Sensors and Sensing
Sensor Data Processing

Todor Stoyanov

Centre for Applied Autonomous Sensor Systems
Örebro University
Sweden

Outline

1. Visual Odometry
2. Scan Matching and Registration
3. Point Cloud Processing

Outline

1 Visual Odometry

2 Scan Matching and Registration

3 Point Cloud Processing

Visual Odometry

- Odometry is inherently noisy and suffers from drift.
- How can we correct it using on-board sensors?

Visual Odometry

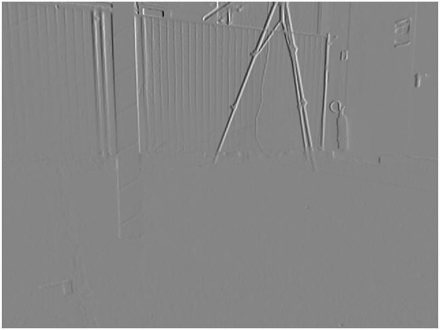
- Odometry is inherently noisy and suffers from drift.
- How can we correct it using on-board sensors?

Images and Image Gradients

- Visual odometry uses camera observations and estimates the pose offset between subsequent views.
- A key concept for many of the approaches is the image gradient $\nabla I = [G_x, G_y]$
- Obtained by convolving with a filter kernel: in this case the sobel operator:

Images and Image Gradients

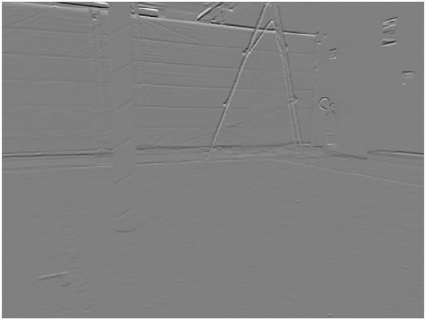
- Visual odometry uses camera observations and estimates the pose offset between subsequent views.
- A key concept for many of the approaches is the image gradient $\nabla I = [G_x, G_y]$
- Obtained by convolving with a filter kernel: in this case the sobel operator:



$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix};$$

Images and Image Gradients

- Visual odometry uses camera observations and estimates the pose offset between subsequent views.
- A key concept for many of the approaches is the image gradient $\nabla I = [G_x, G_y]$
- Obtained by convolving with a filter kernel: in this case the sobel operator:



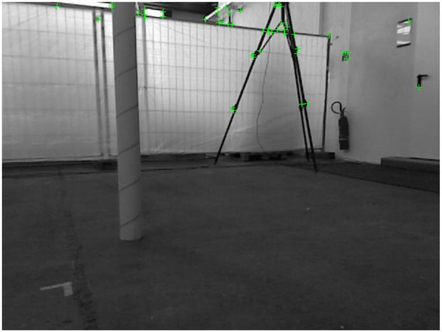
$$S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix};$$

Features

- The Sobel operators are useful for detecting edges in an image.
- Places with high gradient in several directions are an indication of a corner.
- If we can detect corners reliably in several images, we can use them as landmarks.
- Points in the image which can be reliably re-detected and compared are called features or keypoints.

Harris Corner Detector

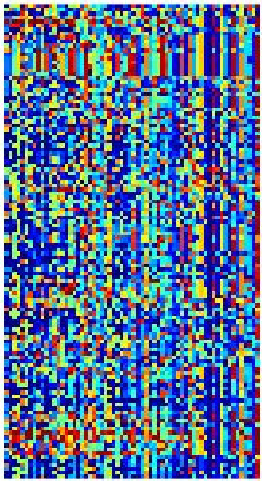
- The Harris corner detector is one of the classical approaches for choosing good features
- Based on observing the shape of derivatives while sliding a window through an image.
- Without going into details, we are interested in the smallest eigenvalue λ_{min} of the matrix:



$$M = \sum_{x,y \in W} \begin{bmatrix} G_x^2 & G_x G_y \\ G_x G_y & G_y^2 \end{bmatrix}$$

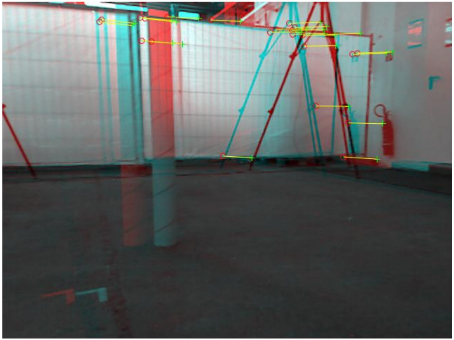
Feature Descriptors

- Once we have detected keypoints, we need to uniquely describe each point.
- Idea: represent 2D neighborhood into a 1D feature vector.
- What should go into the feature vector?
- Many descriptors possible (SIFT, SURF, ORB)



Feature matching

- Given two sets of images, we look for corresponding features.
- We can compare the Euclidean distance between descriptor vectors.
- → often with an adaptive threshold.
- Forward and reverse matches.
- We look for a consistent set of matches, often with RANSAC.



Outline

1. Visual Odometry
2. Scan Matching and Registration
3. Point Cloud Processing

Registration and Map Building — Overview

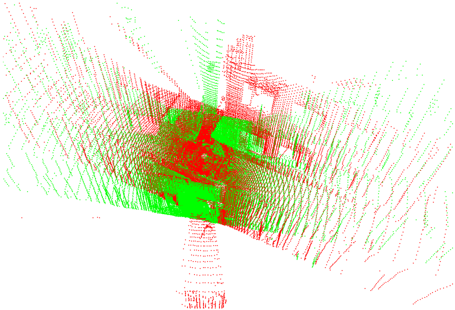
- Most navigation approaches require a map of the environment
- Mapping integrates multiple sensor views in a consistent model
- Registration is a sub-problem in mapping:

- Given two sets of points $\mathcal{P}_1$ and $\mathcal{P}_2$
- Find the transformation $\mathrm{T} = (\mathrm{R}, \mathrm{t})$, which brings $\mathcal{P}_2$ in alignment with $\mathcal{P}_1$

Registration and Map Building — Overview

- Most navigation approaches require a map of the environment
- Mapping integrates multiple sensor views in a consistent model
- Registration is a sub-problem in mapping:

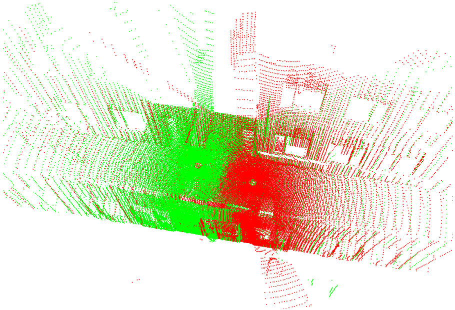
- Given two sets of points $\mathcal{P}_1$ and $\mathcal{P}_2$
- Find the transformation $\mathrm{T} = (\mathrm{R}, \mathrm{t})$, which brings $\mathcal{P}_2$ in alignment with $\mathcal{P}_1$



Registration and Map Building — Overview

- Most navigation approaches require a map of the environment
- Mapping integrates multiple sensor views in a consistent model
- Registration is a sub-problem in mapping:

- Given two sets of points $\mathcal{P}_1$ and $\mathcal{P}_2$
- Find the transformation $\mathrm{T} = (\mathrm{R}, \mathrm{t})$, which brings $\mathcal{P}_2$ in alignment with $\mathcal{P}_1$



Registration and Map Building — Overview

- Most navigation approaches require a map of the environment
- Mapping integrates multiple sensor views in a consistent model
- Registration is a sub-problem in mapping:

- Given two sets of points $\mathcal{P}_1$ and $\mathcal{P}_2$
- Find the transformation $\mathrm{T} = (\mathrm{R}, \mathrm{t})$, which brings $\mathcal{P}_2$ in alignment with $\mathcal{P}_1$

Iterative Closest Point (ICP)

- Classical method for registration is the iterative closest point (ICP) algorithm [1]

Iterative Closest Point (ICP)

- Classical method for registration is the iterative closest point (ICP) algorithm [1]

Algorithm 1: ICP algorithm

```
while not converged do
    For each point in P1 associate closest point in P2;
    Estimate a transformation T = (R, t) from the associated
      points;
    Transform P2 by T;
    Check for convergence;
```

Iterative Closest Point (ICP)

- Classical method for registration is the iterative closest point (ICP) algorithm [1]

Algorithm 2: ICP algorithm

```
while not converged do
    For each point in P_1 associate closest point in P_2;
    Estimate a transformation T = (R, t) from the associated
     points;
    Transform P_2 by T;
    Check for convergence;
```

- Data association for matching points can use different metrics (poin-to-point, point-to-plane distance).
- Important to reject outliers!
- Estimating T can be done using the SVD algorithm

A Simple 1D Example — ICP algorithm

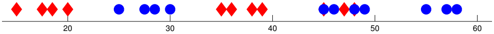


Figure: Iterative Closest Point (ICP)

A Simple 1D Example — ICP algorithm

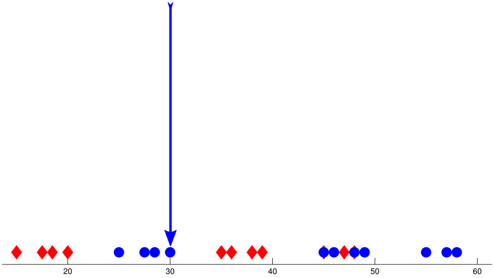


Figure: Iterative Closest Point (ICP)

A Simple 1D Example — ICP algorithm

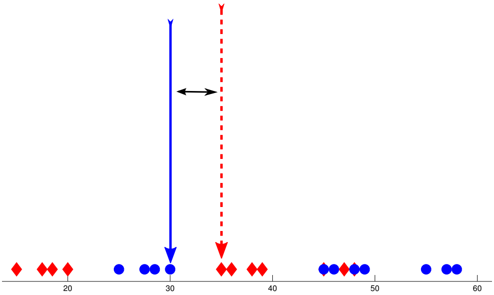


Figure: Iterative Closest Point (ICP)

A Simple 1D Example — ICP algorithm

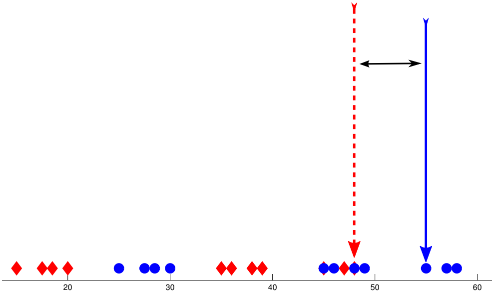


Figure: Iterative Closest Point (ICP)

A Simple 1D Example — ICP algorithm

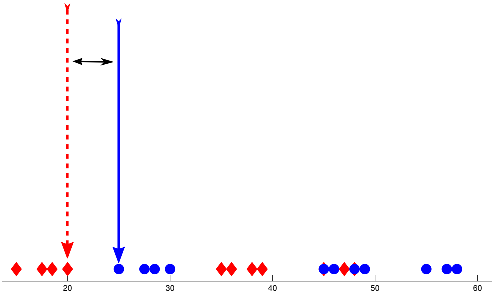


Figure: Iterative Closest Point (ICP)

A Simple 1D Example — ICP algorithm

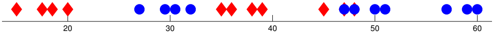


Figure: Iterative Closest Point (ICP)

ICP Limitations and Varriants

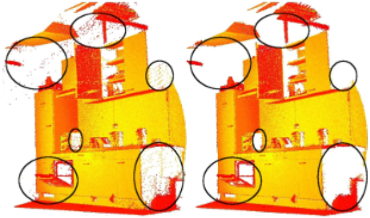
- Sensitive to scan overlap issues
- Local method, requires an initial guess
- Can be slow, as it iterates over all points
- Sped up varriants using efficient spatial search (e.g. Kd-Trees)

Outline

1. Visual Odometry
2. Scan Matching and Registration
3. Point Cloud Processing

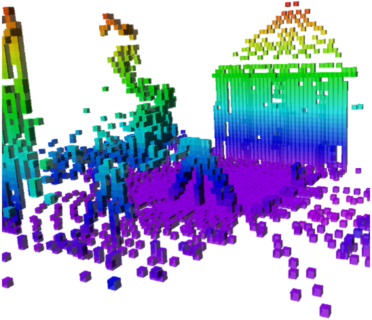
Typical Objectives

- Outlier removal
- Voxelization and spatial indexing
- Extracting high-level structures
- Building hig-level models (occupancy maps, triangle meshes)



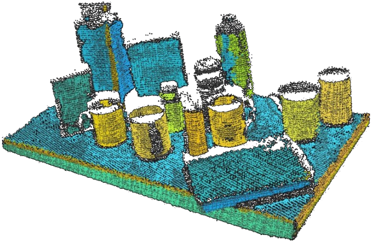
Typical Objectives

- Outlier removal
- Voxelization and spatial indexing
- Extracting high-level structures
- Building hig-level models (occupancy maps, triangle meshes)



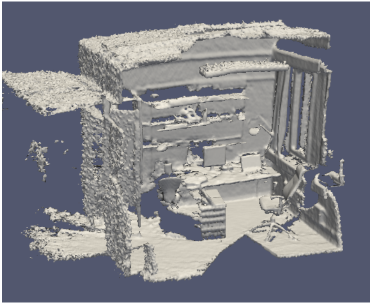
Typical Objectives

- Outlier removal
- Voxelization and spatial indexing
- Extracting high-level structures
- Building hig-level models (occupancy maps, triangle meshes)



Typical Objectives

- Outlier removal
- Voxelization and spatial indexing
- Extracting high-level structures
- Building hig-level models (occupancy maps, triangle meshes)

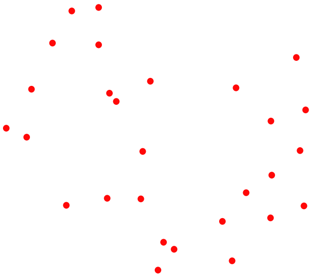


Subsampling and Volumetric Filters

- Often it is necessary to decrease the sample density of 3D points
- e. g., for faster ICP convergence
- Also, uniform density may be desireable to counter scanner position related bias.
- Common approach is to use a voxel grid and only select a single (or a set number) point from each cell.

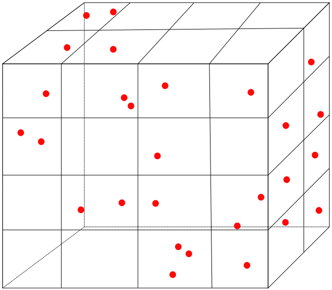
Subsampling and Volumetric Filters

- Often it is necessary to decrease the sample density of 3D points
- e. g., for faster ICP convergence
- Also, uniform density may be desireable to counter scanner position related bias.
- Common approach is to use a voxel grid and only select a single (or a set number) point from each cell.



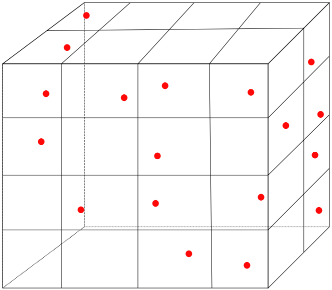
Subsampling and Volumetric Filters

- Often it is necessary to decrease the sample density of 3D points
- e. g., for faster ICP convergence
- Also, uniform density may be desireable to counter scanner position related bias.
- Common approach is to use a voxel grid and only select a single (or a set number) point from each cell.



Subsampling and Volumetric Filters

- Often it is necessary to decrease the sample density of 3D points
- e. g., for faster ICP convergence
- Also, uniform density may be desireable to counter scanner position related bias.
- Common approach is to use a voxel grid and only select a single (or a set number) point from each cell.



Fitting models and Data association

- Data association: How do we choose matching measurements?
- Common problem in several areas of robotics: matching landmarks for localization, points for registration, features for object detection, etc.
- Problem: noisy associations produce outliers that significantly affect solution.
- Problem: least-squares fit is sensitive to outliers.
- Solutions: use a robust norm (L1 instead of L2)
- Solutions: use a randomized algorithm for data association and iterative refinement.

Random Sample Consensus (RANSAC)

- Basic idea:
  - Select a random subset R of the data D.
  - Fit a model M to R. E.g. least squares plane fit. For matching keypoints, we estimate the transform using just the correspondences in R.
  - Check all points in D for fitness wrt M. Points that fulfill a fitness criteria are in the consensus set C.
  - Optionally, refine M with all points from C.
  - Compute a model fitness score based on the number of points in C. If fitness is better than previous models, keep M.
  - Iterate a set number of times or until convergence.

Clustering

- Clustering points together can be useful for several applications
- Clusters contain fewer points and typically fewer outliers, thus can be better suited for model fitting
- Different types of clustering can be formulated based on the distance metric used to decide if a point belongs to a cluster
- Typically, Euclidean distance metric is used
- Machine learning can be useful as a basis (k-means clustering for example), but often adhoc cluster growing approaches

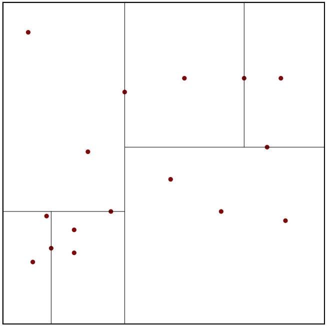
Spatial Indexing

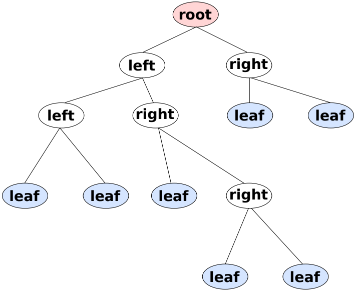
- Various data structures can be used to store point clouds.
- Typical raw data is stored as a 1D array (or 2D for a depth image parametrization).
- Depending on application, different criteria may be important:
  - Spatial complexity (memory storage).
  - Complexity of inserting a new point.
  - Complexity of retrieving a point / nearest neighbour.
- Raw data typically has $O(N)$ search and storage complexity, with $O(1)$ insertion complexity.
- Voxel grids are another often used data structure: storage complexity $O(M^3)$, average time search complexity is $O(\frac{N}{M^3})$, insertion at $O(1)$.

kd Trees and oc Trees

- Maintaining a voxel grid is often prohibitatively expensive (storage) for higher grid resolutions.
- Expected search times depend on the grid resolution as large cells can result in high number of points per cell.
- Tree structures are often used to obtain lower memory requirements at logarithmic search rates

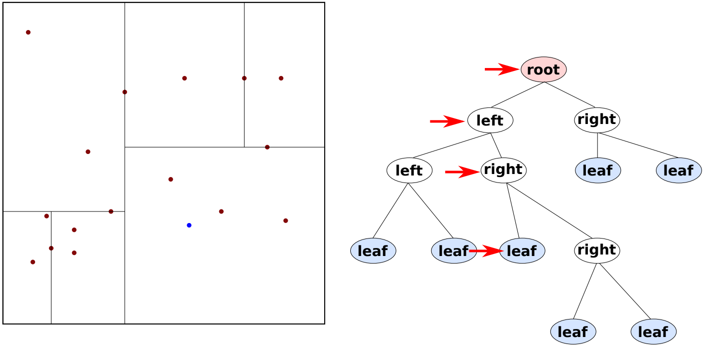
kd Trees and oc Trees





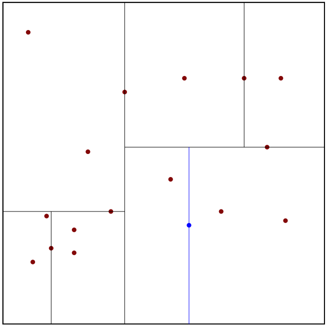
- kd Trees subdivide space and keep a hierarchical indexing structure
- Subdivide at the median point along each dimension.
- Storage: $O(N)$, Search: $O(\log N)/O(N)$, Insert: $O(\log N)/O(N)$

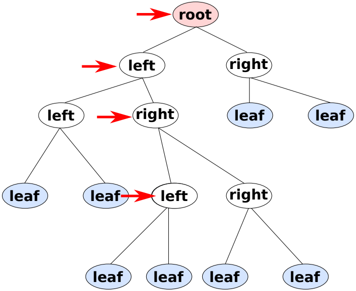
kd Trees and oc Trees



- kd Trees subdivide space and keep a hierarchical indexing structure
- Subdivide at the median point along each dimension.
- Storage: $O(N)$, Search: $O(\log N)/O(N)$, Insert: $O(\log N)/O(N)$

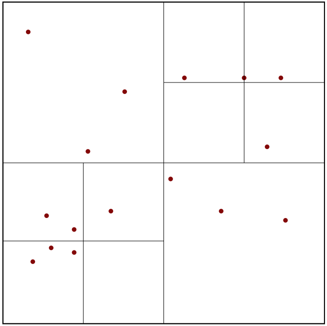
kd Trees and oc Trees

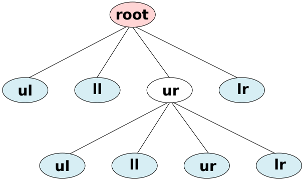




- kd Trees subdivide space and keep a hierarchical indexing structure
- Subdivide at the median point along each dimension.
- Storage: $O(N)$, Search: $O(\log N)/O(N)$, Insert: $O(\log N)/O(N)$

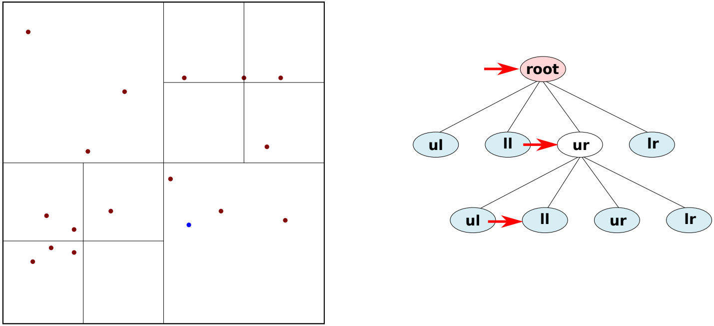
kd Trees and oc Trees





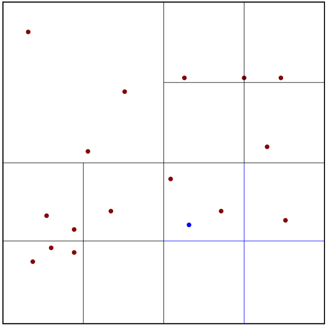
- oc Trees use a regular grid subdisivison for leafs, avoiding complex algorithms for finding the median.
- Similar complexity, worse tree balance.

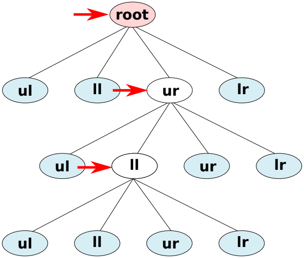
kd Trees and oc Trees



- oc Trees use a regular grid subdisivison for leafs, avoiding complex algorithms for finding the median.
- Similar complexity, worse tree balance.

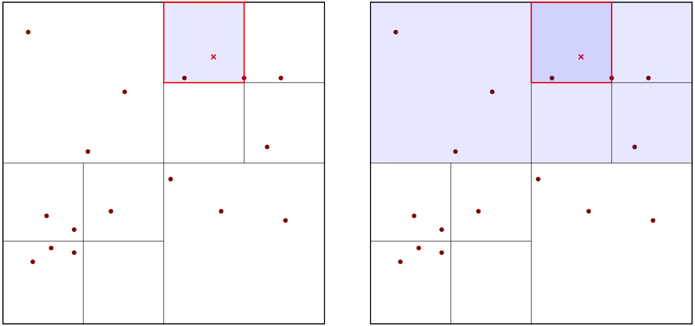
kd Trees and oc Trees





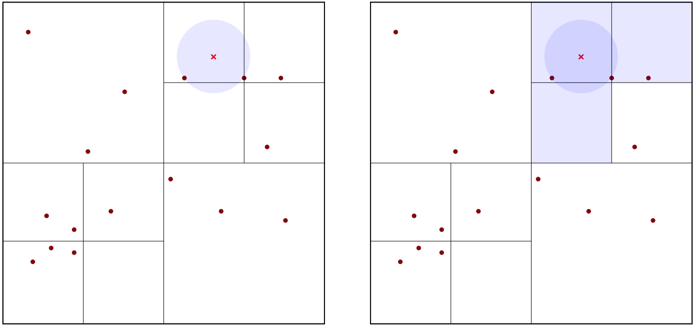
- oc Trees use a regular grid subdisivison for leafs, avoiding complex algorithms for finding the median.
- Similar complexity, worse tree balance.

kd Trees and oc Trees



- Often possible to speed up search using an approximate algorithm and/or looking for neighbors within a given radius.

kd Trees and oc Trees



- Often possible to speed up search using an approximate algorithm and/or looking for neighbors within a given radius.

Sensors and Sensing
Sensor Data Processing

Todor Stoyanov

Centre for Applied Autonomous Sensor Systems
Örebro University
Sweden

References

P.J. Besl and N.D. McKay.
A Method for Registration of 3D Shapes.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 14:239–256, 1992.