Perception, Part 2
Range-Based Feature Extraction

Computer Science 4766/6912

Department of Computer Science
Memorial University of Newfoundland

June 8, 2018

1. Feature Extraction
   - Line Extraction
   - Other Range-Based Features

Feature Extraction

- Sometimes sensor values are interpreted directly (e.g. front sonar sensor tied to emergency stop command)

Feature Extraction

- Sometimes sensor values are interpreted directly (e.g. front sonar sensor tied to emergency stop command)
- However, we often require higher-level information

Feature Extraction

- Sometimes sensor values are interpreted directly (e.g. front sonar sensor tied to emergency stop command)
- However, we often require higher-level information
- A **feature** abstracts away the particular sensor values to yield a description of some characteristic of the environment (or its appearance)

Feature Extraction

- Sometimes sensor values are interpreted directly (e.g. front sonar sensor tied to emergency stop command)
- However, we often require higher-level information
- A **feature** abstracts away the particular sensor values to yield a description of some characteristic of the environment (or its appearance)
    - Low-level: lines and other geometric features

Feature Extraction

- Sometimes sensor values are interpreted directly (e.g. front sonar sensor tied to emergency stop command)
- However, we often require higher-level information
- A **feature** abstracts away the particular sensor values to yield a description of some characteristic of the environment (or its appearance)
    - Low-level: lines and other geometric features
    - High-level: objects (e.g. doors, cars)

Feature Extraction

- Sometimes sensor values are interpreted directly (e.g. front sonar sensor tied to emergency stop command)
- However, we often require higher-level information
- A **feature** abstracts away the particular sensor values to yield a description of some characteristic of the environment (or its appearance)
  - Low-level: lines and other geometric features
  - High-level: objects (e.g. doors, cars)
- The process of **feature extraction** extracts a feature's description (i.e. its parameters) from the raw sensor data

Feature Extraction

- Sometimes sensor values are interpreted directly (e.g. front sonar sensor tied to emergency stop command)
- However, we often require higher-level information
- A **feature** abstracts away the particular sensor values to yield a description of some characteristic of the environment (or its appearance)
  - Low-level: lines and other geometric features
  - High-level: objects (e.g. doors, cars)
- The process of **feature extraction** extracts a feature's description (i.e. its parameters) from the raw sensor data
- Features are very useful for localization and mapping

Line Extraction

- Assume we have a sensor (e.g. laser rangefinder) that returns a sequence of (range, angle) points

Line Extraction

- Assume we have a sensor (e.g. laser rangefinder) that returns a sequence of (range, angle) points
- For the moment, we assume that all of these points come from a single line in the environment (e.g. a wall)

Line Extraction

- Assume we have a sensor (e.g. laser rangefinder) that returns a sequence of (range, angle) points
- For the moment, we assume that all of these points come from a single line in the environment (e.g. a wall)
- If we have more than two points then it is unlikely that we can find a single line which passes through all of them

Line Extraction

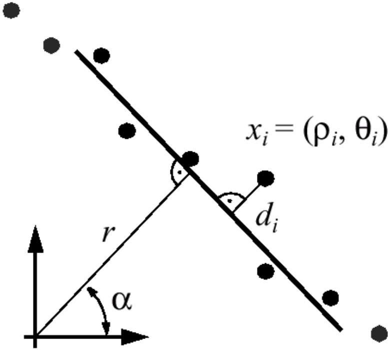
- Assume we have a sensor (e.g. laser rangefinder) that returns a sequence of (range, angle) points
- For the moment, we assume that all of these points come from a single line in the environment (e.g. a wall)
- If we have more than two points then it is unlikely that we can find a single line which passes through all of them
  - The system is *overdetermined* (too many equations)

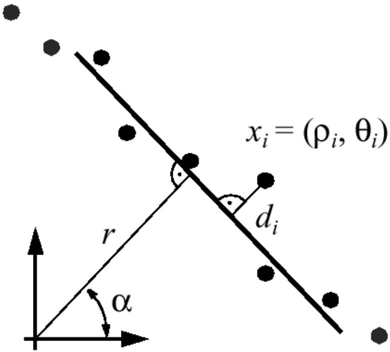
Line Extraction

- Assume we have a sensor (e.g. laser rangefinder) that returns a sequence of (range, angle) points
- For the moment, we assume that all of these points come from a single line in the environment (e.g. a wall)
- If we have more than two points then it is unlikely that we can find a single line which passes through all of them
    - The system is *overdetermined* (too many equations)
    - We apply *least squares optimization* to find a line of best fit

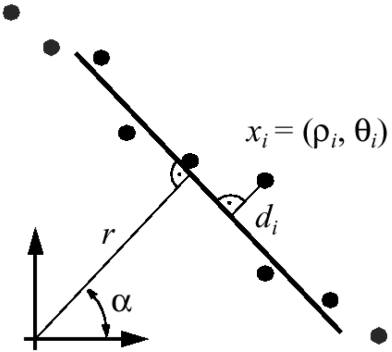
Line Extraction

- Assume we have a sensor (e.g. laser rangefinder) that returns a sequence of (range, angle) points
- For the moment, we assume that all of these points come from a single line in the environment (e.g. a wall)
- If we have more than two points then it is unlikely that we can find a single line which passes through all of them
  - The system is *overdetermined* (too many equations)
  - We apply *least squares optimization* to find a line of best fit
  - In fact, we apply *weighted least squares* using the uncertainty of data points to weight their impact on the solution

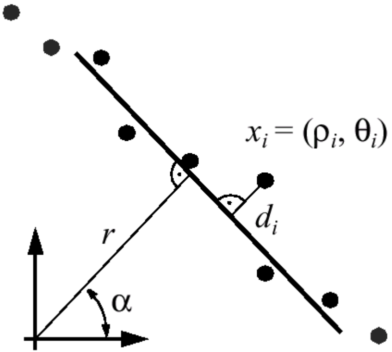

$x_i = (\rho_i, \theta_i)$
d_i
r
α



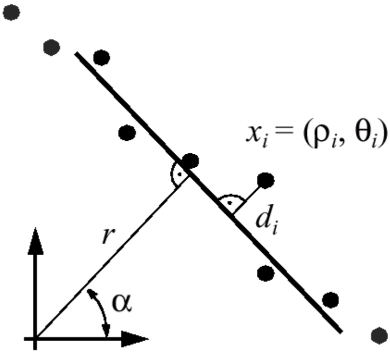
- The line is described by the perpendicular distance from the origin, r, and the angle of the perpendicular, α



- The line is described by the perpendicular distance from the origin, r, and the angle of the perpendicular, α
- We have n data points, expressed in polar coordinates: $x_i = (\rho_i, \theta_i)$



- The line is described by the perpendicular distance from the origin, r, and the angle of the perpendicular, α
- We have n data points, expressed in polar coordinates: $x_i = (\rho_i, \theta_i)$

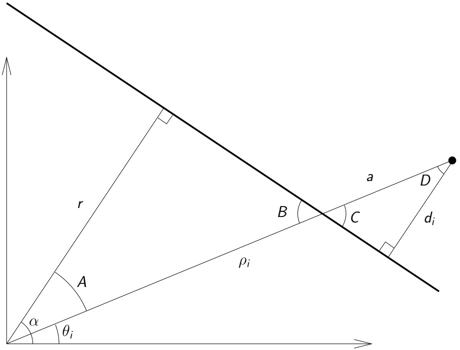


- The line is described by the perpendicular distance from the origin, r, and the angle of the perpendicular, α
- We have n data points, expressed in polar coordinates: $x_i = (\rho_i, \theta_i)$

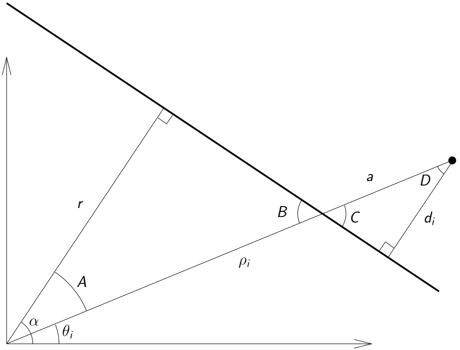
We need to formulate the orthogonal distance d_i from a data point (ρ_i, θ_i) to a line with parameters r and α...

Compute d_i given data point (ρ_i, θ_i) and line parameters r and α:

Compute d_i given data point (ρ_i, θ_i) and line parameters r and α:

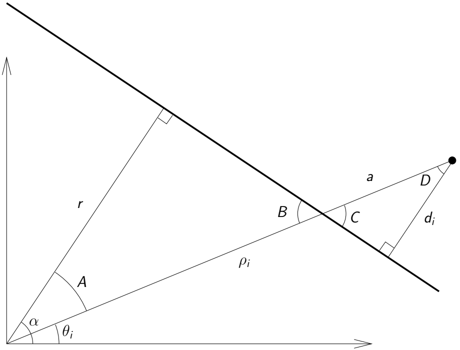


Compute d_i given data point (ρ_i, θ_i) and line parameters r and α:



From similar triangles we obtain $a = \frac{\rho_i d_i}{r + d_i}$

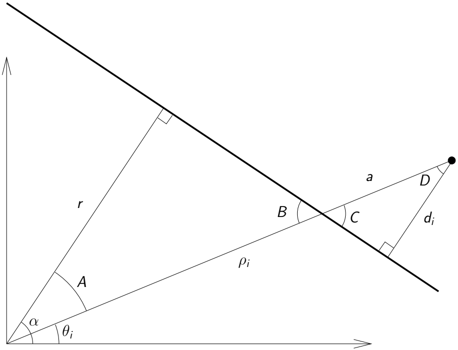
Compute d_i given data point (ρ_i, θ_i) and line parameters r and α:



From similar triangles we obtain $a = \frac{\rho_i d_i}{r + d_i}$

We can observe that, $B = C$

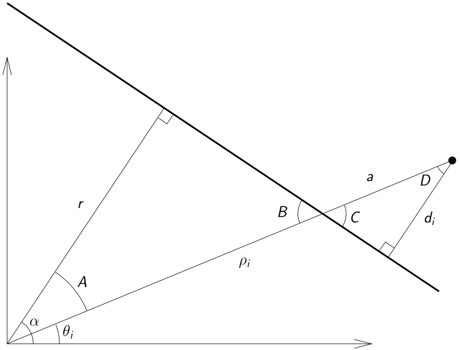
Compute d_i given data point (ρ_i, θ_i) and line parameters r and α:



From similar triangles we obtain $a = \frac{\rho_i d_i}{r + d_i}$

We can observe that, $B = C$. Therefore, $A = D$

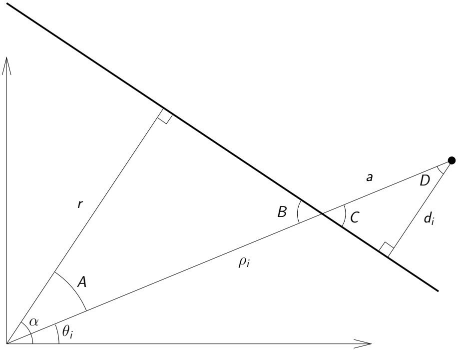
Compute d_i given data point (ρ_i, θ_i) and line parameters r and α:



From similar triangles we obtain $a = \frac{\rho_i d_i}{r + d_i}$

We can observe that, $B = C$. Therefore, $A = D = \alpha - \theta_i$.

Compute d_i given data point (ρ_i, θ_i) and line parameters r and α:

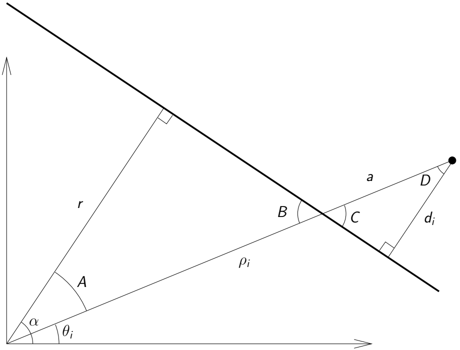


From similar triangles we obtain $a = \frac{\rho_i d_i}{r + d_i}$

We can observe that, $B = C$. Therefore, $A = D = \alpha - \theta_i$.

$$\cos(\alpha - \theta_i) = \frac{d_i}{a}$$

Compute d_i given data point (ρ_i, θ_i) and line parameters r and α:



From similar triangles we obtain $a = \frac{\rho_i d_i}{r + d_i}$

We can observe that, $B = C$. Therefore, $A = D = \alpha - \theta_i$.

$$\begin{aligned} \cos(\alpha - \theta_i) &= \frac{d_i}{a} \\ d_i &= \rho_i \cos(\theta_i - \alpha) - r \end{aligned}$$

$$d_i = \rho_i \cos(\theta_i - \alpha) - r$$

$$d_i = \rho_i \cos(\theta_i - \alpha) - r$$

This equation gives the $\perp$ distance to the line

$$d_i = \rho_i \cos(\theta_i - \alpha) - r$$

This equation gives the $\perp$ distance to the line. It also serves as an equation for the line when $d_i = 0$.

$$d_i = \rho_i \cos(\theta_i - \alpha) - r$$

This equation gives the $\perp$ distance to the line. It also serves as an equation for the line when $d_i = 0$.

To determine how line parameters r and α fit with the data, we could take the sum of squared distances,

$$d_i = \rho_i \cos(\theta_i - \alpha) - r$$

This equation gives the $\perp$ distance to the line. It also serves as an equation for the line when $d_i = 0$.

To determine how line parameters r and α fit with the data, we could take the sum of squared distances,

$$S = \sum_i d_i^2$$

$$d_i = \rho_i \cos(\theta_i - \alpha) - r$$

This equation gives the $\perp$ distance to the line. It also serves as an equation for the line when $d_i = 0$.

To determine how line parameters r and α fit with the data, we could take the sum of squared distances,

$$S = \sum_i d_i^2 = \sum_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

$$d_i = \rho_i \cos(\theta_i - \alpha) - r$$

This equation gives the $\perp$ distance to the line. It also serves as an equation for the line when $d_i = 0$.

To determine how line parameters r and α fit with the data, we could take the sum of squared distances,

$$S = \sum_i d_i^2 = \sum_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

We would then find parameters to minimize S

$$d_i = \rho_i \cos(\theta_i - \alpha) - r$$

This equation gives the $\perp$ distance to the line. It also serves as an equation for the line when $d_i = 0$.

To determine how line parameters r and α fit with the data, we could take the sum of squared distances,

$$S = \sum_i d_i^2 = \sum_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

We would then find parameters to minimize S. This approach is reasonable, but does not take advantage of the uncertainty of data points

$$d_i = \rho_i \cos(\theta_i - \alpha) - r$$

This equation gives the $\perp$ distance to the line. It also serves as an equation for the line when $d_i = 0$.

To determine how line parameters r and α fit with the data, we could take the sum of squared distances,

$$S = \sum_i d_i^2 = \sum_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

We would then find parameters to minimize S. This approach is reasonable, but does not take advantage of the uncertainty of data points. We should pay more attention to data points with the least variance...

We wish to weight individual data points by our confidence in their values.

We wish to weight individual data points by our confidence in their values. We will be more confident about values which are closer.

We wish to weight individual data points by our confidence in their values. We will be more confident about values which are closer. Therefore, we employ the following weight:

We wish to weight individual data points by our confidence in their values. We will be more confident about values which are closer. Therefore, we employ the following weight:

$$w_i = 1/\rho_i$$

We wish to weight individual data points by our confidence in their values. We will be more confident about values which are closer. Therefore, we employ the following weight:

$$w_i = 1/\rho_i$$

We can now redefine S to incorporate the estimated uncertainty of the data points,

We can now redefine S to incorporate the estimated uncertainty of the data points,

$$S = \sum_i w_i d_i^2$$

We can now redefine S to incorporate the estimated uncertainty of the data points,

$$S = \sum_i w_i d_i^2 = \sum_i w_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

We can now redefine S to incorporate the estimated uncertainty of the data points,

$$S = \sum_i w_i d_i^2 = \sum_i w_i(\rho_i \cos(\theta_i - \alpha) - r)^2$$

We wish to find r and α that yield a minimum value of S

We can now redefine S to incorporate the estimated uncertainty of the data points,

$$S = \sum_i w_i d_i^2 = \sum_i w_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

We wish to find r and α that yield a minimum value of S. A standard technique is to differentiate S w.r.t. r and α and solve for these values when the derivative is zero.

We can now redefine S to incorporate the estimated uncertainty of the data points,

$$S = \sum_i w_i d_i^2 = \sum_i w_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

We wish to find r and α that yield a minimum value of S. A standard technique is to differentiate S w.r.t. r and α and solve for these values when the derivative is zero.

$$\frac{\partial S}{\partial r} = 0 \qquad \frac{\partial S}{\partial \alpha} = 0$$

We can now redefine S to incorporate the estimated uncertainty of the data points,

$$S = \sum_i w_i d_i^2 = \sum_i w_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

We wish to find r and α that yield a minimum value of S. A standard technique is to differentiate S w.r.t. r and α and solve for these values when the derivative is zero.

$$\frac{\partial S}{\partial r} = 0 \qquad \frac{\partial S}{\partial \alpha} = 0$$

First apply the following trigonometric identity to S

We can now redefine S to incorporate the estimated uncertainty of the data points,

$$S = \sum_i w_i d_i^2 = \sum_i w_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

We wish to find r and α that yield a minimum value of S. A standard technique is to differentiate S w.r.t. r and α and solve for these values when the derivative is zero.

$$\frac{\partial S}{\partial r} = 0 \qquad \frac{\partial S}{\partial \alpha} = 0$$

First apply the following trigonometric identity to S

$$\cos(a - b) = \cos a \cos b + \sin a \sin b$$

$$S = \sum_{i=1}^{n} w_i(\rho_i \cos\theta_i \cos\alpha + \rho_i \sin\theta_i \sin\alpha - r)^2$$

$$S = \sum_{i=1}^{n} w_i(\rho_i \cos\theta_i \cos\alpha + \rho_i \sin\theta_i \sin\alpha - r)^2$$

$$\frac{\partial S}{\partial r} = 0 = \sum 2 w_i(\rho_i \cos\theta_i \cos\alpha + \rho_i \sin\theta_i \sin\alpha - r)(-1)$$

$$= -2\sum w_i \rho_i (\cos\theta_i \cos\alpha + \sin\theta_i \sin\alpha) + 2\sum w_i r$$

$$= -2\sum w_i \rho_i \cos(\theta_i - \alpha) + 2r\sum w_i$$

$$2r\sum w_i = 2\sum w_i \rho_i \cos(\theta_i - \alpha)$$

$$r = \frac{\sum w_i \rho_i \cos(\theta_i - \alpha)}{\sum w_i}$$

See [Arras, 1998] for the derivation of α,

See [Arras, 1998] for the derivation of α,

$$\alpha = \frac{1}{2}\operatorname{atan2}\left(\frac{\sum w_i \rho_i^2 \sin 2\theta_i - \frac{2}{\sum w_i}\sum\sum w_i w_j \rho_i \rho_j \cos\theta_i \sin\theta_j}{\sum w_i \rho_i^2 \cos 2\theta_i - \frac{1}{\sum w_i}\sum\sum w_i w_j \rho_i \rho_j \cos(\theta_i + \theta_j)}\right) + \frac{\pi}{2}$$

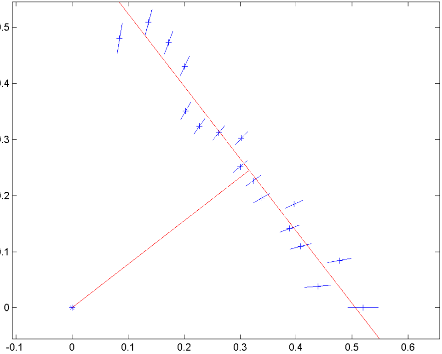
See [Arras, 1998] for the derivation of α,

$$\alpha = \frac{1}{2}\operatorname{atan2}\left(\frac{\sum w_i\rho_i^2\sin 2\theta_i - \frac{2}{\sum w_i}\sum\sum w_i w_j \rho_i \rho_j \cos\theta_i \sin\theta_j}{\sum w_i\rho_i^2\cos 2\theta_i - \frac{1}{\sum w_i}\sum\sum w_i w_j \rho_i \rho_j \cos(\theta_i + \theta_j)}\right) + \frac{\pi}{2}$$

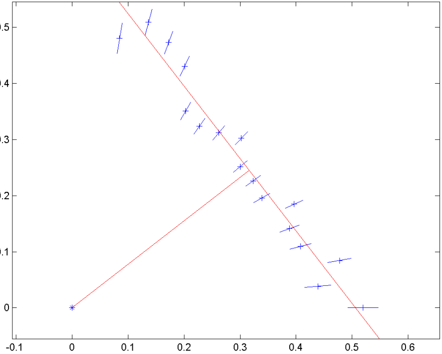
You should use `atan2` to return a result in $[-\pi, \pi]$

Example result:

Example result:



Example result:



Segmentation

- We made the assumption above that all data points came from the same line

Segmentation

- We made the assumption above that all data points came from the same line

Segmentation

- We made the assumption above that all data points came from the same line $\rightarrow$ this is generally a poor assumption

Segmentation

- We made the assumption above that all data points came from the same line $\rightarrow$ this is generally a poor assumption
- It is necessary to separate a set of measurements into subsets that all belong to the same feature

Segmentation

- We made the assumption above that all data points came from the same line $\rightarrow$ this is generally a poor assumption
- It is necessary to separate a set of measurements into subsets that all belong to the same feature

Segmentation

- We made the assumption above that all data points came from the same line $\rightarrow$ this is generally a poor assumption
- It is necessary to separate a set of measurements into subsets that all belong to the same feature; This is known as **segmentation**

Segmentation

- We made the assumption above that all data points came from the same line → this is generally a poor assumption
- It is necessary to separate a set of measurements into subsets that all belong to the same feature; This is known as **segmentation**
- The *Split-and-Merge* algorithm described below is relatively simple and was found to be the fastest, and among the most accurate in a recent study [Nguyen et al., 2005]

Segmentation

- We made the assumption above that all data points came from the same line $\rightarrow$ this is generally a poor assumption
- It is necessary to separate a set of measurements into subsets that all belong to the same feature; This is known as **segmentation**
- The *Split-and-Merge* algorithm described below is relatively simple and was found to be the fastest, and among the most accurate in a recent study [Nguyen et al., 2005]
- The idea:

Segmentation

- We made the assumption above that all data points came from the same line $\rightarrow$ this is generally a poor assumption
- It is necessary to separate a set of measurements into subsets that all belong to the same feature; This is known as **segmentation**
- The *Split-and-Merge* algorithm described below is relatively simple and was found to be the fastest, and among the most accurate in a recent study [Nguyen et al., 2005]
- The idea:
  - Fit a line using all points

Segmentation

- We made the assumption above that all data points came from the same line $\rightarrow$ this is generally a poor assumption
- It is necessary to separate a set of measurements into subsets that all belong to the same feature; This is known as **segmentation**
- The *Split-and-Merge* algorithm described below is relatively simple and was found to be the fastest, and among the most accurate in a recent study [Nguyen et al., 2005]
- The idea:
  - Fit a line using all points
  - Determine the point at maximum distance to this line

Segmentation

- We made the assumption above that all data points came from the same line $\rightarrow$ this is generally a poor assumption
- It is necessary to separate a set of measurements into subsets that all belong to the same feature; This is known as **segmentation**
- The *Split-and-Merge* algorithm described below is relatively simple and was found to be the fastest, and among the most accurate in a recent study [Nguyen et al., 2005]
- The idea:
  - Fit a line using all points
  - Determine the point at maximum distance to this line
  - If the distance of this worst fit point is greater than a threshold, split the line in two

Segmentation

- We made the assumption above that all data points came from the same line $\rightarrow$ this is generally a poor assumption
- It is necessary to separate a set of measurements into subsets that all belong to the same feature; This is known as **segmentation**
- The *Split-and-Merge* algorithm described below is relatively simple and was found to be the fastest, and among the most accurate in a recent study [Nguyen et al., 2005]
- The idea:
  - Fit a line using all points
  - Determine the point at maximum distance to this line
  - If the distance of this worst fit point is greater than a threshold, split the line in two
  - Continue fitting and splitting until no splittable segments remain

Segmentation

- We made the assumption above that all data points came from the same line $\rightarrow$ this is generally a poor assumption
- It is necessary to separate a set of measurements into subsets that all belong to the same feature; This is known as **segmentation**
- The *Split-and-Merge* algorithm described below is relatively simple and was found to be the fastest, and among the most accurate in a recent study [Nguyen et al., 2005]
- The idea:
  - Fit a line using all points
  - Determine the point at maximum distance to this line
  - If the distance of this worst fit point is greater than a threshold, split the line in two
  - Continue fitting and splitting until no splittable segments remain
  - Merge similar lines together

Split-and-Merge

1. Initialize a list of lists L. Create list of all points and place it into L.

Split-and-Merge

1. Initialize a list of lists L. Create list of all points and place it into L.
2. Get the next list, l_i from L. Fit a line to l_i.

Split-and-Merge

1. Initialize a list of lists L. Create list of all points and place it into L.
2. Get the next list, l_i from L. Fit a line to l_i.
3. Detect the worst fit point p, which has the largest distance d_p to the line.

Split-and-Merge

1. Initialize a list of lists L. Create list of all points and place it into L.
2. Get the next list, l_i from L. Fit a line to l_i.
3. Detect the worst fit point p, which has the largest distance d_p to the line.
4. If d_p is less than a threshold, go to step 2.

Split-and-Merge

1. Initialize a list of lists L. Create list of all points and place it into L.
2. Get the next list, l_i from L. Fit a line to l_i.
3. Detect the worst fit point p, which has the largest distance d_p to the line.
4. If d_p is less than a threshold, go to step 2.
5. Otherwise, split l_i at p into l_{i1} and l_{i2}. Replace l_i in L with l_{i1} and l_{i2}. Go to step 2.

Split-and-Merge

1. Initialize a list of lists L. Create list of all points and place it into L.
2. Get the next list, l_i from L. Fit a line to l_i.
3. Detect the worst fit point p, which has the largest distance d_p to the line.
4. If d_p is less than a threshold, go to step 2.
5. Otherwise, split l_i at p into l_{i1} and l_{i2}. Replace l_i in L with l_{i1} and l_{i2}. Go to step 2.
6. When all lists in L have been checked, merge similar lines.

Split-and-Merge

1. Initialize a list of lists L. Create list of all points and place it into L.
2. Get the next list, l_i from L. Fit a line to l_i.
3. Detect the worst fit point p, which has the largest distance d_p to the line.
4. If d_p is less than a threshold, go to step 2.
5. Otherwise, split l_i at p into l_{i1} and l_{i2}. Replace l_i in L with l_{i1} and l_{i2}. Go to step 2.
6. When all lists in L have been checked, merge similar lines.

More details are required for steps 3 and 6...

Split-and-Merge

1. Initialize a list of lists L. Create list of all points and place it into L.
2. Get the next list, l_i from L. Fit a line to l_i.
3. Detect the worst fit point p, which has the largest distance d_p to the line.
4. If d_p is less than a threshold, go to step 2.
5. Otherwise, split l_i at p into l_{i1} and l_{i2}. Replace l_i in L with l_{i1} and l_{i2}. Go to step 2.
6. When all lists in L have been checked, merge similar lines.

More details are required for steps 3 and 6...

Step 3:

We introduce a parameter called `minPointsPerLine` to prevent fitting lines to small sets of points to reduce the impact of noise

Split-and-Merge

1. Initialize a list of lists L. Create list of all points and place it into L.
2. Get the next list, l_i from L. Fit a line to l_i.
3. Detect the worst fit point p, which has the largest distance d_p to the line.
4. If d_p is less than a threshold, go to step 2.
5. Otherwise, split l_i at p into l_{i1} and l_{i2}. Replace l_i in L with l_{i1} and l_{i2}. Go to step 2.
6. When all lists in L have been checked, merge similar lines.

More details are required for steps 3 and 6...

Step 3:

We introduce a parameter called `minPointsPerLine` to prevent fitting lines to small sets of points to reduce the impact of noise. Therefore, the worst fit point should not be one of the first or last `minPointsPerLine`/2 points of the list.

Further details...

Step 6: "When all lists in L have been checked, merge similar lines."

Further details...

Step 6: "When all lists in L have been checked, merge similar lines."

We can measure the similarity of two lines $A = (r_1, \alpha_1)$ and $B = (r_2, \alpha_2)$ by comparing the two differences $|r_1 - r_2|$ and $|\alpha_1 - \alpha_2|$ to threshold values

Further details...

Step 6: "When all lists in L have been checked, merge similar lines."

We can measure the similarity of two lines $A = (r_1, \alpha_1)$ and $B = (r_2, \alpha_2)$ by comparing the two differences $|r_1 - r_2|$ and $|\alpha_1 - \alpha_2|$ to threshold values. Note that some care must be taken in computing $|\alpha_1 - \alpha_2|$

Further details...

Step 6: "When all lists in L have been checked, merge similar lines."

We can measure the similarity of two lines $A = (r_1, \alpha_1)$ and $B = (r_2, \alpha_2)$ by comparing the two differences $|r_1 - r_2|$ and $|\alpha_1 - \alpha_2|$ to threshold values. Note that some care must be taken in computing $|\alpha_1 - \alpha_2|$. e.g. The difference between 170° and -170° is 20°, not 340°

Further details...

Step 6: "When all lists in L have been checked, merge similar lines."

We can measure the similarity of two lines $A = (r_1, \alpha_1)$ and $B = (r_2, \alpha_2)$ by comparing the two differences $|r_1 - r_2|$ and $|\alpha_1 - \alpha_2|$ to threshold values. Note that some care must be taken in computing $|\alpha_1 - \alpha_2|$. e.g. The difference between 170° and -170° is 20°, not 340°.

Similar adjacent lines are merged by taking all of their source points and fitting a new line.

Further details...

Step 6: "When all lists in L have been checked, merge similar lines."

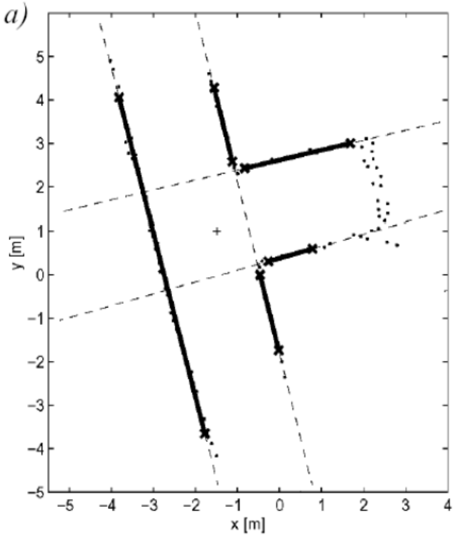
We can measure the similarity of two lines $A = (r_1, \alpha_1)$ and $B = (r_2, \alpha_2)$ by comparing the two differences $|r_1 - r_2|$ and $|\alpha_1 - \alpha_2|$ to threshold values. Note that some care must be taken in computing $|\alpha_1 - \alpha_2|$. e.g. The difference between $170°$ and $-170°$ is $20°$, not $340°$.

Similar adjacent lines are merged by taking all of their source points and fitting a new line.

(The method described above is similar, but slightly simplified from that described in [Nguyen et al., 2005])

Once we have lines, we can use them directly or else determine **line segments**; The projection of the start and end data points onto the line yield a line segment.

Once we have lines, we can use them directly or else determine **line segments**; The projection of the start and end data points onto the line yield a line segment.



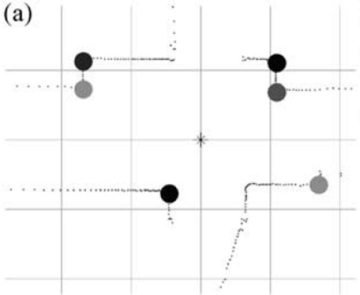
Corners are points of intersection between two lines

Corners are points of intersection between two lines. We may require that the angle between the lines is large

Corners are points of intersection between two lines. We may require that the angle between the lines is large. A corner can be further classified as convex or concave depending upon the orientation of the lines w.r.t. the robot.

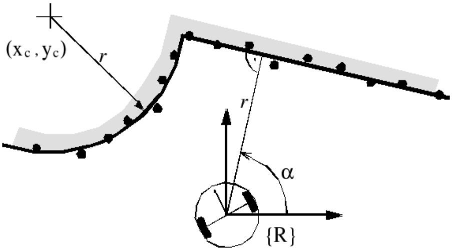
Corners are points of intersection between two lines. We may require that the angle between the lines is large. A corner can be further classified as convex or concave depending upon the orientation of the lines w.r.t. the robot.[Tomatis et al., 2003]

Corners are points of intersection between two lines. We may require that the angle between the lines is large. A corner can be further classified as convex or concave depending upon the orientation of the lines w.r.t. the robot.[Tomatis et al., 2003]



We can use the same general approach for extracting **circles** as we we used for extracting lines

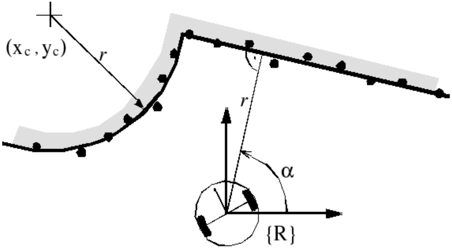
We can use the same general approach for extracting **circles** as we we used for extracting lines



Model for *circles:*

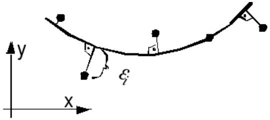
$$(x + x_c)^2 + (y + y_c)^2 - r^2 = 0$$

We can use the same general approach for extracting **circles** as we we used for extracting lines



Model for *circles:*

$$(x + x_c)^2 + (y + y_c)^2 - r^2 = 0$$



$$(x_i + x_c)^2 + (y_i + y_c)^2 - r^2 = \varepsilon_i$$

References

Arras, K. (1998).
An introduction to error propagation.
Technical Report EPFL-ASL-TR-98-01 R3, EPFL.

Nguyen, V., Martinelli, A., Tomatis, N., and Siegwart, R. (2005).
A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics.
In *Proceedings of the IEEE/RSJ Intenational Conference on Intelligent Robots and Systems*.

Tomatis, N., Nourbakhsh, I., and Siegwart, R. (2003).
Hybrid simultaneous localization and map building: a natural integration of topological and metric.
Robotics and Autonomous Systems, 44:3–14.