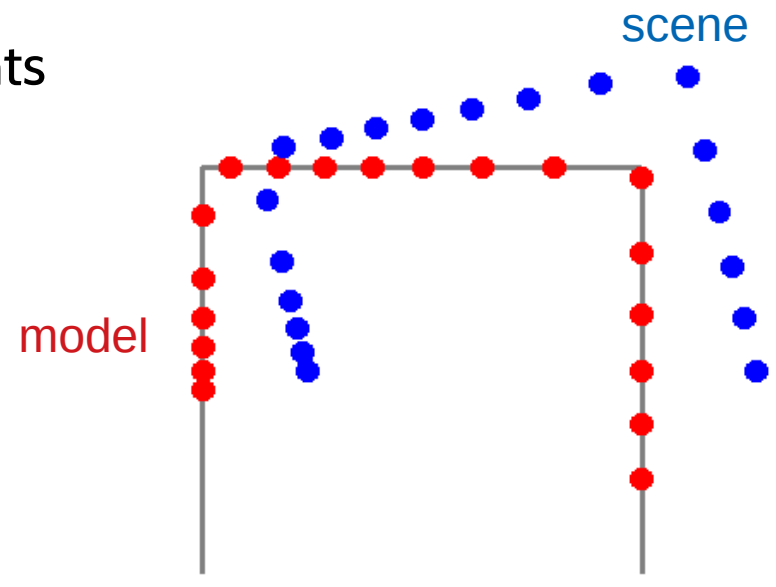
MOBILE ROBOTICS

Iterative Closest Point (ICP)

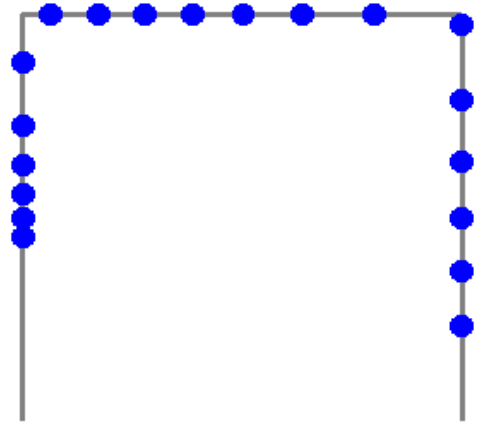
Introduction

- Find the transformation between a model (reference, target) and a scene (source)
  - model and scene: two sets of points
  - Transformation: (x, z, θ)



Introduction

- Find the transformation between a model (reference, target) and a scene (source)
  - model and scene: two sets of points
  - Transformation: (x, z, θ)

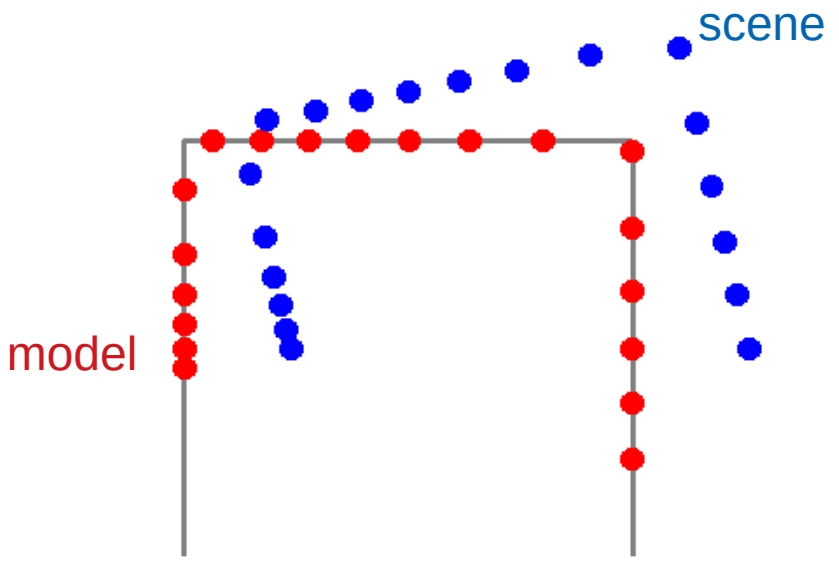


Basic ICP Steps

- Associate the points in the scene to the points in the model (**closest points** may be associated together)
- Estimate the scene pose (translation & rotation) that minimizes the distance between the model and the scene
- Transform the scene according to the previously computed pose
- **Iterate**

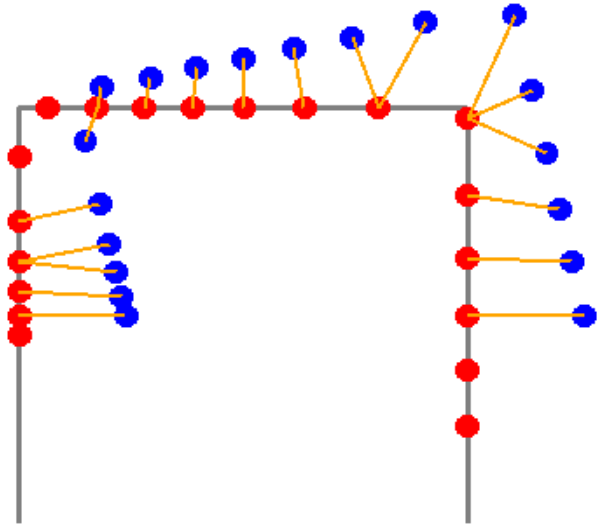
Basic ICP Steps - Example

- model and scene



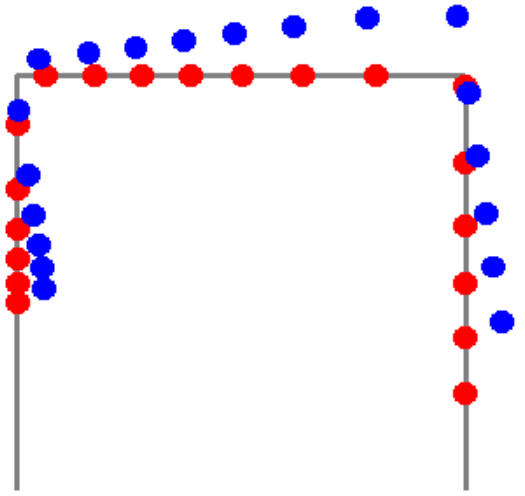
Basic ICP Steps - Example

- Association



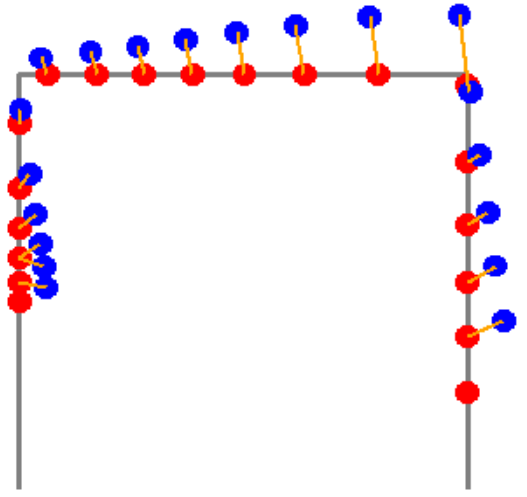
Basic ICP Steps - Example

- Estimate the transformation
  - transform the scene



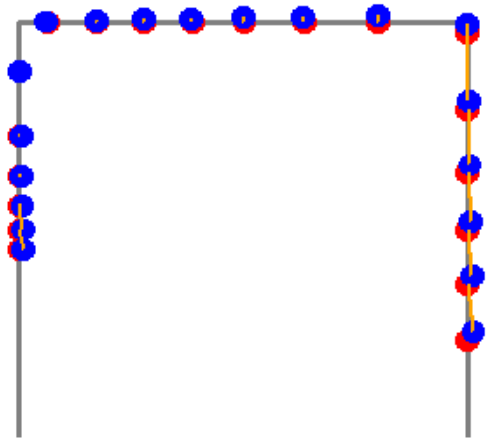
Basic ICP Steps - Example

- Iterate...



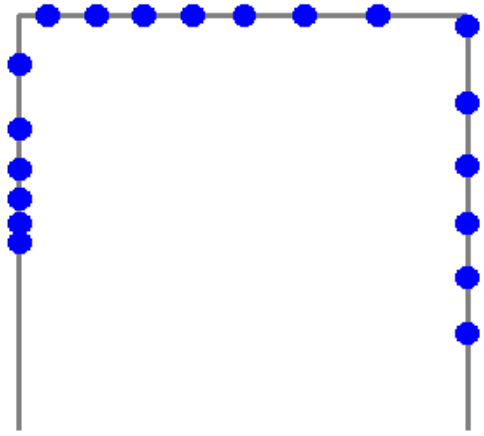
Basic ICP Steps - Example

- Iterate...



Basic ICP Steps - Example

- Iterate...



Association

- The association step provides a association between the points of the model and the points of the scene
- At each point of the scene must be associated one and only one point in the model
- The Association class is used, an association corresponds to a
  - index in the model set
  - index in the scene set
  - distance (between the point of the model and the point of the scene)

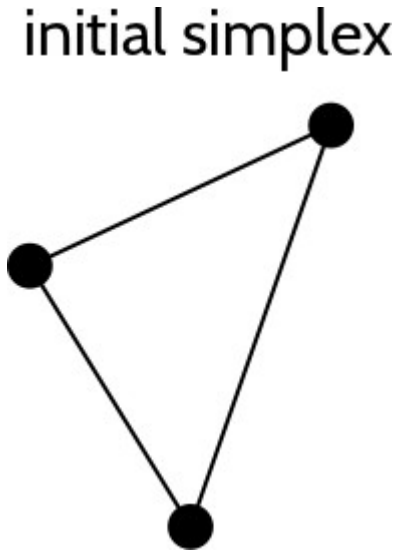
Estimate the “best” pose

- Once the association is done the idea is to find the best pose for the scene that will minimize the distance between the model and the scene (according to the associated points)
  - Several algorithms can be used
  - Nelder and Mead algorithm

Nelder and Mead

- The Nelder–Mead method (also downhill simplex method, amoeba method, or polytope method) is a commonly applied numerical method used to find the minimum or maximum of an objective function
  - wikipedia
- It uses the concept of simplex
  - dimension 3 in our case => “a pyramid”
  - Each “summit” of the simplex is called a vertex (4 vertices in our case)
  - A vertex correspond to a pose (3D pose in our case)

Nelder and Mead

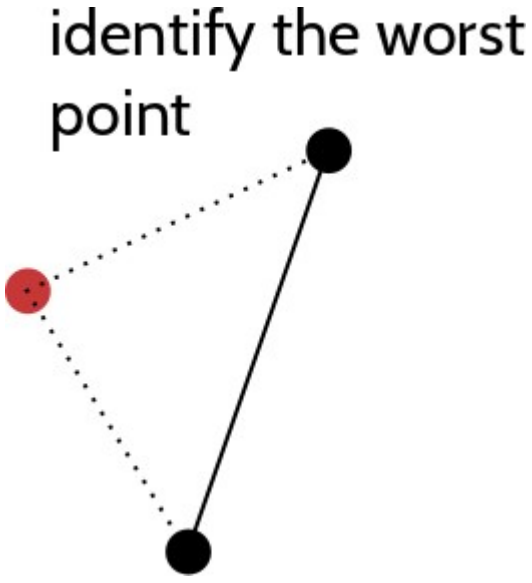


We want to find **x** that minimize the function f(**x**)
(f being the cost function)

The current test points (initial simplex) are $\mathbf{x}_1$, $\mathbf{x}_2$, … , $\mathbf{x}_{n+1}$

n being the dimension of the simplex
- n=2 in the example depicted on the left
- n=3 for our ICP implementation

Nelder and Mead



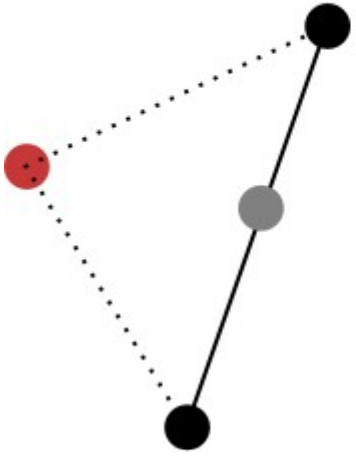
STEP 1

Order the vertices according to their cost:
$f(\mathbf{x}_1) <= f(\mathbf{x}_2) <= \ldots <= f(\mathbf{x}_{n+1})$

The worst vertex being $f(\mathbf{x}_{n+1})$

Nelder and Mead

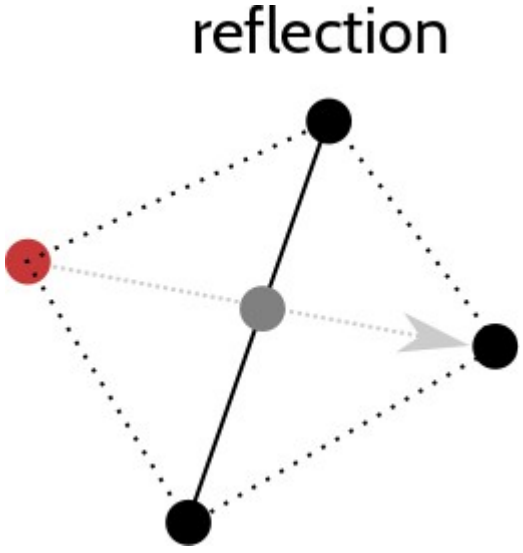
get the center of gravity
(without the worst point)



STEP 2

Calculate $\mathbf{x}_o$, the centroid of all points except $\mathbf{x}_{n+1}$

Nelder and Mead



STEP 3

Compute the reflected point $\mathbf{x}_r = \mathbf{x}_o + \text{alpha}(\mathbf{x}_o - \mathbf{x}_{n+1})$

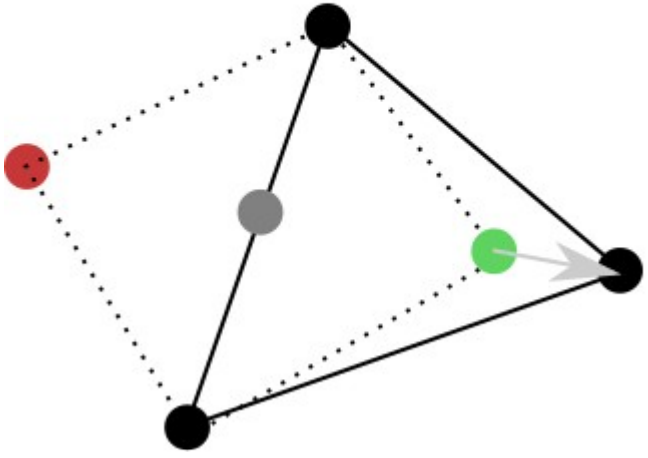
If the reflected point is better than the second worst but not better than the best : $f(\mathbf{x}_1) <= f(\mathbf{x}_r) <= f(\mathbf{x}_n)$

then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ with the reflected point $\mathbf{x}_r$ and go to step 1

Otherwise go to the next step

Nelder and Mead – 2D example

the reflected point is good
=> Expansion



STEP 4

If the reflected point is the best so far f($\mathbf{x}_r$) <= f($\mathbf{x}_1$)

compute the expanded point
$\mathbf{x}_e = \mathbf{x}_o + \text{gamma}(\mathbf{x}_r - \mathbf{x}_o)$
If the expanded point is better than
the reflected one f($\mathbf{x}_e$) < f($\mathbf{x}_r$)

obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ with the expanded point $\mathbf{x}_e$ and
go to step 1

else

obtain a new simplex by replacing the wost point $\mathbf{x}_{n+1}$ with the reflected point $\mathbf{x}_r$ and
go to step 1

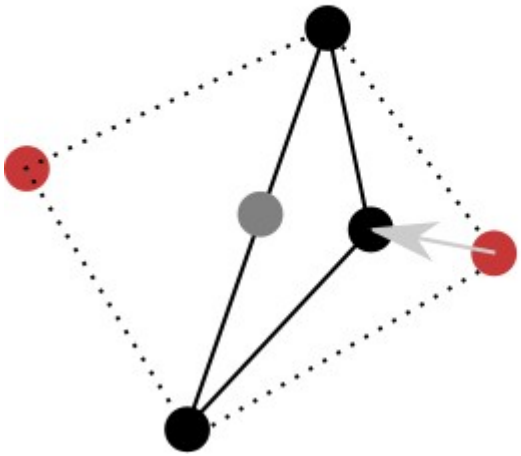
else

go to next step

Nelder and Mead – 2D example

the reflected point is worst

=> contraction



STEP 5

Here we have $f(\mathbf{x}_r) >= f(\mathbf{x}_n)$

Compute contracted point $\mathbf{x}_c = \mathbf{x}_o + rho(\mathbf{x}_{n+1} - \mathbf{x}_o)$

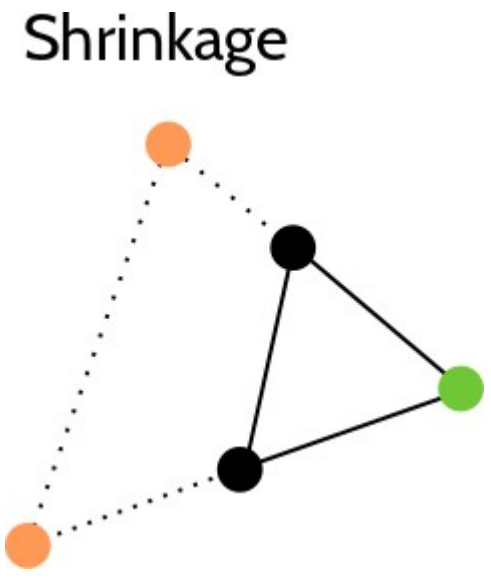
if the contracted point is better than the worst point $f(\mathbf{x}_c) < f(\mathbf{x}_{n+1})$

then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1}$ by the contraced point $\mathbf{x}_c$ and go to step 1

else

go to next step

Nelder and Mead – 2D example



STEP 6

Replace all points except the best ($\mathbf{x}_1$) with
$\mathbf{x}_i = \mathbf{x}_1 + \text{sigma}(\mathbf{x}_i - \mathbf{x}_1)$

go to step 1

Work to do

- Files to send by email
  - tp_icp/icp.py
  - tp_icp/simplex.py
  - tp_icp/vertice.py
- Warnings
  - Test your code functions after functions, you can use the "Test" button as you want
  - Do not modify the other files
    - You may however modify the btn_test_event() function in tp_icp/simulator.py for your tests
  - Do not add any library (numpy for instance…)
- You can use run_icp_tests to run unit tests

associate_by_index()

- Associate the points of the model and the scene according to the indexes of the points
  - first point of the model is associated to the first point of the scene and so on...

associate_by_closest()

- Associate the points of the scene to the points of the model according to the euclidean distance
  - the points of the scene are associated to the closest points of the model
  - A point of the model can be associate to several points of the scene
    - Some points of the model may not be associated at all

update_cost()

- This function computes the distance between the model and the scene according to the pose of the vertex
  - in other words, the distance between the model and the scene when the scene is transformed with the self.pose transformation
  - Note that in our case a transformation is two translations (x, z) and one rotation (theta): a pose
  - This distance corresponds to the cost of the vertex
  - To compute a new point according to a vertex

$$\mathbf{x} = (x, z)^T$$
$$\mathbf{vertex} = (tx, tz, \Theta)$$
$$\mathbf{new\_x} = (\cos(\Theta)x - \sin(\Theta)z + tx, \sin(\Theta)x + \cos(\Theta)z + tz)^T$$

sort_vertices()

- Function that sorts the vertices of the simplex:
  - sort the self.vertices list, ascending order

update_cost()

- Function that updates the cost of all the vertices of the simplex
  - loop over the vertices list to update the cost of each vertex

get_centroid()

- Function that returns the centroid of the simplex computed with all the vertices **except the worst one**
  - It is resumed in computing the average vertex based on all the verticesT

nelder_and_mead()

- Function that returns pose that minimize the distance between the model and the scene according to the association

tp_icp/icp.py

Nelder and Mead Algotithm summary

stopping criteria:
- maximal number of total iterations
- maximal number of iterations without updating the best vertex

- sort the vertices: $f(\mathbf{x}_1) <= f(\mathbf{x}_2) <= f(\mathbf{x}_3) <= f(\mathbf{x}_4)$
- get the centroid $\mathbf{x}_o$
- get the reflected point $\mathbf{x}_r = \mathbf{x}_o + \text{alpha}(\mathbf{x}_o - \mathbf{x}_4)$
- if $f(x1) <= f(\mathbf{x}_r) < f(x3)$
  - $\mathbf{x}_4 = \mathbf{x}_r$ and loop again
- else if $f(\mathbf{x}_r) < f(\mathbf{x}_1)$
  - get the expanded point $\mathbf{x}_e = \mathbf{x}_o + \text{gamma}(\mathbf{x}_r - \mathbf{x}_o)$
  - if $f(\mathbf{x}_e) < f(\mathbf{x}_r)$
    - $\mathbf{x}_4 = \mathbf{x}_e$ and loop again
  - else
    - $\mathbf{x}_4 = \mathbf{x}_r$ and loop again
- else (here we have $f(\mathbf{x}_r) >= f(\mathbf{x}_n)$)
  - get the contracted point $\mathbf{x}_c = \mathbf{x}_o + \text{rho}(\mathbf{x}_4 - \mathbf{x}_o)$
  - if $f(\mathbf{x}_c) < f(\mathbf{x}_4)$
    - $\mathbf{x}_4 = \mathbf{x}_c$ and loop again
  - else
    - replace all vertices (except the best one) with
    - $\mathbf{x}_i = \mathbf{x}_1 + \text{sigma}(\mathbf{x}_i - \mathbf{x}_1)$, i = 2,3,4