Path Planning in 3D Environments using the Normal Distributions Transform

Todor Stoyanov, Martin Magnusson, Henrik Andreasson and Achim J. Lilienthal

Learning Systems Lab
Center for Applied Autonomous Sensor Systems (AASS)
Örebro University, Sweden
todor.stoyanov@oru.se
Outline

1. Motivation
2. Spatial Representation through 3D-NDT
3. Path Planning in 3D-NDT Maps
4. Evaluation
5. Conclusion
Outline

1. Motivation
2. Spatial Representation through 3D-NDT
3. Path Planning in 3D-NDT Maps
4. Evaluation
5. Conclusion
The Autonomous Navigation Task

- Mobile Robots are a well studied research area
- Tasks commonly automated include:
  - Mapping and Localization
  - Path Planning
  - Path Following
- These tasks are commonly referred to as Autonomous Navigation
- Recent advances in sensing technology provide 3D information
  - Custom-built laser solutions
  - Commercial lasers and time of flight cameras
- Sensors produce 3D point clouds
- Already used for 6DOF SLAM
The Autonomous Navigation Task

- Mobile Robots are a well studied research area
- Tasks commonly automated include:
  - Mapping and Localization
  - Path Planning
  - Path Following
- These tasks are commonly referred to as *Autonomous Navigation*

- Recent advances in sensing technology provide 3D information
  - Custom-built laser solutions
  - Commercial lasers and time of flight cameras
- Sensors produce 3D point clouds
- Already used for 6DOF SLAM
The Autonomous Navigation Task

- Mobile Robots are a well studied research area
- Tasks commonly automated include:
  - Mapping and Localization
  - Path Planning
  - Path Following
- These tasks are commonly referred to as *Autonomous Navigation*
- Recent advances in sensing technology provide 3D information
  - Custom-built laser solutions
  - Commercial lasers and time of flight cameras
- Sensors produce 3D point clouds
- Already used for 6DOF SLAM
The Autonomous Navigation Task

- Mobile Robots are a well studied research area
- Tasks commonly automated include:
  - Mapping and Localization
  - Path Planning
  - Path Following
- These tasks are commonly referred to as *Autonomous Navigation*
- Recent advances in sensing technology provide 3D information
  - Custom-built laser solutions
  - Commercial lasers and time of flight cameras
- Sensors produce 3D point clouds
- Already used for 6DOF SLAM
The Autonomous Navigation Task

- Mobile Robots are a well studied research area
- Tasks commonly automated include:
  - Mapping and Localization
  - Path Planning
  - Path Following
- These tasks are commonly referred to as *Autonomous Navigation*
- Recent advances in sensing technology provide 3D information
  - Custom-built laser solutions
  - Commercial lasers and time of flight cameras
- Sensors produce 3D point clouds
- Already used for 6DOF SLAM
Spatial Modeling and Path Planning

- Path planning in 3D environments is a difficult problem
- What spatial representation should be used?

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Cloud</td>
<td>large size, difficult to manipulate</td>
</tr>
<tr>
<td>Triangle Mesh</td>
<td>difficult to construct</td>
</tr>
<tr>
<td>Elevation Grid</td>
<td>low expressive power</td>
</tr>
<tr>
<td>Multilevel Surface Map</td>
<td>no 3D path planners</td>
</tr>
</tbody>
</table>

Motivation: Develop a planner that works on a spatial representation that is directly available, expressive and efficient
Spatial Modeling and Path Planning

- Path planning in 3D environments is a difficult problem
- What spatial representation should be used?

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Cloud</td>
<td>large size, difficult to manipulate</td>
</tr>
<tr>
<td>Triangle Mesh</td>
<td>difficult to construct</td>
</tr>
<tr>
<td>Elevation Grid</td>
<td>low expressive power</td>
</tr>
<tr>
<td>Multilevel Surface Map</td>
<td>no 3D path planners</td>
</tr>
</tbody>
</table>

**Motivation:** Develop a planner that works on a spatial representation that is directly available, expressive and efficient
Path planning in 3D environments is a difficult problem.

What spatial representation should be used?

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Cloud</td>
<td>large size</td>
</tr>
<tr>
<td></td>
<td>difficult to manipulate</td>
</tr>
<tr>
<td>Triangle Mesh</td>
<td>difficult to construct</td>
</tr>
<tr>
<td>Elevation Grid</td>
<td>low expressive power</td>
</tr>
<tr>
<td>Multilevel Surface Map</td>
<td>no 3D path planners</td>
</tr>
</tbody>
</table>

Motivation: Develop a planner that works on a spatial representation that is directly available, expressive and efficient.
Spatial Modeling and Path Planning

- Path planning in 3D environments is a difficult problem
- What spatial representation should be used?

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Cloud</td>
<td>large size, difficult to manipulate</td>
</tr>
<tr>
<td>Triangle Mesh</td>
<td>difficult to construct</td>
</tr>
<tr>
<td>Elevation Grid</td>
<td>low expressive power</td>
</tr>
<tr>
<td>Multilevel</td>
<td></td>
</tr>
<tr>
<td>Surface Map</td>
<td>no 3D path planners</td>
</tr>
</tbody>
</table>

Motivation: Develop a planner that works on a spatial representation that is directly available, expressive and efficient.
Spatial Modeling and Path Planning

- Path planning in 3D environments is a difficult problem
- What spatial representation should be used?

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Cloud</td>
<td>large size</td>
</tr>
<tr>
<td></td>
<td>difficult to manipulate</td>
</tr>
<tr>
<td>Triangle Mesh</td>
<td>difficult to construct</td>
</tr>
<tr>
<td>Elevation Grid</td>
<td>low expressive power</td>
</tr>
<tr>
<td>Multilevel Surface Map</td>
<td>no 3D path planners</td>
</tr>
</tbody>
</table>

Motivation: Develop a planner that works on a spatial representation that is directly available, expressive and efficient
Path planning in 3D environments is a difficult problem

What spatial representation should be used?

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Cloud</td>
<td>large size, difficult to manipulate</td>
</tr>
<tr>
<td>Triangle Mesh</td>
<td>difficult to construct</td>
</tr>
<tr>
<td>Elevation Grid</td>
<td>low expressive power</td>
</tr>
<tr>
<td>Multilevel Surface Map</td>
<td>no 3D path planners</td>
</tr>
</tbody>
</table>

Motivation: Develop a planner that works on a spatial representation that is directly available, expressive and efficient
Spatial Modeling and Path Planning

- Path planning in 3D environments is a difficult problem
- What spatial representation should be used?

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point Cloud</td>
<td>large size, difficult to manipulate</td>
</tr>
<tr>
<td>Triangle Mesh</td>
<td>difficult to construct</td>
</tr>
<tr>
<td>Elevation Grid</td>
<td>low expressive power</td>
</tr>
<tr>
<td>Multilevel Surface Map</td>
<td>no 3D path planners</td>
</tr>
</tbody>
</table>

**Motivation:** Develop a planner that works on a spatial representation that is directly available, expressive and efficient
Outline

1. Motivation

2. Spatial Representation through 3D-NDT

3. Path Planning in 3D-NDT Maps

4. Evaluation

5. Conclusion
The Normal Distributions Transform

- The Normal Distributions Transform originally developed for 2D scan registration
- Points grouped into cells
- A Gaussian pdf used to represent space in the cell
- Extension to 3D is expressive and space efficient
- Each cell represented by Covariance matrix $C$ and mean $\mu$
- $C$ and $\mu$ are the only variables available to the planner
The 3D Normal Distributions Transform - Applications

3D-NDT has already been used in a variety of applications:

- 3D scan registration
- place recognition
- change detection
The 3D Normal Distributions Transform - Applications

3D-NDT has already been used in a variety of applications:

- 3D scan registration
- Place recognition
- Change detection
The 3D Normal Distributions Transform - Applications

3D-NDT has already been used in a variety of applications:

- 3D scan registration
- Place recognition
- Change detection
3D-NDT has already been used in a variety of applications:

- 3D scan registration
- place recognition
- change detection
3D-NDT Example

- Point clouds can be directly represented as 3D-NDTs and stacked together, using a SLAM global optimization framework.
Outline

1 Motivation

2 Spatial Representation through 3D-NDT

3 Path Planning in 3D-NDT Maps

4 Evaluation

5 Conclusion
Extending Wavefront to 3D-NDT

- Wavefront is a well known planner in 2D
- Grid approach → difficult to scale in 3D
- Extension to 3D-NDT maps:

Initialize active cell list $Q$
\[ \forall q \in C : q.cost = \infty \]
\[ q_{goal}.cost = 0 \]
\[ Q \leftarrow q_{goal} \]

while $Q$ not empty do
  \[ q_c \leftarrow Q.pop() \]
  if $\text{CollisionCheck}(q_c)$ then
    \[ Q_n \leftarrow \text{Neighbors}(q_c) \]
    if $Q_n.cost > q_{curr}.cost$ then
      \[ Q_n.cost = q_c.cost + \text{cost}(Q_n,q_c) \]
      \[ Q.push(Q_n) \]

Follow gradient
Extending Wavefront to 3D-NDT

Wavefront is a well known planner in 2D
- Grid approach → difficult to scale in 3D
- Extension to 3D-NDT maps:

Initialize active cell list \( Q \)
\[
\forall q \in \mathcal{C} : q.\text{cost} = \infty
\]
\[
q_{\text{goal}}.\text{cost} = 0
\]
\[
Q \leftarrow q_{\text{goal}}
\]

while \( Q \) not empty do
  \( q_c \leftarrow Q.\text{pop}() \)
  if \( \text{CollisionCheck}(q_c) \) then
    \( Q_n \leftarrow \text{Neighbors}(q_c) \)
    if \( Q_n.\text{cost} > q_{\text{curr}}.\text{cost} \) then
      \( Q_n.\text{cost} = q_c.\text{cost} + \text{cost}(Q_n, q_c) \)
      \( Q.\text{push}(Q_n) \)

Follow gradient
Extending Wavefront to 3D-NDT

- Wavefront is a well known planner in 2D
- Grid approach → difficult to scale in 3D
- Extension to 3D-NDT maps:

Initialize active cell list $Q$

\[ \forall q \in C : q.cost = \infty \]

$q_{goal}.cost = 0$

$Q \leftarrow q_{goal}$

while $Q$ not empty do

\[ q_c \leftarrow Q.pop() \]

if $\text{CollisionCheck}(q_c)$ then

\[ Q_n \leftarrow \text{Neighbors}(q_c) \]

if $Q_n.cost > q_{curr}.cost$ then

\[ Q_n.cost = q_c.cost + \text{cost}(Q_n, q_c) \]

$Q.push(Q_n)$

Follow gradient
Extending Wavefront to 3D-NDT

- Wavefront is a well known planner in 2D
- Grid approach → difficult to scale in 3D
- Extension to 3D-NDT maps:

Initialize active cell list $Q$

\[ \forall q \in C : q.cost = \infty \]

$q_{\text{goal}}.cost = 0$

$Q \leftarrow q_{\text{goal}}$

**while** $Q$ not empty **do**

\[ q_c \leftarrow Q.pop() \]

**if** CollisionCheck$(q_c)$ **then**

\[ Q_n \leftarrow \text{Neighbors}(q_c) \]

**if** $Q_n.cost > q_{\text{curr}}.cost$ **then**

\[ Q_n.cost = q_c.cost + \text{cost}(Q_n, q_c) \]

$Q.push(Q_n)$

Follow gradient
Extending Wavefront to 3D-NDT

- Wavefront is a well known planner in 2D
- Grid approach → difficult to scale in 3D
- Extension to 3D-NDT maps:

Initialize active cell list $Q$

$\forall q \in C : q.cost = \infty$

$q_{goal}.cost = 0$

$Q \leftarrow q_{goal}$

while $Q$ not empty do

$q_c \leftarrow Q.pop()$

if CollisionCheck($q_c$) then

$Q_n \leftarrow Neighbors(q_c)$

if $Q_n.cost > q_{curr}.cost$ then

$Q_n.cost = q_c.cost + cost(Q_n, q_c)$

$Q.push(Q_n)$

Follow gradient
Extending Wavefront to 3D-NDT

- Wavefront is a well known planner in 2D
- Grid approach → difficult to scale in 3D
- Extension to 3D-NDT maps:

Initialize active cell list $Q$

$\forall q \in C : q.cost = \infty$

$q_{goal}.cost = 0$

$Q \leftarrow q_{goal}$

while $Q$ not empty do

$q_c \leftarrow Q.pop()$

if CollisionCheck($q_c$) then

$Q_n \leftarrow Neighbors(q_c)$

if $Q_n.cost > q_{curr}.cost$ then

$Q_n.cost = q_c.cost + cost(Q_n, q_c)$

$Q.push(Q_n)$

Follow gradient
Extending Wavefront to 3D-NDT

- Wavefront is a well known planner in 2D
- Grid approach → difficult to scale in 3D
- Extension to 3D-NDT maps:

Initialize active cell list \( Q \)
\[
\forall q \in C : q.cost = \infty
\]
\[q_{goal}.cost = 0\]
\[Q \leftarrow q_{goal}\]

\textbf{while} \( Q \) not empty \textbf{do}

\[q_c \leftarrow Q.pop()\]

\textbf{if} \( \text{CollisionCheck}(q_c) \) \textbf{then}

\[Q_n \leftarrow \text{Neighbors}(q_c)\]

\textbf{if} \( Q_n.cost > q_{curr}.cost \) \textbf{then}

\[Q_n.cost = q_c.cost + \text{cost}(Q_n, q_c)\]

\[Q.push(Q_n)\]

Follow gradient
Collision Checking

- Geometrically, the Gaussian \textbf{pdf} in each cell interpreted as an ellipsoid.
- Depending on eigenvalues of $\mathbf{C}$ can be approximately spherical, linear or planar.
- The shape and orientation of each ellipsoid used to label cells.
Collision Checking

**Input:** $q \leftarrow$ current cell  
$N \leftarrow$ neighbor cells  

for $\forall n \in N$ do  
  if $\text{Angle}(n, q) < \text{maxPitch}$ then  
    $\text{PossibleSupport} \leftarrow n$  
  else  
    $\text{PossibleCollision} \leftarrow n$  
  if $\exists n_i \in \text{PossibleSupport} : n_i$ not horizontal or inclined then  
    return Collision  
  if $\exists n_i \in \text{PossibleCollision} : \text{Collides}(q, n_i)$ then  
    return Collision  
return No Collision
Collision Checking

**Input:** $q \leftarrow$ current cell

$N \leftarrow$ neighbor cells

for $\forall n \in N$ do

if $\text{Angle}(n, q) < \text{maxPitch}$ then

$\text{PossibleSupport} \leftarrow n$

else

$\text{PossibleCollision} \leftarrow n$

if $\exists n_i \in \text{PossibleSupport} : n_i$ not horizontal or inclined then

return Collision

if $\exists n_i \in \text{PossibleCollision} : \text{Collides}(q, n_i)$ then

return Collision

return No Collision
Collision Checking

Input: $q \leftarrow$ current cell
$N \leftarrow$ neighbor cells

for $\forall n \in N$ do
  if $\text{Angle}(n, q) < \text{maxPitch}$ then
    $\text{PossibleSupport} \leftarrow n$
  else
    $\text{PossibleCollision} \leftarrow n$
  if $\exists n_i \in \text{PossibleSupport}$ : $n_i$ not horizontal or inclined then
    return Collision
  if $\exists n_i \in \text{PossibleCollision}$ : $\text{Collides}(q, n_i)$ then
    return Collision
return No Collision
Collision Checking

**Input:** \( q \leftarrow \) current cell  
\( N \leftarrow \) neighbor cells  

for \( \forall n \in N \) do  
  if \( \text{Angle}(n,q) < \text{maxPitch} \) then  
    \( \text{PossibleSupport} \leftarrow n \)  
  else  
    \( \text{PossibleCollision} \leftarrow n \)  
  if \( \exists n_i \in \text{PossibleSupport} : n_i \) not horizontal or inclined then  
    return Collision  
  if \( \exists n_i \in \text{PossibleCollision} : \text{Collides}(q,n_i) \) then  
    return Collision  
return No Collision
Collision Checking

**Input:** $q \leftarrow$ current cell  
$N \leftarrow$ neighbor cells  

for $\forall n \in N$ do  
  if $\text{Angle}(n, q) < \text{maxPitch}$  
  then  
    $\text{PossibleSupport} \leftarrow n$  
  else  
    $\text{PossibleCollision} \leftarrow n$  

if $\exists n_i \in \text{PossibleSupport} : n_i$ not horizontal or inclined then  
  return Collision  

if $\exists n_i \in \text{PossibleCollision} : \text{Collides}(q, n_i)$ then  
  return Collision  
return No Collision
Collision Checking

**Input:** $q \leftarrow$ current cell  
$N \leftarrow$ neighbor cells  
for $\forall n \in N$ do  
    if $\text{Angle}(n, q) < \text{maxPitch}$  
        then  
            $\text{PossibleSupport} \leftarrow n$  
        else  
            $\text{PossibleCollision} \leftarrow n$  
    if $\exists n_i \in \text{PossibleSupport} : n_i$ not horizontal or inclined then  
        return Collision  
    if $\exists n_i \in \text{PossibleCollision} : \text{Collides}(q, n_i)$ then  
        return Collision  
return No Collision
Collision Checking

**Input:** $q \leftarrow$ current cell  
$N \leftarrow$ neighbor cells

**for** $\forall n \in N$ **do**

- **if** $\text{Angle}(n, q) < \text{maxPitch}$ **then**
  - PossibleSupport $\leftarrow n$
- **else**
  - PossibleCollision $\leftarrow n$

**if** $\exists n_i \in \text{PossibleSupport}: n_i$ not horizontal or inclined **then**
- **return** Collision

**if** $\exists n_i \in \text{PossibleCollision}: \text{Collides}(q, n_i)$ **then**
- **return** Collision

**return** No Collision
Collision Checking

**Input:**
- $q \leftarrow$ current cell
- $N \leftarrow$ neighbor cells

**Algorithm:**

```plaintext
for $\forall n \in N$ do
    if $\text{Angle}(n, q) < \text{maxPitch}$
    then
        $\text{PossibleSupport} \leftarrow n$
    else
        $\text{PossibleCollision} \leftarrow n$
    
if $\exists n_i \in \text{PossibleSupport} : n_i \text{ not horizontal or inclined}$ then
    return Collision

if $\exists n_i \in \text{PossibleCollision} : \text{Collides}(q, n_i)$ then
    return Collision

return No Collision
```
Finding Accessible Cells

- Next step requires to propagate the wavefront to accessible cells
- The support cells from previous step are used
- All cells are guaranteed to be flat and drivable
- Choose only immediate neighbors
Finding Accessible Cells

- Next step requires to propagate the wavefront to accessible cells.
- The support cells from previous step are used.
- All cells are guaranteed to be flat and drivable.
- Choose only immediate neighbors.
Finding Accessible Cells

- Next step requires to propagate the wavefront to accessible cells
- The support cells from previous step are used
- All cells are guaranteed to be flat and drivable
- Choose only immediate neighbors
Outline

1. Motivation
2. Spatial Representation through 3D-NDT
3. Path Planning in 3D-NDT Maps
4. Evaluation
5. Conclusion
Test Environments

- The performance of the path planner was evaluated empirically
- Four diverse data sets were used
  - Hallway Environment
  - Mining Tunnels (straight)
  - Mining Tunnels (curving)
  - Outdoors asphalt processing site
- Configurations deemed reachable were given as start/goal locations
- The produced maps show:
  - cells reachable from the goal
  - path between goal and start
  - cost to reach each cell
Evaluating Path Feasibility

size = 23x16 meters
Evaluating Path Feasibility
Evaluating Path Feasibility
Evaluating Path Feasibility

Size = 59x45 meters
Evaluating Path Feasibility

Following a Path
Evaluation

Runtime Statistics

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Points</th>
<th>NDT Cells</th>
<th>Reachable Cells</th>
<th>NDT Build Time</th>
<th>Planning Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hallway</td>
<td>636k</td>
<td>4 423</td>
<td>282</td>
<td>12.5s</td>
<td>0.2s</td>
</tr>
<tr>
<td>Mine</td>
<td>863k</td>
<td>21 460</td>
<td>1 635</td>
<td>16.6s</td>
<td>1.8s</td>
</tr>
<tr>
<td>Kvantorp</td>
<td>1 528k</td>
<td>31 269</td>
<td>4 120</td>
<td>26.5s</td>
<td>3.1s</td>
</tr>
<tr>
<td>Piles</td>
<td>183k</td>
<td>1 361</td>
<td>782</td>
<td>3.3s</td>
<td>0.6s</td>
</tr>
</tbody>
</table>

Table: Environment complexity and runtime statistics

- Complexity of the algorithm depends strongly on number of reachable cells $K$
- Proportionate to $\Theta(K\log(N))$
- Note: Cost is dominated by building the NDT Map
Outline

1. Motivation
2. Spatial Representation through 3D-NDT
3. Path Planning in 3D-NDT Maps
4. Evaluation
5. Conclusion
Contributions

- Proposed a path planning algorithm that works in 3D-NDT Maps:

  Demonstrates the utility of 3D-NDT as a spatial representation for path planning

  Uses data readily available from the underlying SLAM module

  Plans directly in the 3D data structure, avoiding projection to 2D

  Lays the foundation for future work on 3D-NDT based planners
Contributions

- Proposed a path planning algorithm that works in 3D-NDT Maps:
  
  - Demonstrates the utility of 3D-NDT as a spatial representation for path planning
  - Uses data readily available from the underlying SLAM module
  - Plans directly in the 3D data structure, avoiding projection to 2D
  - Lays the foundation for future work on 3D-NDT based planners
Contributions

- Proposed a path planning algorithm that works in 3D-NDT Maps:
  
  - Demonstrates the utility of 3D-NDT as a spatial representation for path planning
  - Uses data readily available from the underlying SLAM module
  - Plans directly in the 3D data structure, avoiding projection to 2D
  
  Lays the foundation for future work on 3D-NDT based planners
Contributions

- Proposed a path planning algorithm that works in 3D-NDT Maps:

  - Demonstrates the utility of 3D-NDT as a spatial representation for path planning
  - Uses data readily available from the underlying SLAM module
  - Plans directly in the 3D data structure, avoiding projection to 2D
  - Lays the foundation for future work on 3D-NDT based planners
Future Work

- Comparison with similar approaches like MLS can demonstrate even better the benefits of the full 3D representation
- Many path planning primitives already implemented:
  - Collision Check
  - Stability Evaluation
  - State Transition Evaluation
- a logical next step is to implement an RRT planner working on 3D-NDTs
- A benefit from this direction is the incorporation of *dynamic* constraints
Thank You!

Thank you for your attention.
Questions?