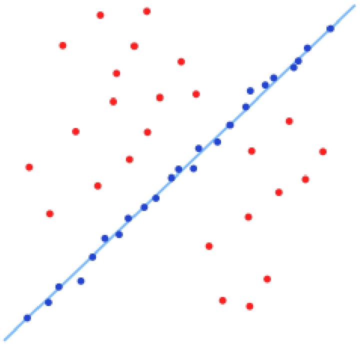
# Probabilistic Robotics Course

# RANSAC

**Giorgio Grisetti**
grisetti@diag.uniroma1.it

Department of Computer, Control and Management Engineering
Sapienza University of Rome

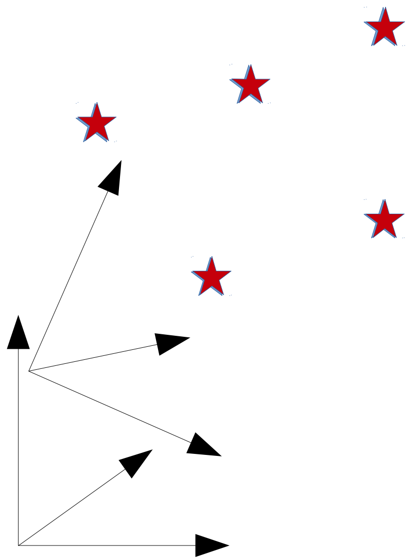


# 3D Point Registration

Unknown correspondences and initial guess!

# 3D Point Registration

We want to find a transform that minimizes the distance between corresponding points

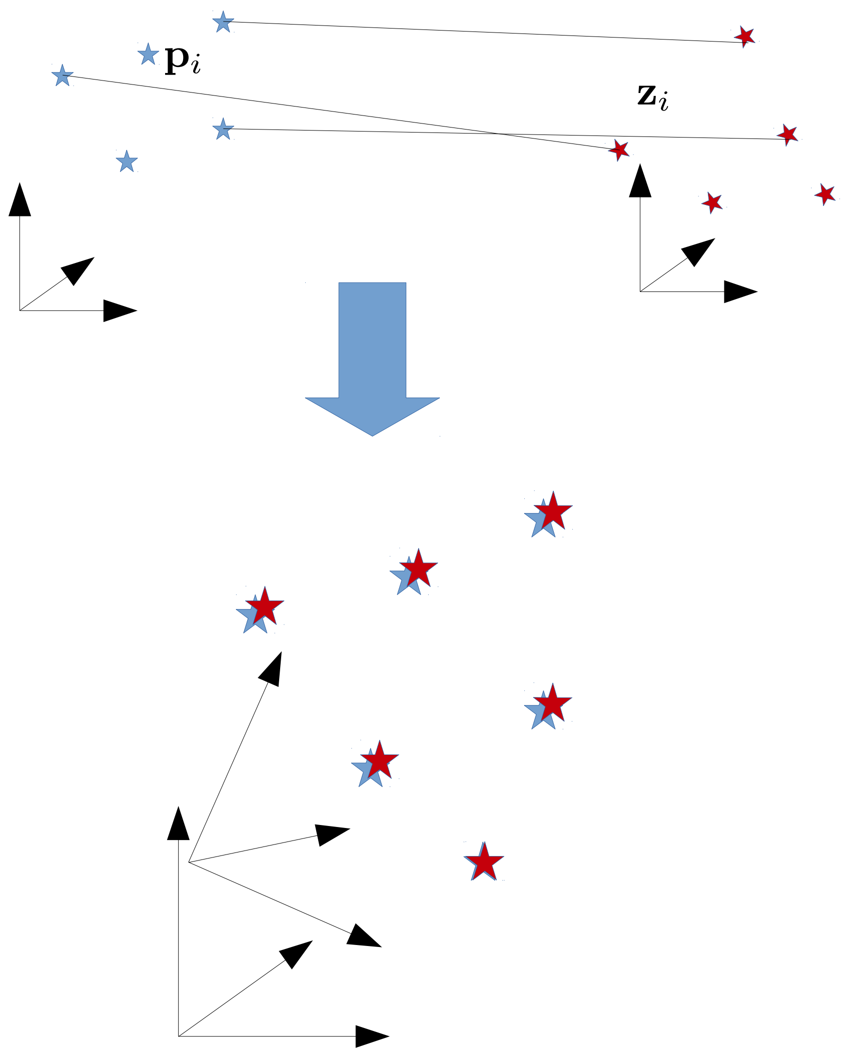


# What if...

we knew a minimal set of correct correspondences?

We could:

- **find an initial guess of our system by using the tools we have learned so far**
- with this initial guess, we could find more “good” correspondences
- we could determine the solution by considering all “good” correspondences, and dropping the bad ones

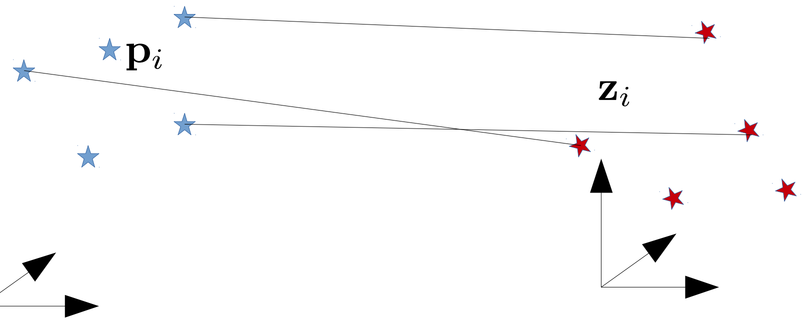


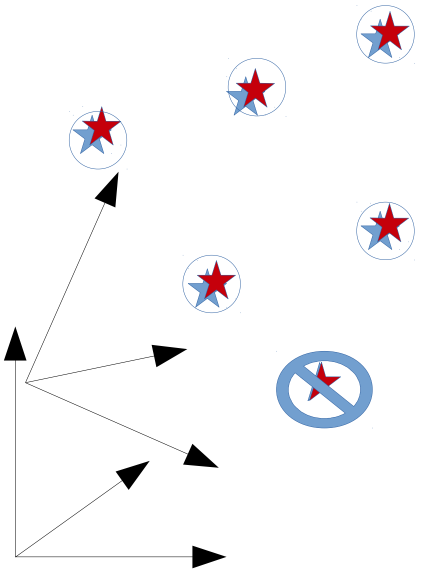
# What if...

we knew a minimal set of correct correspondences?

We could:

- find an initial guess of our system by using the tools we have learned so far
- **with this initial guess, we could find more "good" correspondences**
- we could determine the solution by considering all "good" correspondences, and dropping the bad ones



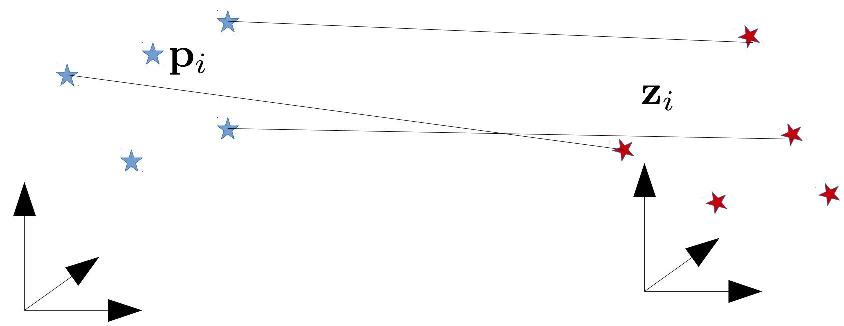



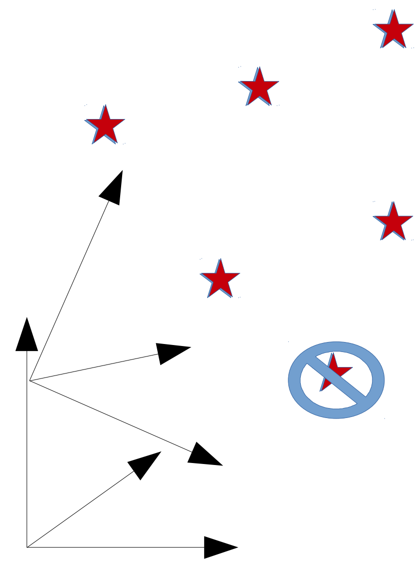
# What if…

we knew a minimal set of correct correspondences?

We could:

- find an initial guess of our system by using the tools we have learned so far
- with this initial guess, we could find more “good” correspondences
- **we could determine the solution by considering all “good” correspondences, and dropping the bad ones**



# RANSAC

Random Sample Consensus

For N times:

- Sample (randomly) a minimal set of correspondences among the candidate ones
- With this minimal set compute an initial alignment
- Use this alignment to determine the number of good/bad correspondences
- Compute the “consensus” of the guess as a function of number of inliers and error
- Repeat the above steps N times, and at each time keep the “best” solution

When done, use only inliers to improve the final solution.

# RANSAC: What do we need

RANSAC is a schema to seek for a good solution, not a “closed” algorithm

To implement the schema we need:

- A procedure to seek for correspondences (the better the procedure, the less iterations N are needed)
- A procedure to smartly select a set of pseudo-random (worst case: uniform) set between candidate correspondences
- A procedure to compute a solution, immune to poor initial guesses
- A procedure to count the inliers

# Correspondences

The correspondence search is usually done exploiting the appearance of features.

- This usually leads to few “good” correspondences.
- The correspondence search is characterized by an “inlier ratio” $w$, that is the number of good matches divided the number of points in the model.

In this example:

- we assume “all” points match with all other points, so we have N*N correspondences (worst case)
- the inlier ratio is 1/N (pretty bad)

# Determining a Solution

In this case, the initial guess is poor.

- We prefer linear relaxation to compute the initial solution, as it is immune from the initial guess.

$$\mathbf{x}^T = (\mathbf{r}_1^T\ \mathbf{r}_2^T\ \mathbf{r}_3^T\ \mathbf{t}^T)$$

$$\begin{aligned}
\mathbf{h}^{[i]}(\mathbf{x}) &= \mathbf{R}\mathbf{p}^{[i]} + \mathbf{t} \\
&= \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{pmatrix} \mathbf{p}^{[i]} + \mathbf{t} \\
&= \underbrace{\begin{pmatrix} \mathbf{p}^{[i]T} & & \\ & \mathbf{p}^{[i]T} & \\ & & \mathbf{p}^{[i]T} \end{pmatrix}}_{\mathbf{M}^{[i]}} \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{pmatrix} + \mathbf{t}
\end{aligned}$$

$$\begin{aligned}
\mathbf{H} &= \begin{pmatrix} \sum_i \mathbf{M}^{[i]T}\mathbf{M}^{[i]} & \sum_i {\mathbf{M}^{[i]}}^T \\ \sum_i \mathbf{M}^{[i]} & \sum_i \mathbf{I} \end{pmatrix} \\
\mathbf{b} &= \sum_i \mathbf{M}^{[i]T}(\mathbf{p}^{[i]} - \mathbf{z}^{[i]})
\end{aligned}$$

$$\mathbf{A} = \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{pmatrix} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T$$

# Pruning Correspondences

In absence of a reasonable appearance-based correspondence selection we can exploit some invariants: the distance in the space.

- We can select a triplet of points in the world that are reasonably distant from each other
- We can select a triplet of points in the measurement that have more or less the same distances

**This is not part of RANSAC. It is just a simple optimization that might help us pruning wrong correspondences**

Typical feature-based matching might have inlier ratios: $w > 0.2$

# Selecting Inliers

Given three correspondences, we can "count" the inliers.

- We compute the error of each corresponding point under the solution
- We "count" the number of correspondences that are "good"
- More good points will correspond to a better solution

# How many rounds?

RANSAC is a random procedure, so we are never guaranteed that the solution found is correct.

We can however ask “what is the probability of having a correct solution”

To this extent we need to know

- The inlier ratio $w$ of the correspondences
- The number of points $n$ required to compute the initial solution

# How many rounds?

$w$: inlier ratio

$n$: min number of model points

$p$: desired probability of success

| | | | |
|---|---|---|---|
| prob of fetching n inliers | $w^n$ | | |
| prob of not fetching n inliers | $1 - w^n$ | | |
| prob of not having n inliers in k rounds | $(1 - w^n)^k$ | | |
| prob of failure | $1 - p$ | $=$ | $(1 - w^n)^k$ |
| number of trials required | $k$ | $=$ | $\frac{\log(1-p)}{\log(1-w^n)}$ |