further option: just "ignore" missing values

- i.e. use any estimate for $d_{ij}$ if available
- or simply $d_{ij} = 0$

as there is much structure in D
(think of spring analogue: few springs can make it “rigid”)




more concretely:

- D has a very low rank r
  - i.e., low dimension of the vector space
  - spanned by its columns / rows
- namely, for m-dim MDS: r = m+2
- i.e., 2D: r=4, 3D: r=5

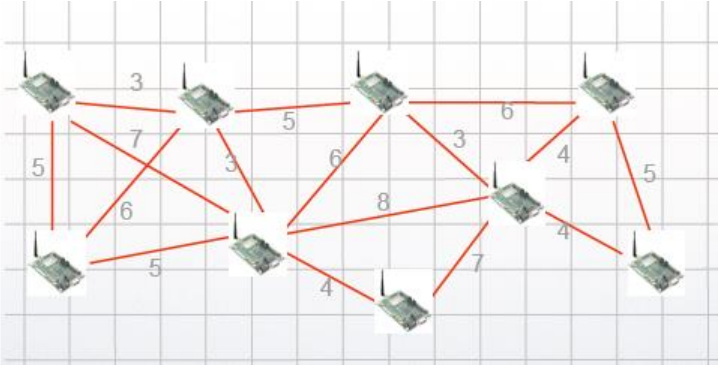
# MDS & missing ranges

D has low rank (2D: r=4, 3D: r=5)

- replace D by its rank r approximation D'
- e.g., via SVD (Eckart–Young theorem)
  - $D = USV^T$, $D' = US'V^T$
  - with S' contains r largest singular values from S

of course, the more estimates the better the precision

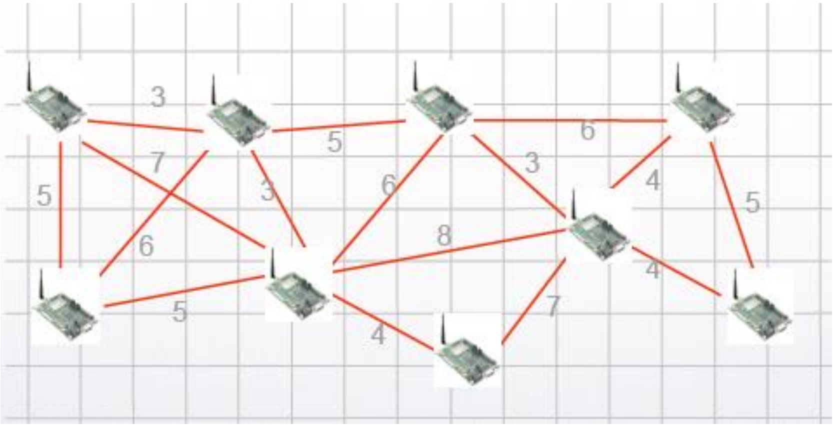
- i.e., do SVD,
- choose "best possible" rank,
- i.e., all "large" singular values



# MDS & missing ranges

note:

- for symmetric matrix ($B = -1/2\ D^{(2)\prime}$):
- singular values $s_i$ and eigenvalues $\lambda_i$ are closely related
- namely $s_i = |\lambda_i|$
- hence
  - both MDS itself as well as rank approximation
  - can be addressed via eigenvalue decomposition

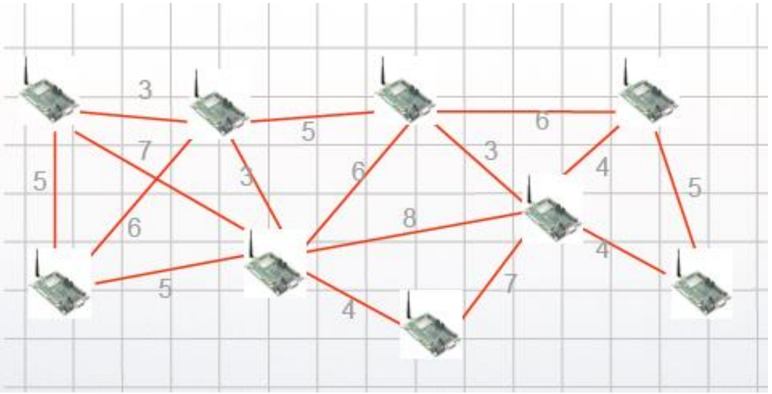


# MDS and global frame

- MDS based on relative measurements
- origin is the centroid of the resulting coordinates
- obviously global orientation undetermined
- but also ***reflections*** possible

hence often

- use of ***anchor nodes***
- i.e., globally localized nodes (beacons)
- to get global reference frame



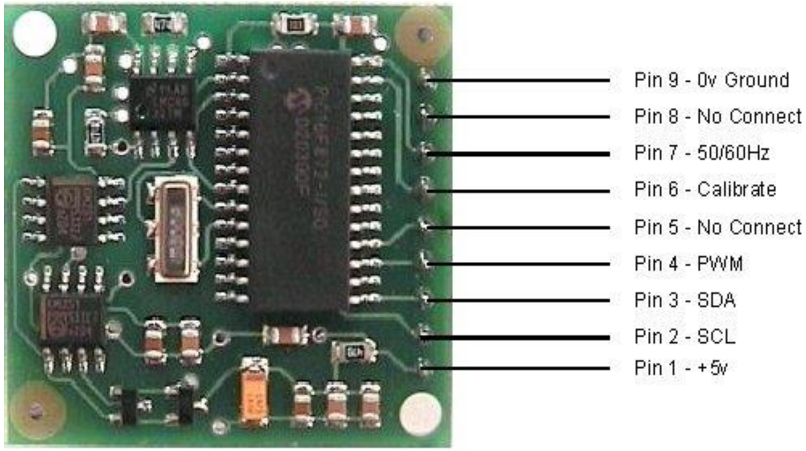
# Orientation Sensors

# Digital Compass

- absolute orientation in 1 DoF, i.e., yaw
- but many error sources
  - on robot: motors, computer, ...
  - environment: computers, electrical machines, ...

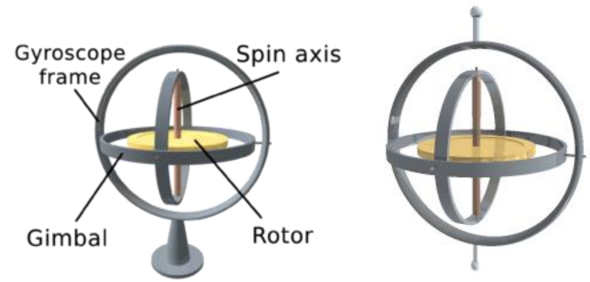
example: Philips KMZ51

- 0.1 deg resolution

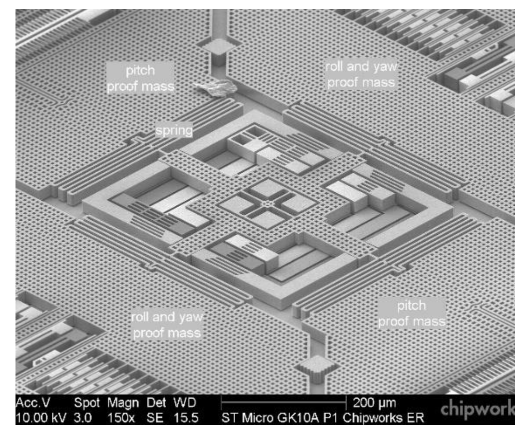


# Gyroscope

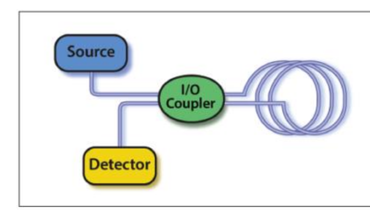
- gimbal
  - classical mechanic
  - using spinning mass (moment of inertia)
  - today not used anymore

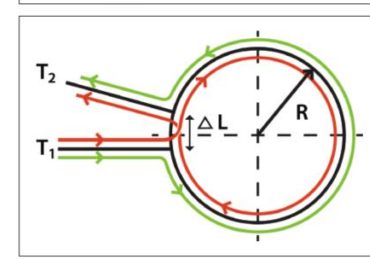



- vibrating structure
  - micro electro-mechanical system (MEMS)
  - small ICs
  - measures rate of turn
  - relative precise, drift compensation (especially temperature) needed

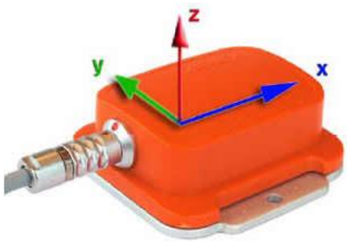



- fiber optic
  - two light paths in opposite directions
  - using spool of glass-fiber
  - rotation changes length of the paths
  - measurement by interference
  - very precise but costly

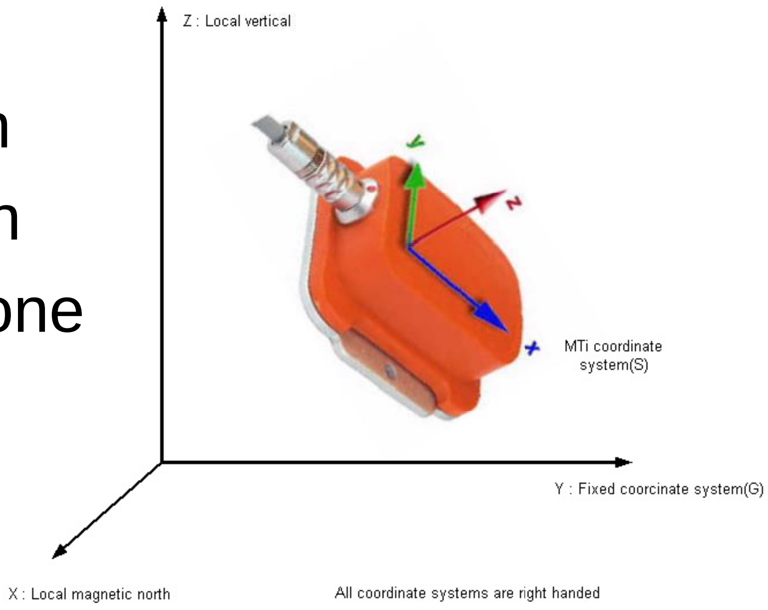
# Inertial Measurement Unit (IMU)




6 degrees of freedom (DOF)
- 1 digital compass
- 3 MEMS gyros
- 3 accelerometers (gravity vector)

but mainly 3-DOF global orientation
- double integration of acceleration
- for location is extremely error prone

example: XSens Mti



# Relative Localization
# Odometry / Dead-Reckoning

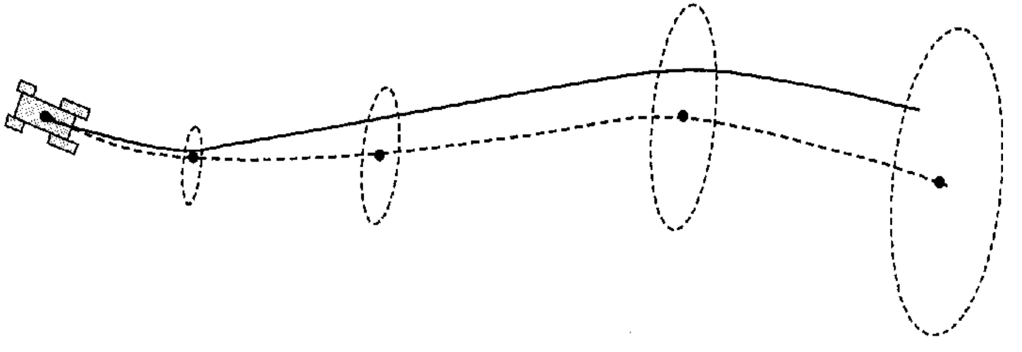
# Dead Reckoning / Odometry

given orientation & speed (& time)
=> vector addition for pose estimate

- **dead reckoning** from "deduced reckoning"
- very old technique from sailing

aka **odometry**

- (cumulative) use of mobile robot forward kinematics



problem: errors accumulate
(especially angular errors are significant)

# Dead Reckoning / Odometry

given orientation & speed (& time)
=> vector addition for pose estimate

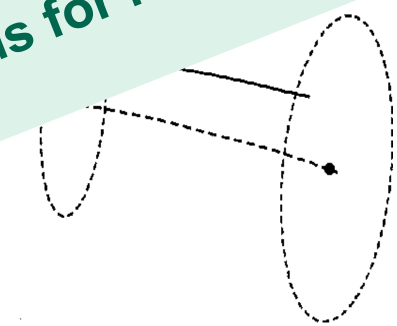
- **dead reckoning** from “deduced reckoni
- very old technique from sailing

aka **odometry**

- (cumula

hence:
- need to model errors,
- keep track of them,
- and minimize when possible

=> **probabilistic methods for localization**



problem: er s accumulate
(especially angular errors are significant)