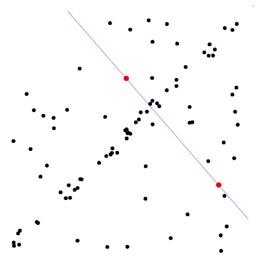
# RANSAC



*Many slides adapted from slides © R. Siegwart, Steve Seitz, J. Tim Oates*

## Where We Are

2

**Last Time**

- Time-of-flight sensors
- Line fitting ("Edge detection")
- Plane fitting
- Split-and-Merge algorithm
- We will **not** cover line regression or Hough-transforms in class
- Testing homework fix

**This Time**

- RANSAC
- Error
- Motors – I hope!

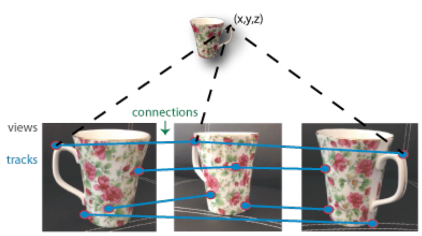
**Overall**

- We're behind, but mostly because we're having good discussions.

## Structure from Motion

3

- Like stereo vision, but moving the camera around
- Find corresponding points and study disparity




*https://www.mathworks.com/help/vision/ug/structure-from-motion.html*

## RANSAC Algorithm

4

- Random Sample Consensus ("Ran-sack")
- General, robust algorithm to fit models in the presence of outliers
- Good tool when goal is to identify points that satisfy a mathematical model or function (like a line)
- Typical applications in robotics
  - Line extraction from 2D range data
  - Plane extraction from 3D range data
  - Structure from motion

## RANSAC Algorithm

5

- RANSAC is an iterative method
- Drawback: A nondeterministic method, results are different between runs.
- Probability to find a line without outliers increases as more iterations are used

## Algorithm 3: RANSAC

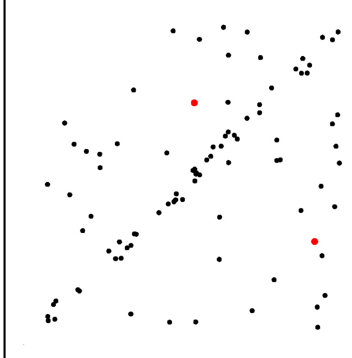
6

**Algorithm 4:** *RANSAC*

1. Initial: let $A$ be a set of $N$ points
2. **repeat**
3. Randomly select a sample of 2 points from $A$
4. Fit a line through the 2 points
5. Compute the distances of all other points to this line
6. Construct the inlier set (i.e. count the number of points with distance to the line $< d$)
7. Store these inliers
8. **until** Maximum number of iterations $k$ reached
9. The set with the maximum number of inliers is chosen as a solution to the problem

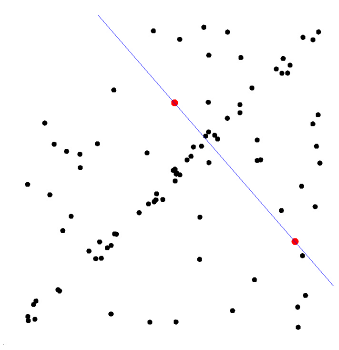
# RANSAC

# RANSAC



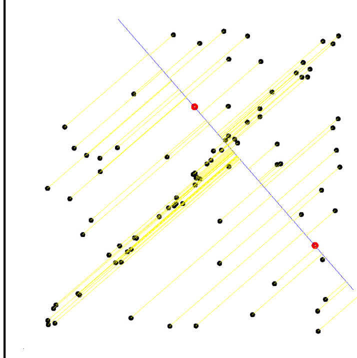
- **Select sample of 2 points at random**

# RANSAC



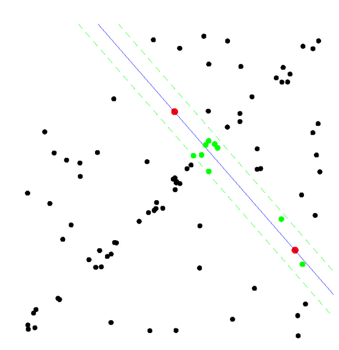
- Select sample of 2 points at random
- **Calculate model parameters that fit the data in the sample**

# RANSAC



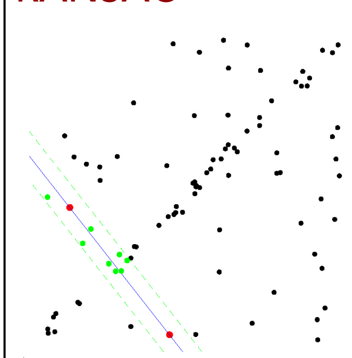
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- **Calculate error function for each data point**

# RANSAC



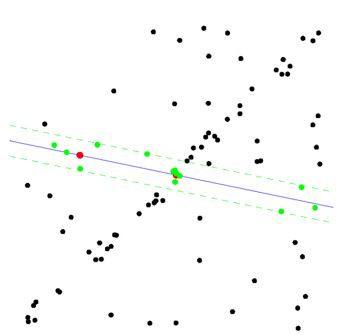
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- **Select data that support current hypothesis**

# RANSAC



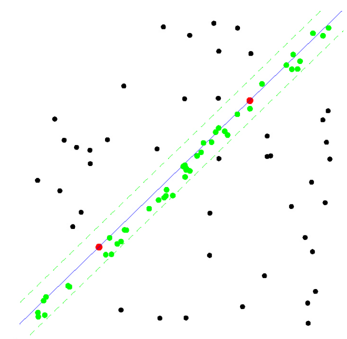
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis
- **Repeat sampling**

## RANSAC



- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis
- **Repeat sampling**

## RANSAC



**ALL-INLIER SAMPLE**

## Algorithm 3: RANSAC

15

**Algorithm 4:** *RANSAC*

1. Initial: let $A$ be a set of $N$ points
2. **repeat**
3. Randomly select a sample of 2 points from $A$
4. Fit $\boxed{\text{a line}}$ through the 2 points
5. Compute the distances of all other points to this line
6. Construct the inlier set (i.e. count the number of points with distance to the line $< d$)
7. Store these inliers
8. **until** Maximum number of iterations $k$ reached
9. The set with the maximum number of inliers is chosen as a solution to the problem

## How Many Iterations?

16

- How many iterations does RANSAC need?
- Can't know in advance if observed set contains maximum number of inliers
- Ideal: check all possible combinations of 2 points
- N(N-1)/2 (for a line) – infeasible if N is too large
- Do not need to check **all** combinations – just a subset if we have a rough estimate of the percentage of inliers in our dataset
- This can be done in a probabilistic way

## RANSAC Iterations

17

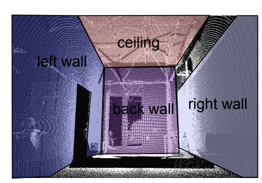
- Let w be fraction of inliners: w = number of inliers / N
  - N is the total number of points.
  - w represents also the probability of selecting an inlier
- p = probability of finding a set of points free of outliers
- w2: probability that both points are inliers
- 1-w2: is the probability that at least one of these two points is an outlier

$$\boxed{k = \frac{\log(1-p)}{\log(1-w^2)}}$$

## Extraction of Planar Features

35

- Goal: extract planar features from a dense point cloud



## RANSAC for Planes

37

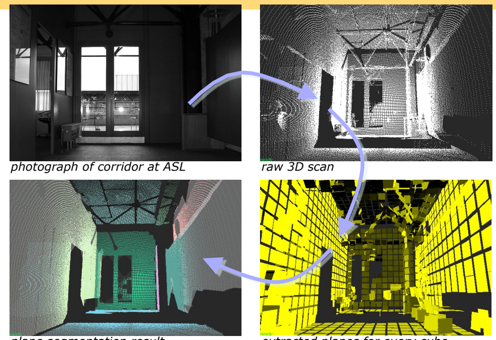
- For every cell, use Ransac to segment the best plane

*pseudo code*

```
threshold = 0.05; // predefined threshold in [m]
m = 100; // predefined the number of ransac iterations

for every cell ci do

        npmax = 0; // initialize maximum number of found closest points

        for m ransac-iterations do

                randomly choose 3 different points p1,p2,p3 from points
                contained in cell ci;

                calculate plane p, i.e. normal n and orth. distance to
                origin d defined by p1,p2,p3;

                np = 0;

                for all points pj in ci do
                        calculate orthogonal distance s from pj to p;
                        if (s < threshold)
                                np++;
                        endif
                endfor

                // maximize the number of points close to the plane
                if (np > npmax)
                        npmax = np; // the number of points close to the plane
                        pbest = p; // this is the found best plane
                endif

        endfor

endfor
```

## Example Result

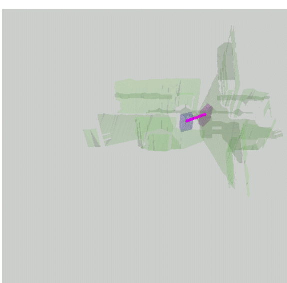
38



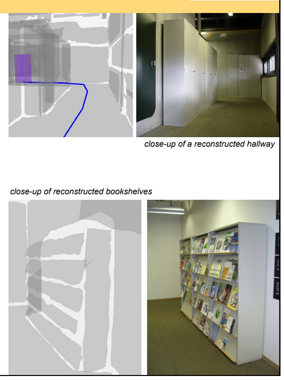

4d - Perception - Features

## Probabilistic Feature Based 3D SLAM

39

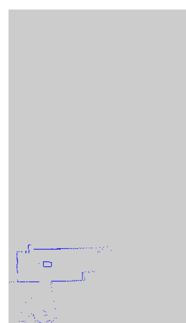


The same experiment as before but this time planar segments are visualized and integrated into the estimation process




## Experimental Results

40



estimation not using planar features

estimation using planar features depicted by red coordinate frames