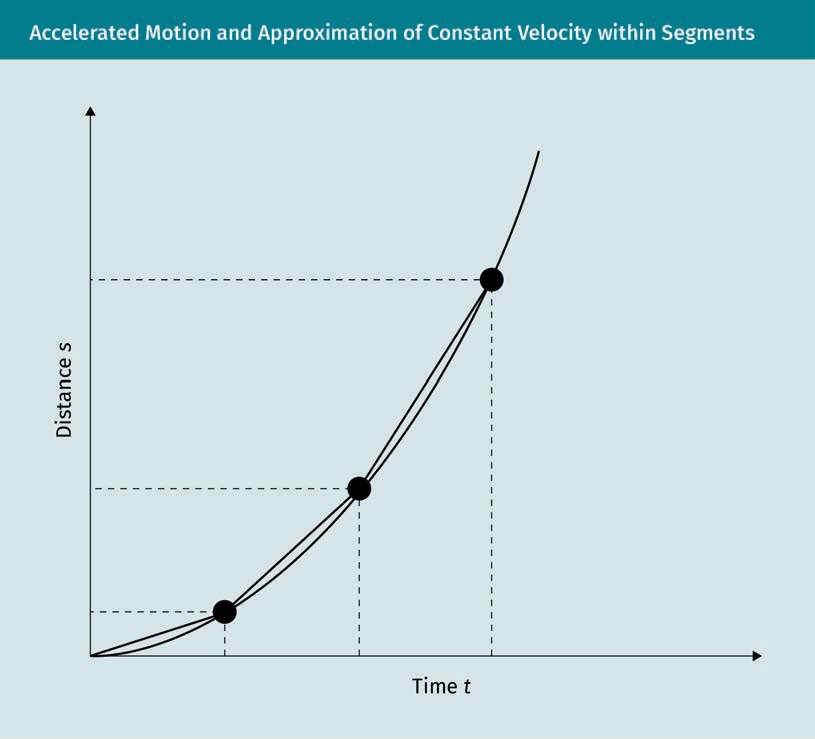
**Segments to Calculate Velocity**

**[Streckenabschnitte bei der Geschwindigkeitsberechnung]**



**Accelerated Motion and Approximation of Constant Velocity within Segements**

**[Beschleunigte Bewegung und Näherung durch konstante Geschwindigkeiten innerhalb der Zeitabschnitte]**



Time

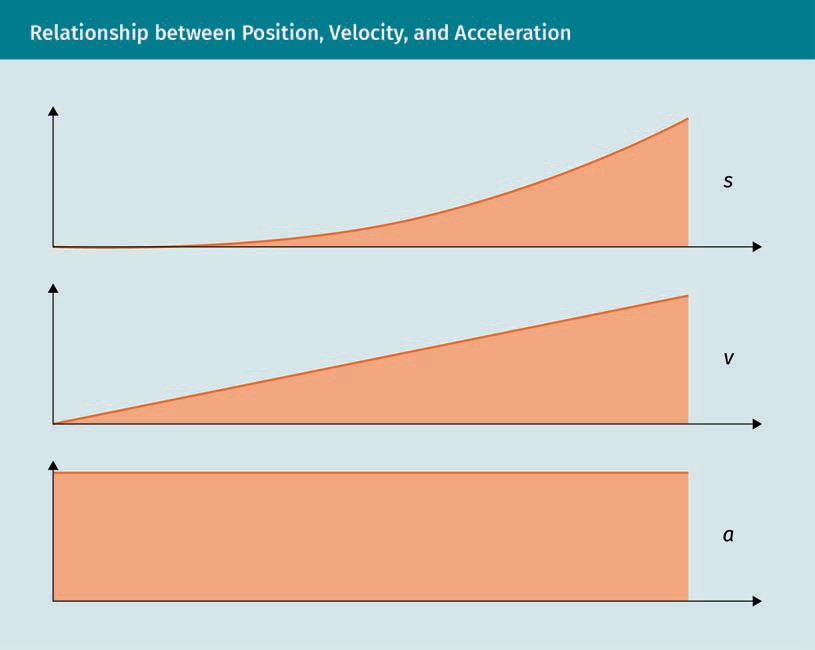
[Zeit]

Distance

[Strecke]

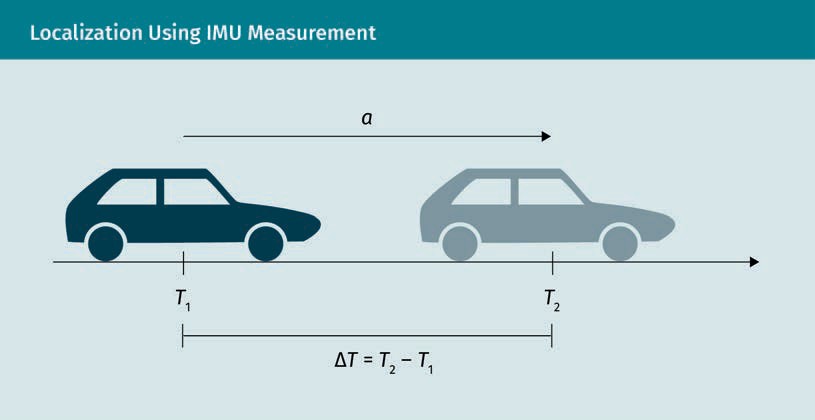
**Relationship between Position, Velocity, and Acceleration**

**[Zusammenhang zwischen Position, Geschwindigkeit und Beschleunigung]**

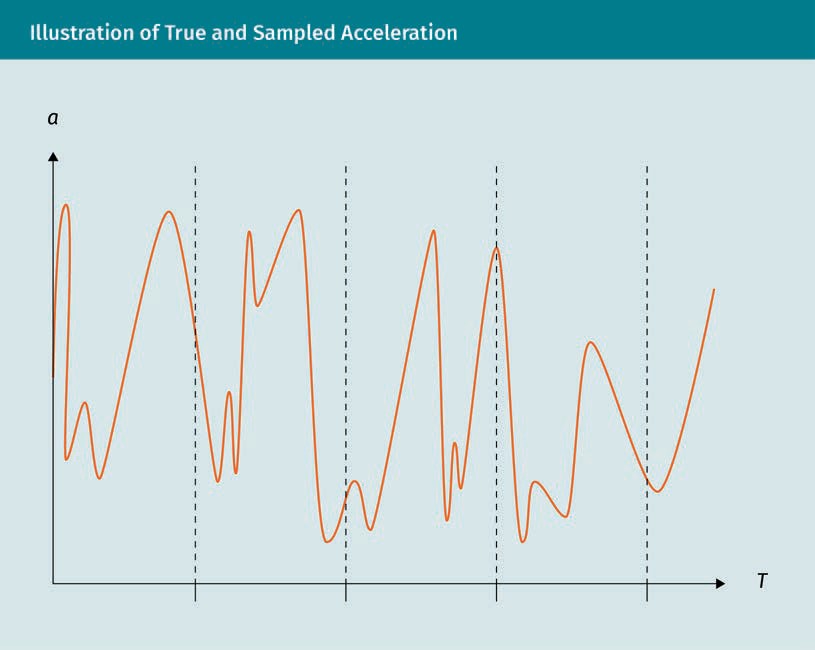


**Localization Using IMU Measurement**

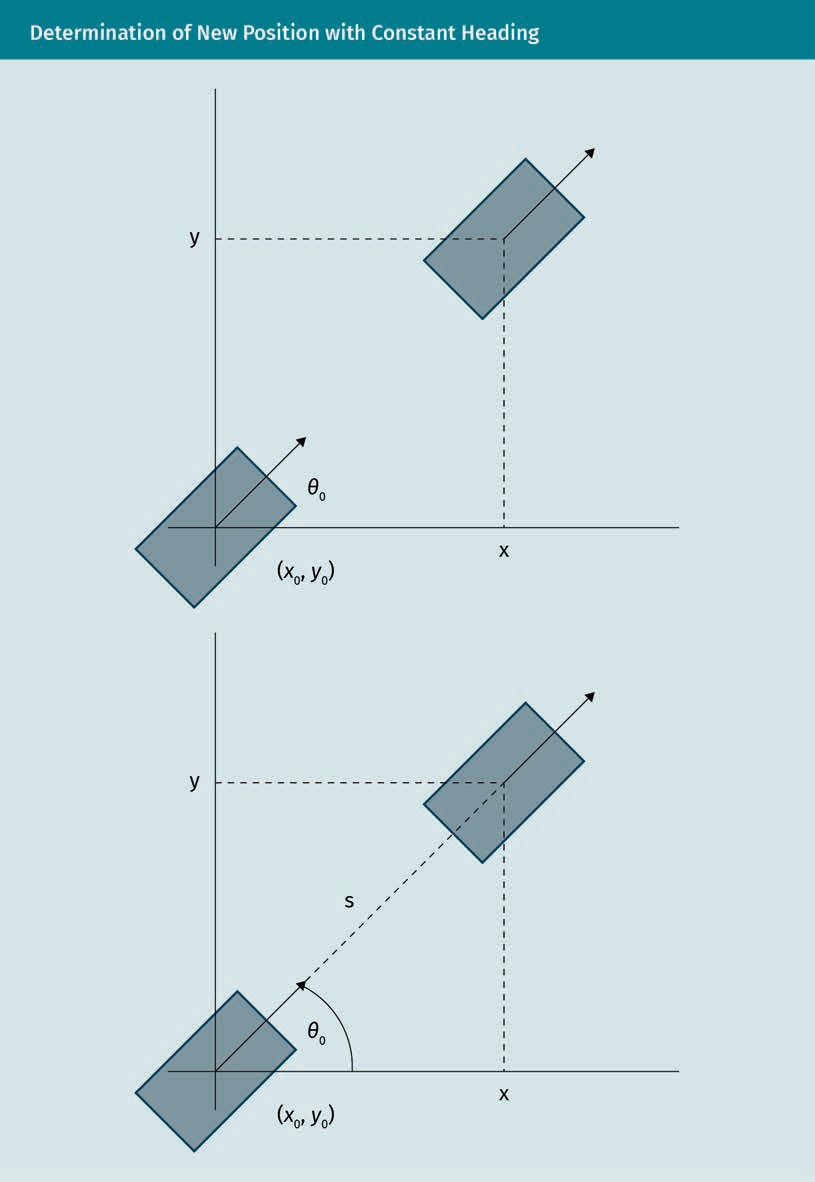
**[Lokalisierung durch IMU-Messung]**



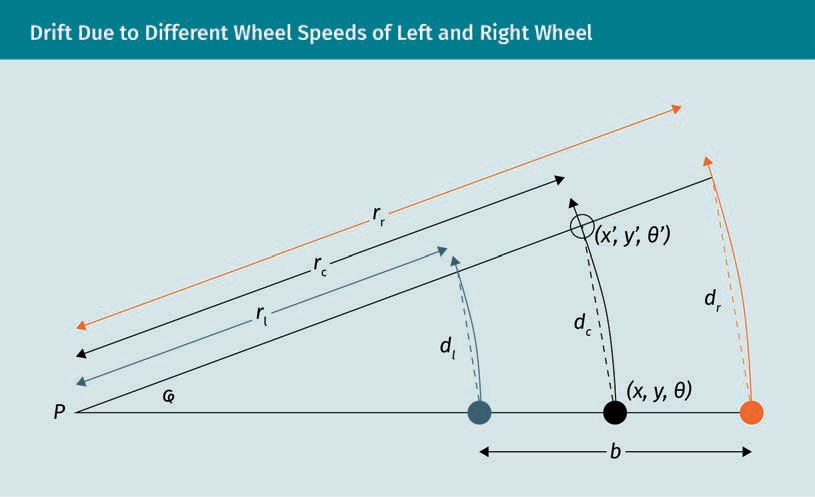
**Illustration of True and Sampled Acceleration**

**[Veranschaulichung der tatsächlichen und der gemessenen Beschleunigung]**

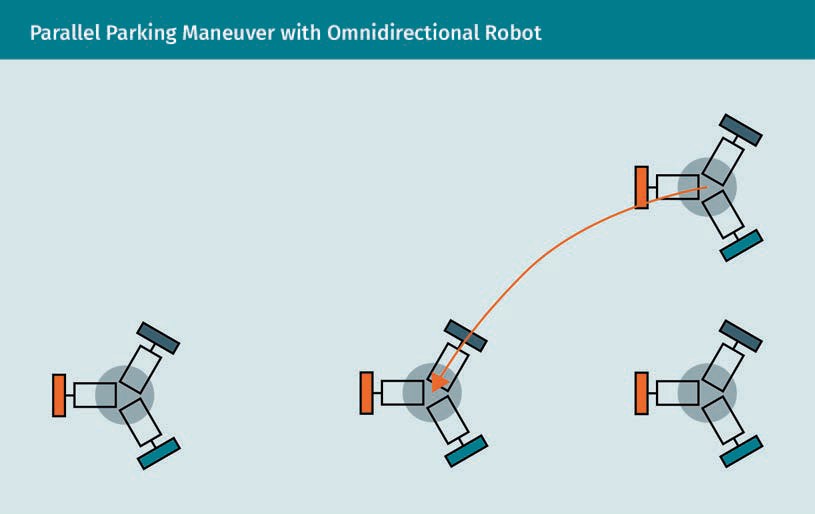
**Determination of New Position with Constant Heading**

**[Bestimmung der neuen Position bei konstantem Kurs]**

**Drift Due to Different Wheel Speeds of Left and Right Wheel**

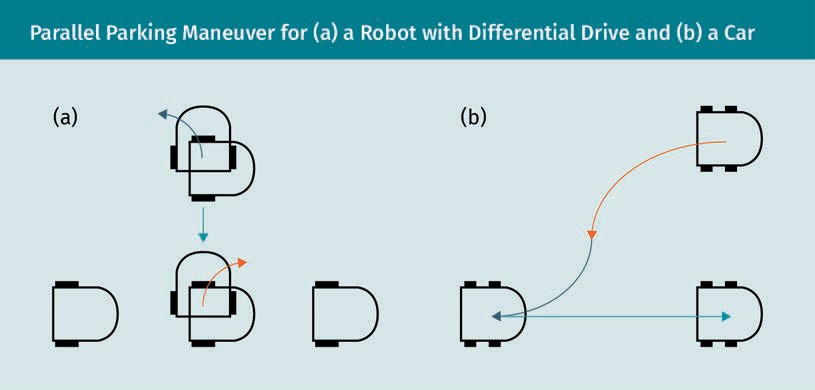
**[Abweichung (Abdrift) aufgrund unterschiedlicher Radgeschwindigkeiten des linken und rechten Rades]**

**Parallel Parking Maneuver with Omnidirectional Robot**

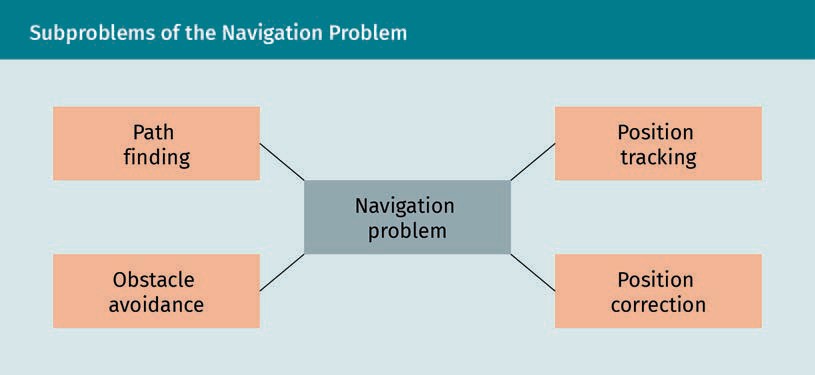
**[Seitliches Einparken mit omnidirektionalem Roboter]**

**Parallel Parking Maneuver for (a) a Robot with Differential Drive and (b) a Car**

**[Seitliches Einparken mit (a) einem Roboter mit Differentialantrieb und (b) einem Auto]**

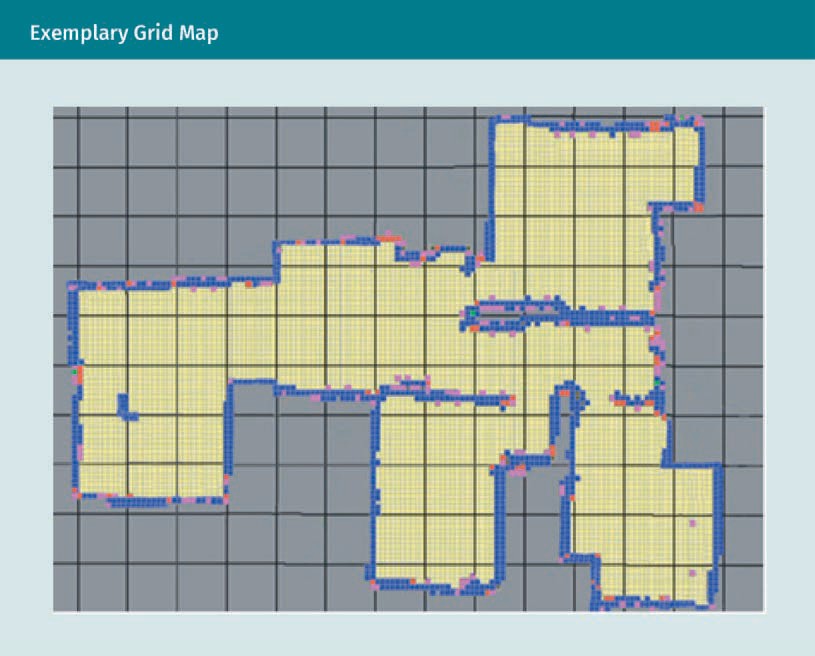


**Subproblems of the Navigation Problem**

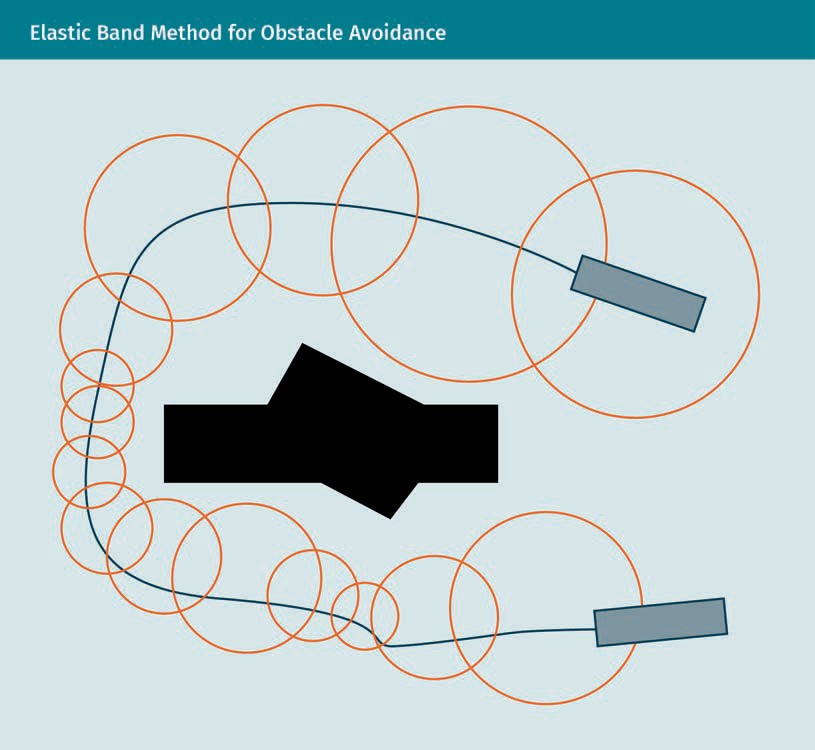
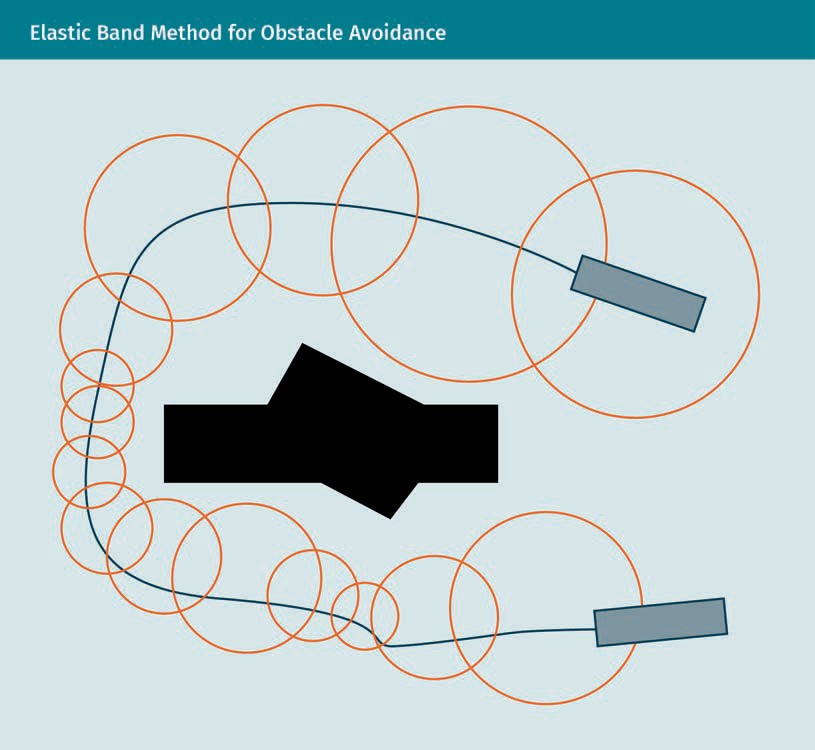
**[Teilprobleme bei der Navigation]**

|  |  |
| --- | --- |
| Path finding | Wegfindung |
| Obstacle avoidance | Hindernisvermeidung |
| Navigation problem | Navigation |
| Position tracking | Positionsbestimmung |
| Position correction | Positionskorrektur |

**Exemplary Grid Map**

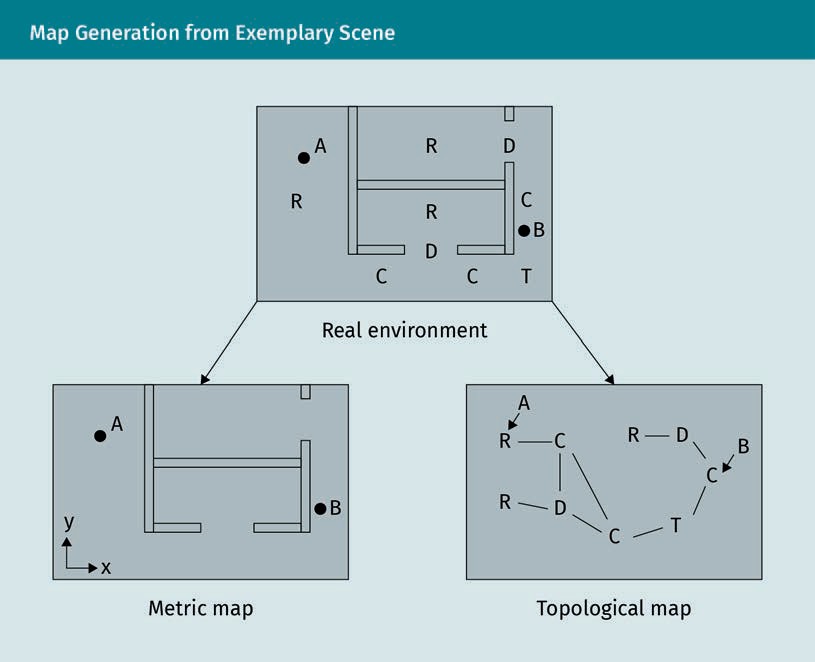
**[Beispielhafte Rasterkarte]**

**Elastic Band Method for Obstacle Avoidance**

**[Gummiband-Methode zur Hindernisvermeidung]**

**Map Generation from Exemplary Scene**

**[Kartenerstellung aus Beispielszenen]**



Real environment

[Reale Umgebung]

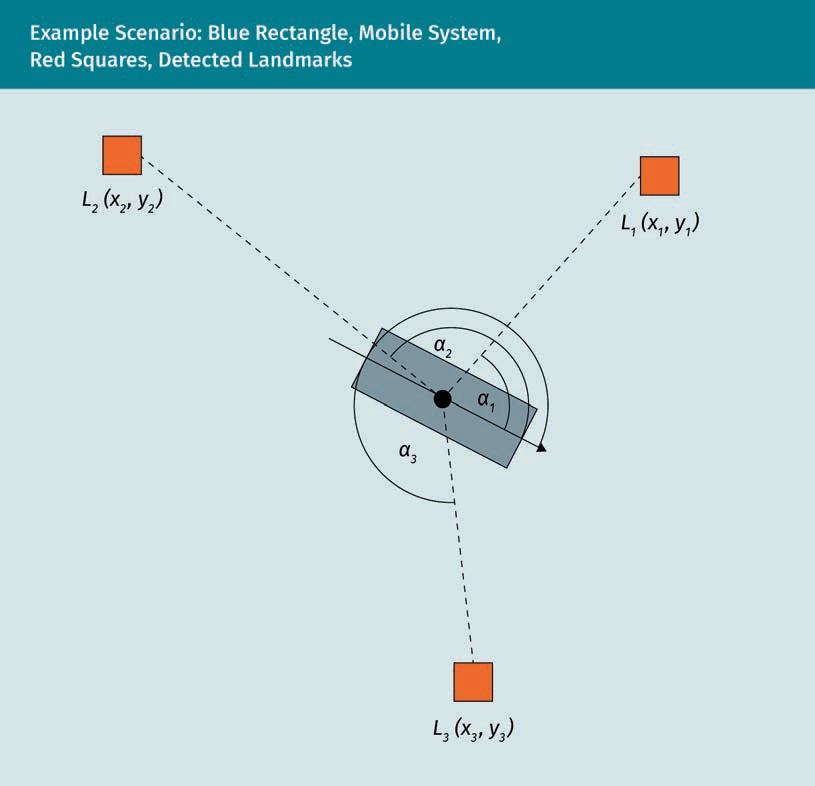
Metric map

[Metrische Karte]

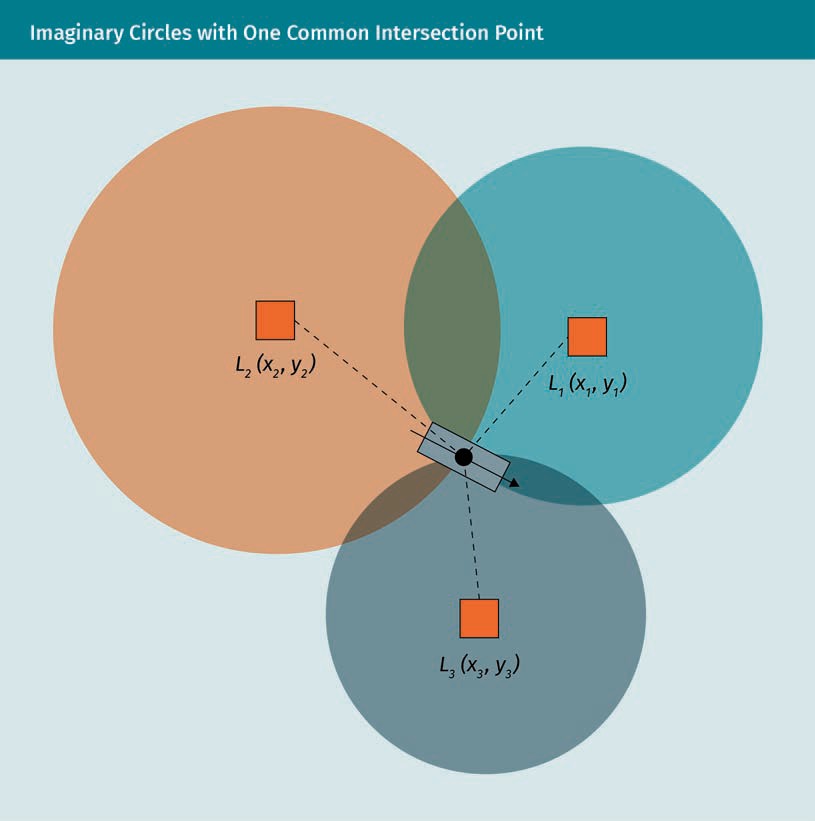
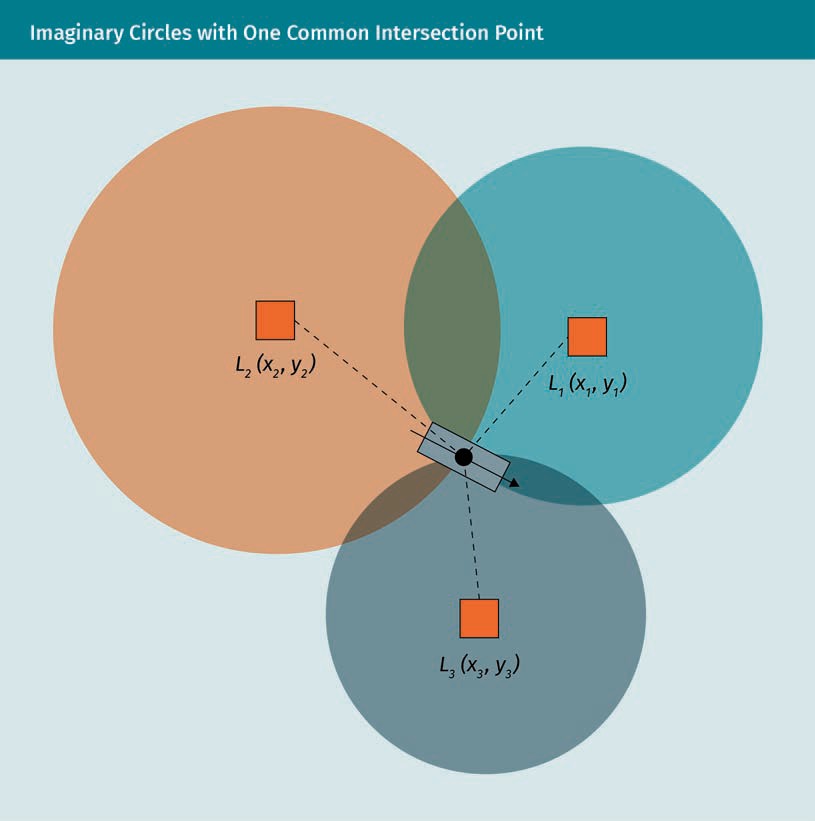
Topological map

[Topologische Karte]

**Example scenario: Blue Rectangle, Mobile System, Red Squares, Detected Landmarks**

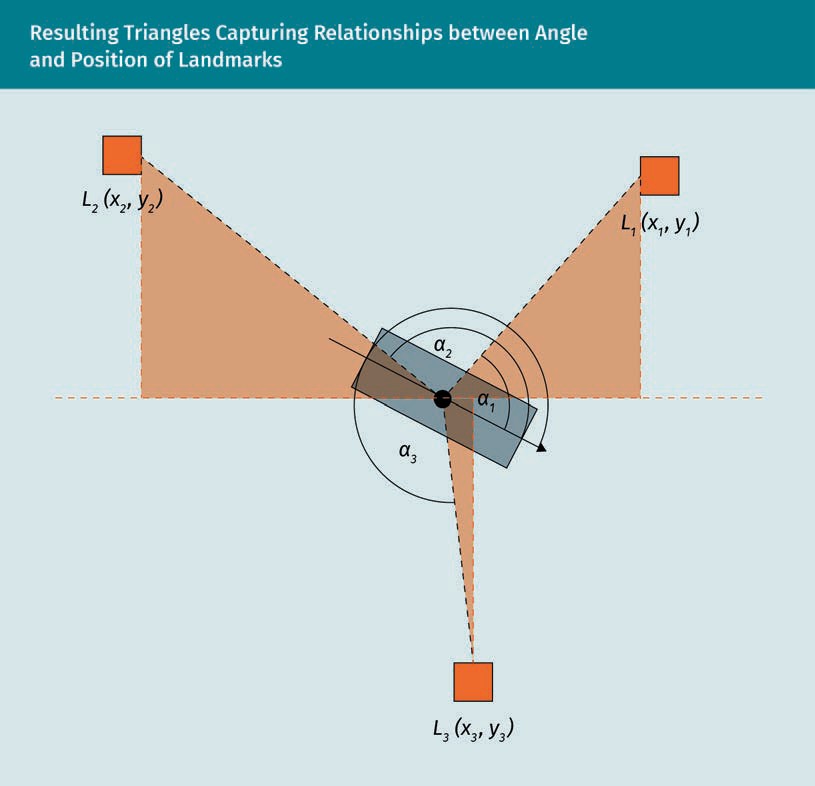
**[Beispielszenario: Blaues Rechteck – Mobiles System, rote Quadrate – ausgemachte Orientierungspunkte]**

**Imaginary Circles with One common Intersection Point**

**[Imaginäre Kreise mit einem gemeinsamen Schnittpunkt]**

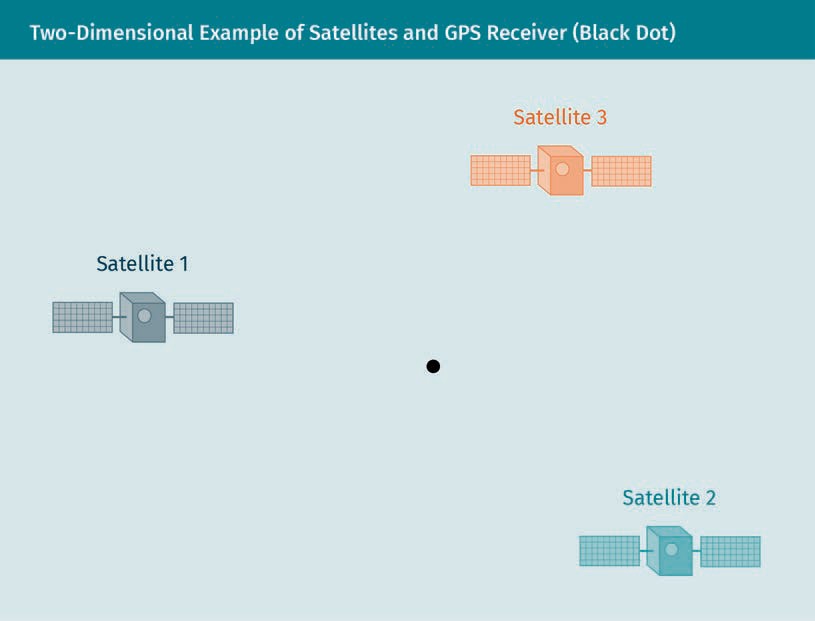
**Resulting Triangles Capturing Relationships between Angle and Position of Landmarks**

**[Daraus resultierende Dreiecke mit den Zusammenhängen zwischen Winkel und Position der Orientierungspunkte]**



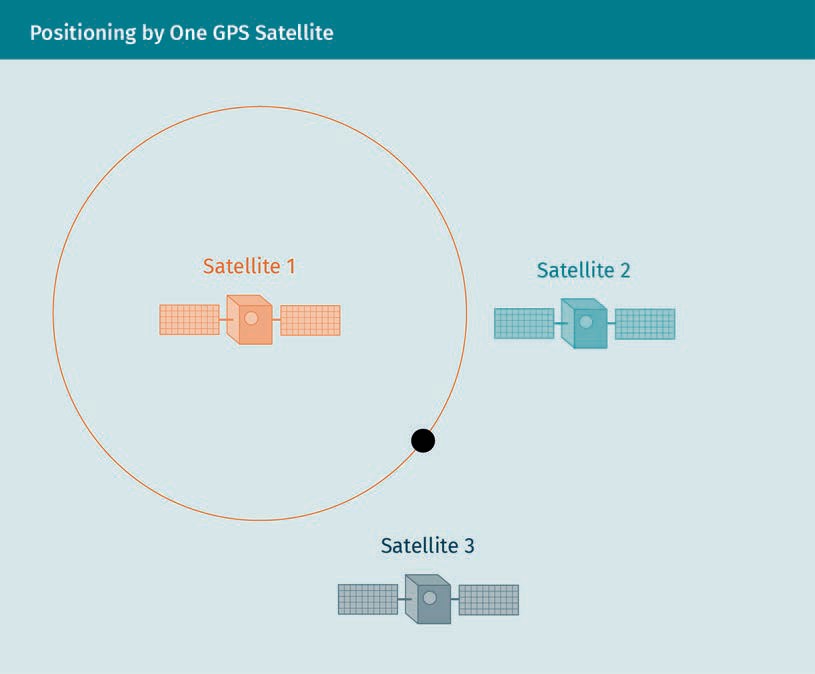
**Two-Dimentional Example of Satellites and GPS Receiver (Black Dot)**

**[Zweidimensionales Beispiel von Satelliten und GPS-Empfänger (schwarzer Punkt)]**



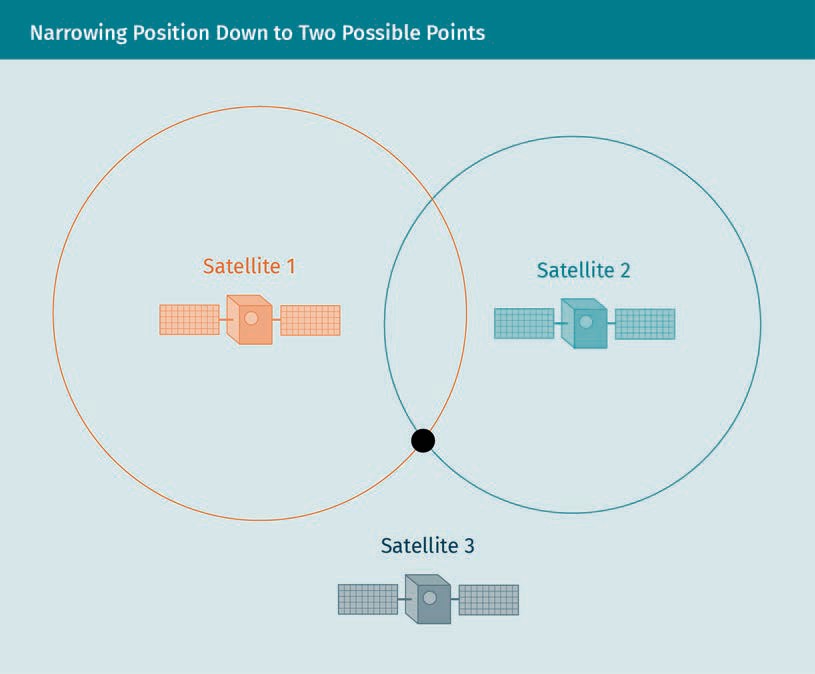
|  |  |
| --- | --- |
| Satellite | Satellit |

**Positioning by One GPS Satellite**

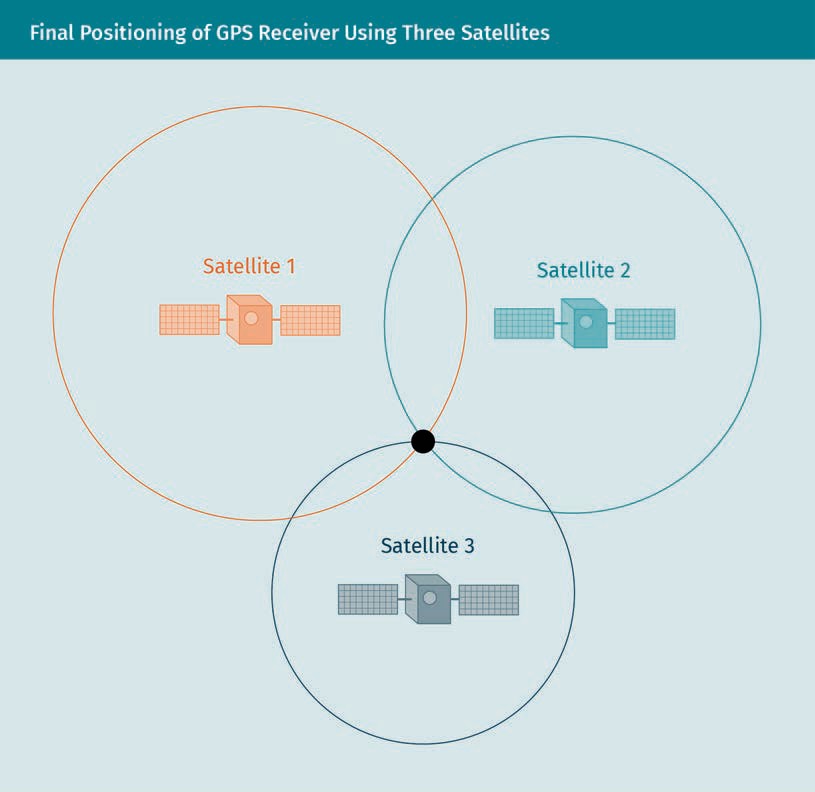
**[Ortung durch einen GPS-Satelliten]**

**Narrowing Position Down to Two Possible Points**

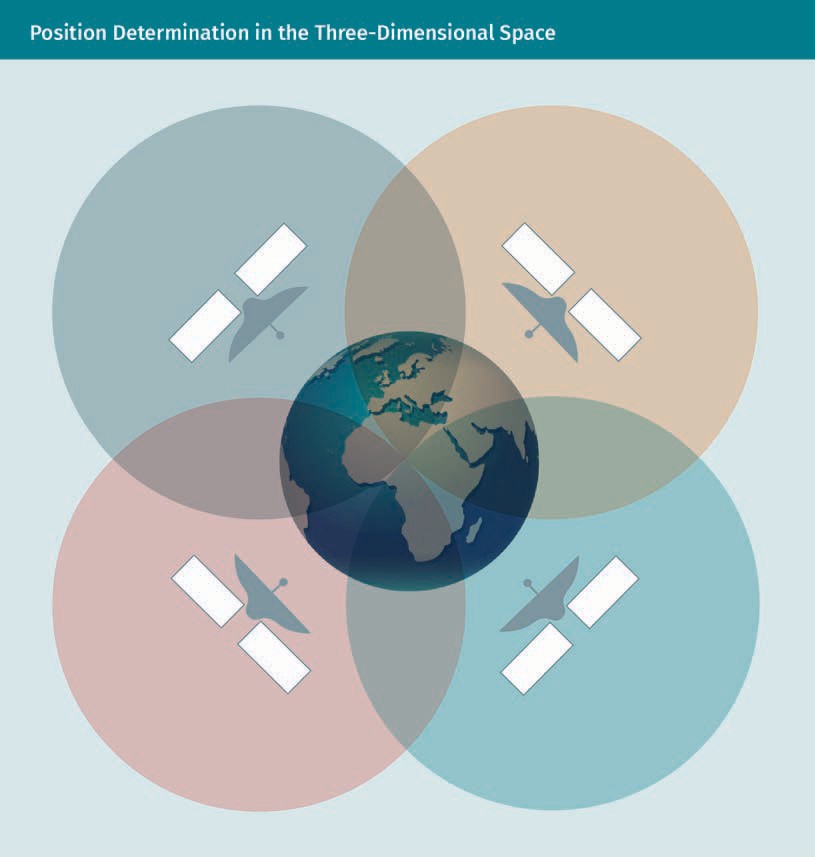
**[Eingrenzung der Position auf zwei mögliche Punkte]**



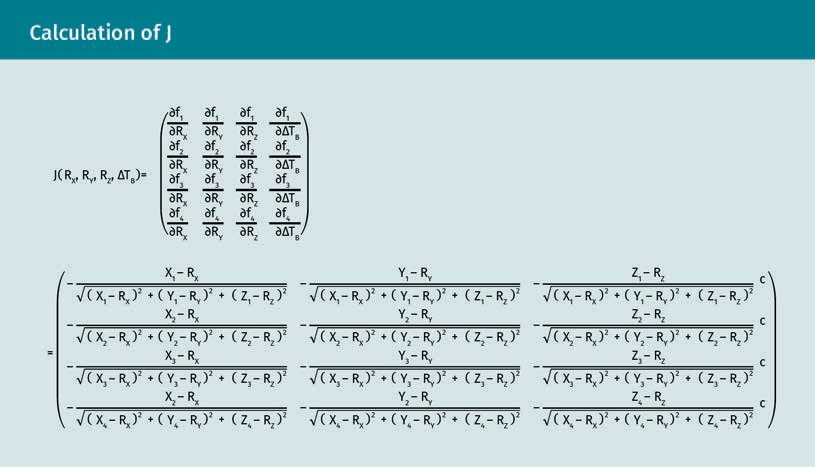
**Final Positioning of GPS Receiver Using Three Satellites**

**[Endgültige Positionierung des GPS-Empfängers mit Hilfe von drei Satelliten]**

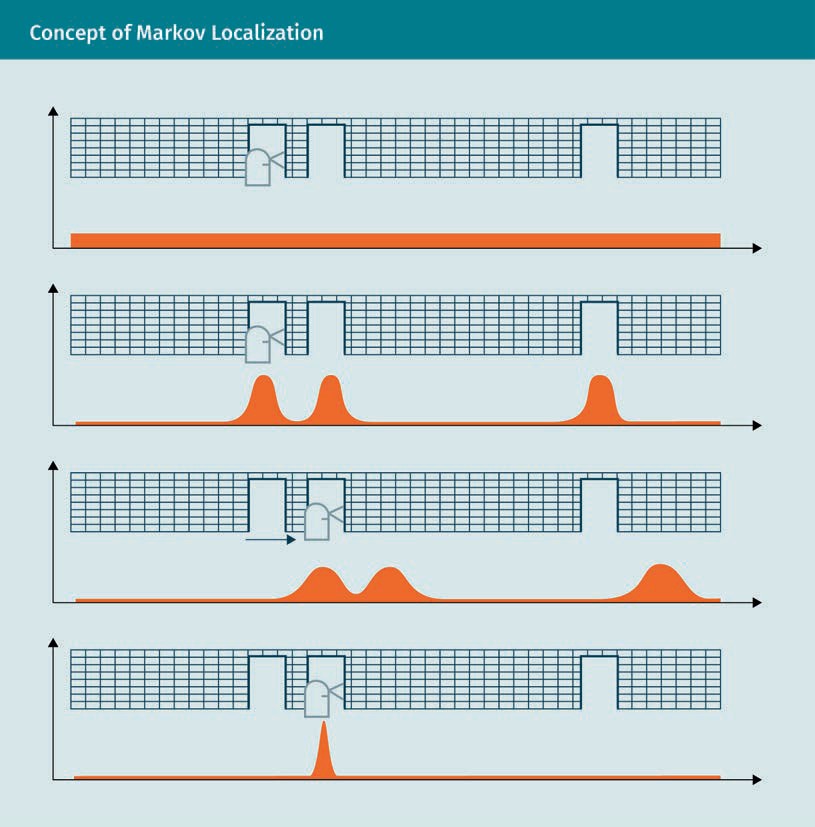
**Position Determionation in the Three-Dimensional Space**

**[Positionsbestimmung im dreidimensionalen Raum]**

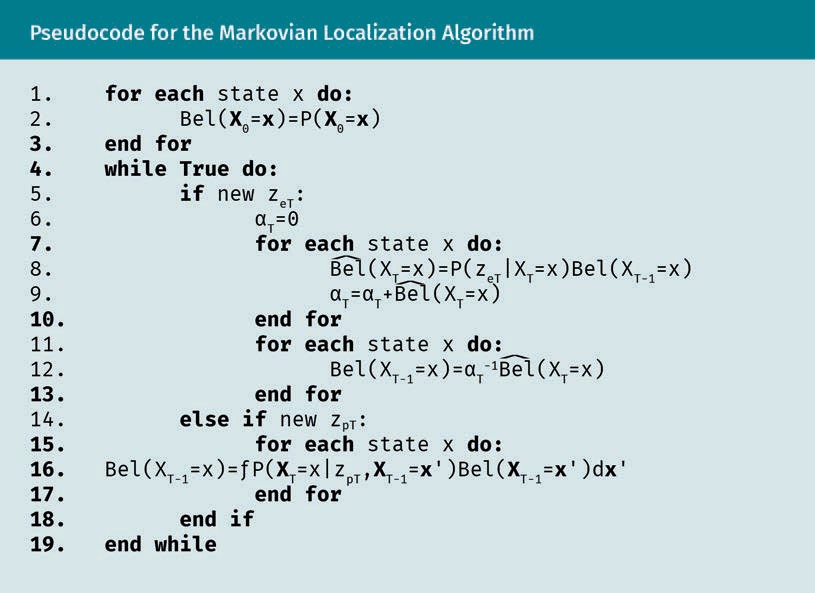
**Calculation of J**

**[Berechnung von J****]**

**Concept of Markov Localization**

**[Konzept der Markow-Lokalisierung]**

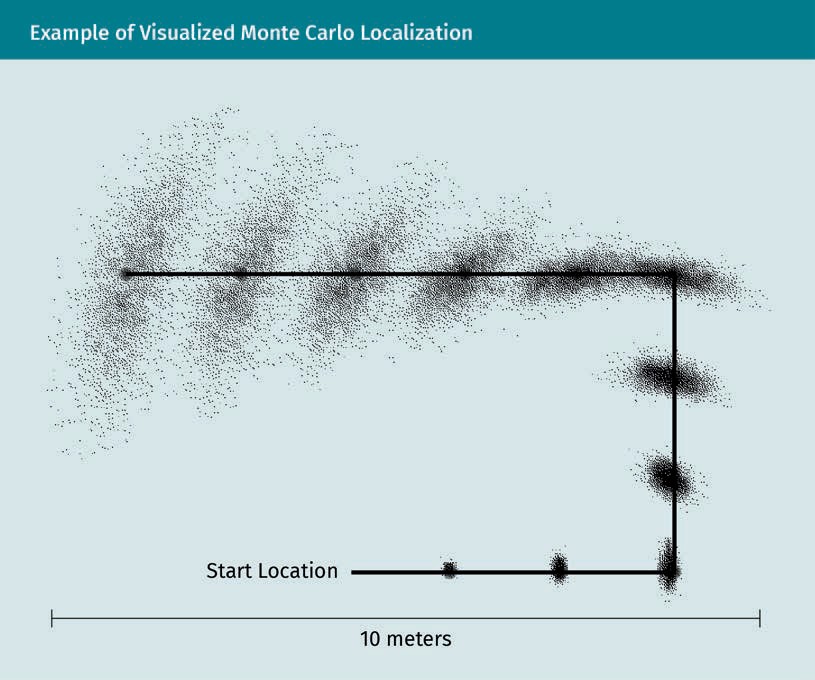
**Pseudocode for the Markovian Localization Algorithm**

**[Pseudocode für den Markow-Lokalisierungsalgorithmus]**

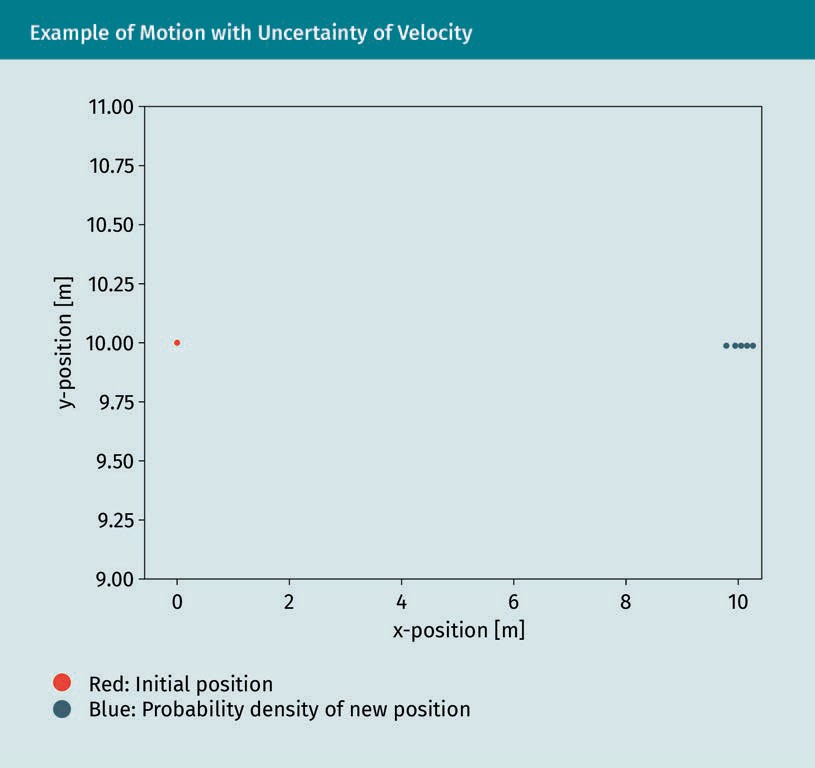
|  |  |
| --- | --- |
| for each state x do: | for each state x do: |
| Bel (XΘ=x) | Bel (XΘ=x) |
| end for | end for |
| while True do: | while True do: |
| if new zeT: | if new zeT: |
| αT=Θ | αT=Θ |
| for each state x do: | for each state x do: |
|  |  |
|  |  |
| end for | end for |
|  |  |
| else if new zpT: | else if new zpT: |
|  |  |
| end for | end for |
| end if | end if |
| end while | end while |

**Example of Visualized Monte Carlo Localization**

**[Beispiel einer visualisierten Monte-Carlo-Lokalisierung]**

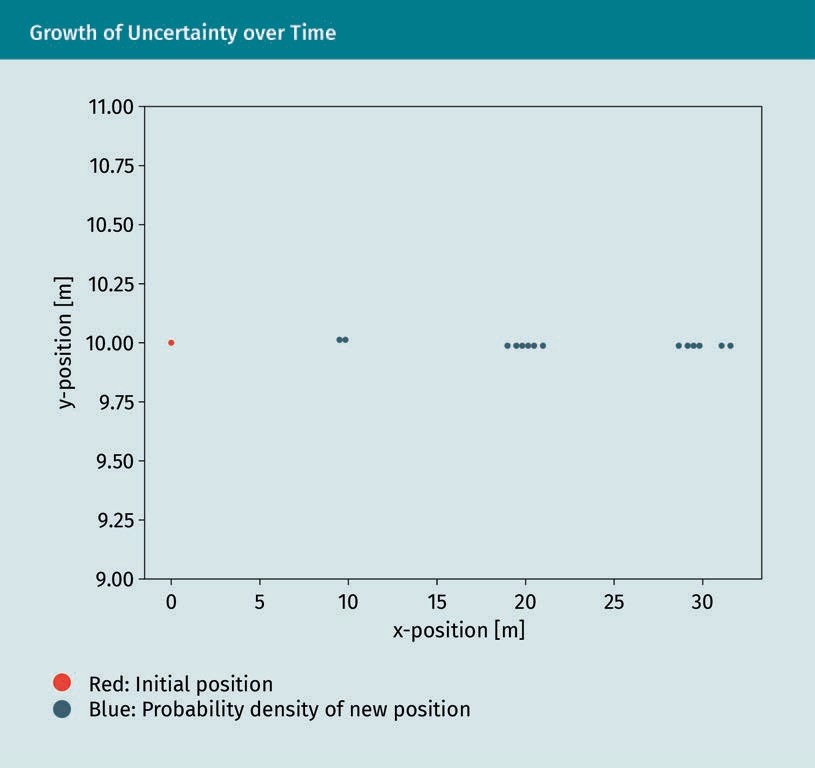


|  |  |
| --- | --- |
| Start Location | Startort |
| 10 meters | 10 Meter |

**[Beispiel für eine Bewegung mit unsicherer Geschwindigkeit]**

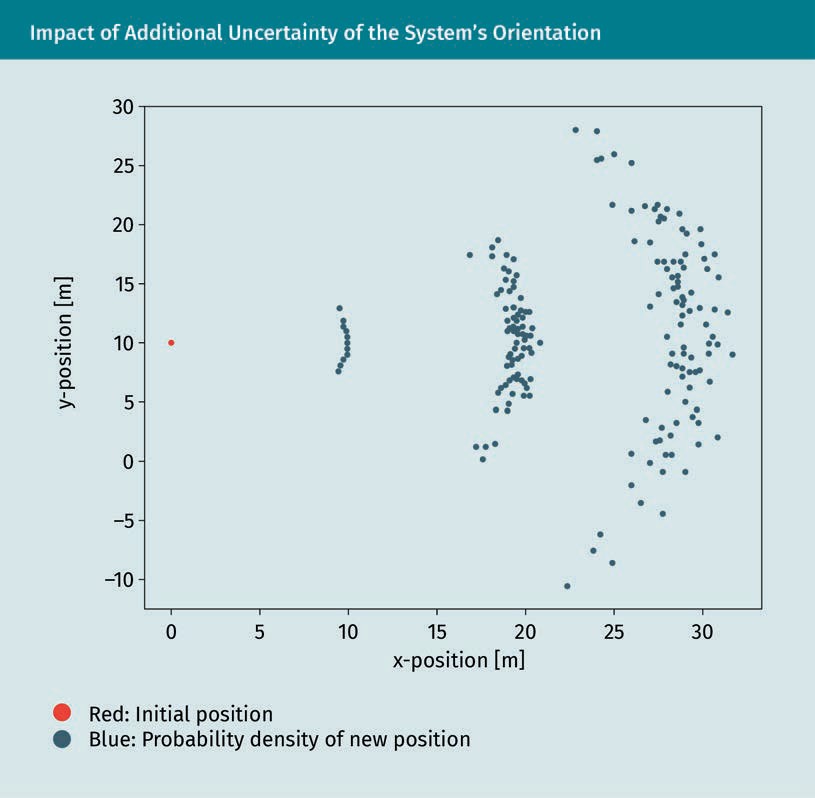
|  |  |
| --- | --- |
| y-position [m] | y-Position [m] |
| x-position [m] | x-Position [m] |
| Red: Initial position | Rot: Ausgangslage |
| Blue: Probability density of new position | Blau: Wahrscheinlichkeitsdichte der neuen Position |

**Growth of uncertainty over time**

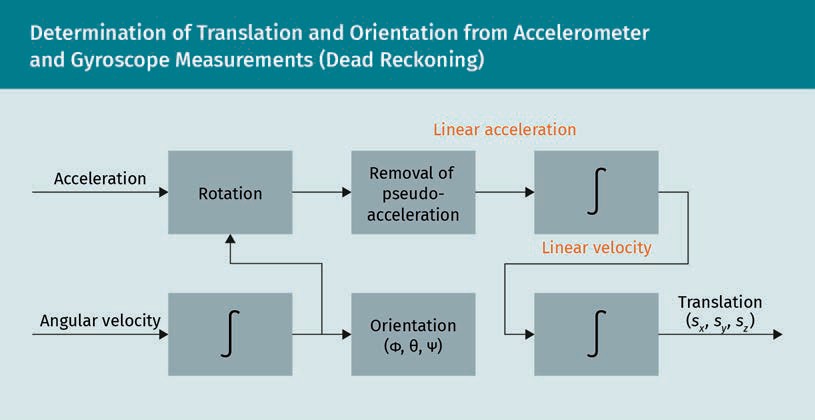
**[Zunahme der Unsicherheit im Laufe der Zeit]**

**Impact of Additional Uncertainty of the System’s Orientation**

**[Auswirkungen zusätzlicher Unsicherheiten bei der Orientierung des Systems]**

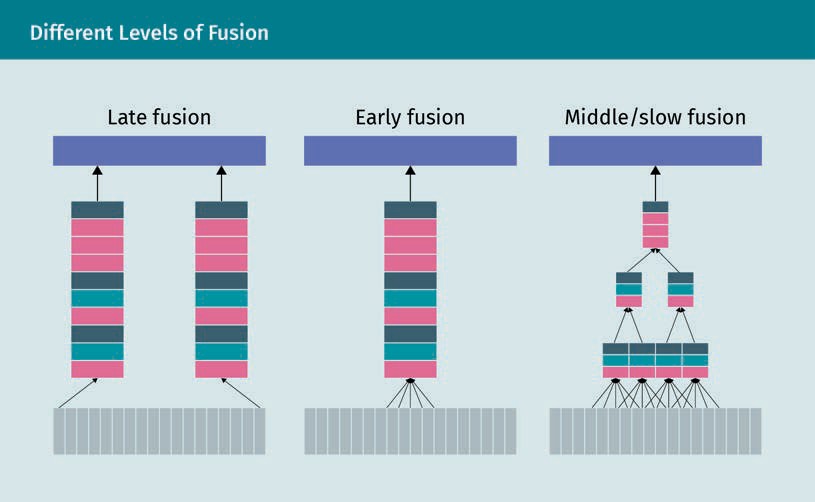


**Determination of Translation and Orientation from Acelerometer and Gyroscope Measurements (Dead Reckoning)**

**[Bestimmung von Translation (Verschiebung) und Orientierung (Drehung) aus Beschleunigungs- und Gyroskop-Messungen (Koppelnavigation)]**

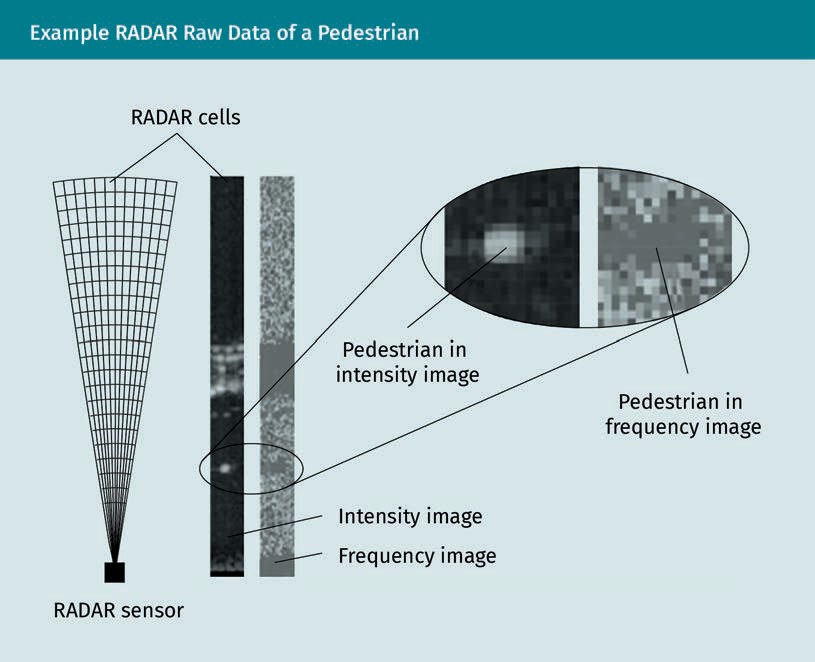
|  |  |
| --- | --- |
| Acceleration | Beschleunigung |
| Rotation | Drehung |
| Removal of pseudo-acceleration | Beseitigung der Pseudobeschleunigung |
| Linear acceleration | Lineare Beschleunigung |
| Linear velocity | Lineare Geschwindigkeit |
| Angular velocity | Winkelgeschwindigkeit |
| Orientation | Orientierung |
| Translation | Translation |

**Different Levels of Fusion**

**[Verschiedene Ebenen der Datenfusion]**

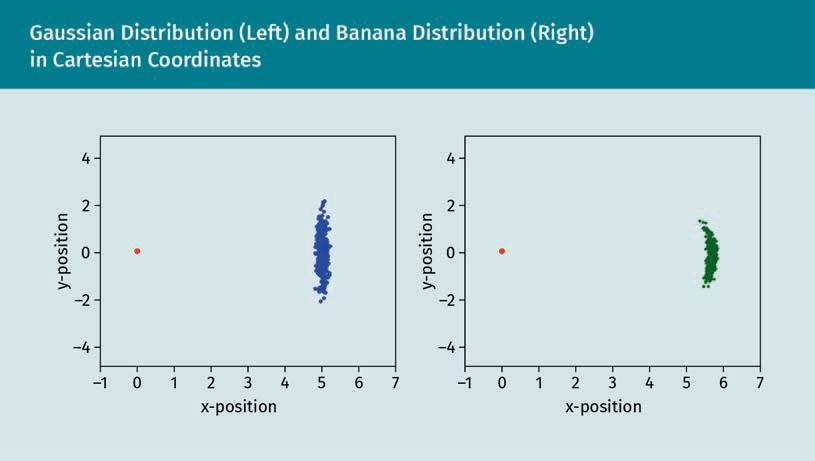
|  |  |
| --- | --- |
| Late fusion | Späte Fusion |
| Early fusion | Frühe Fusion |
| Middle/slow fusion | Mittlere/langsame Fusion |

**Example RADAR Raw Data of a Pedestrian**

**[Beispiel: RADAR-Rohdaten eines Fußgängers]**

|  |  |
| --- | --- |
| RADAR cells | RADAR-Zellen |
| RADAR sensor | RADAR-Sensor |
| Pedestrian in intensity image | Fußgänger im Intensitätsbild |
| Pedestrian in frequency image | Fußgänger im Frequenzbild |
| Intensity image | Intensitätsbild |
| Frequency image | Frequenzbild |

**Gaussian Distribution (Left) and Banana Distribution (Right) in Cartesian Coordinates**

**[Gaußsche Verteilung (links) und Bananenverteilung (rechts) in kartesischen Koordinaten]**

|  |  |
| --- | --- |
| y-position | y-Position |
| x-position | x-Position |

**Example of Python Implementation**

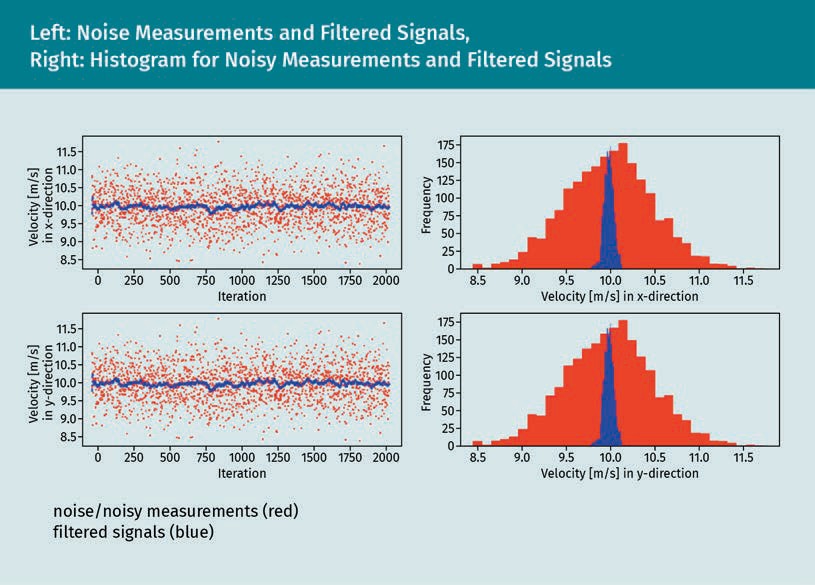
**[Beispiel einer Python-Implementierung]**



|  |  |
| --- | --- |
| import numpy as np | import numpy as np |
| from numpy import transpose, indentity, outer | from numpy import transpose, indentity, outer |
| from numpy.linalg import inv | from numpy.linalg import inv |
| from matplotlib import pyplot as plt | from matplotlib import pyplot as plt |
| std\_acc = 0.001 | std\_acc = 0.001 |
| #we assume a standard deviation of the acceleration of 0.001 (m/s^2) | #Wir nehmen eine Standardabweichung der Beschleunigung von 0,001 (m/s^2) an |
| std\_meas = 0.3 | std\_meas = 0.3 |
| #standard deviation of the measurement | #Standardabweichung der Messung |
| x=np.array([0,0]).transpose | x=np.array([0,0]).transpose |
| #initial state | #Anfangszustand |
| P=10\*identity(“) | P=10\*identity(“) |
| #initial state covariance | #Kovarianz im Anfangszustand |
| R=std\_meas\*\*2 | R=std\_meas\*\*2 |
| #measurement noise | #Messrauschen |
| ###State transition matrix### | ###Zustandsübergangsmatrix### |
| def F(dt): | def F(dt): |
| return np.array([[1,dt],[0,1]]) | return np.array([[1,dt],[0,1]]) |
| ###Process noise covariance### | ###Kovarianz des Prozessrauschens### |
| def Q(dt): | def Q(dt): |
| Q=std\_acc\*np.array([[dt\*\*3/3,dt\*\*2/2],[dt\*\*2/2,dt]]) | Q=std\_acc\*np.array([[dt\*\*3/3,dt\*\*2/2],[dt\*\*2/2,dt]]) |
| return Q | return Q |
| ###Measurement model### | ###Messmodell### |
| def H(): | def H(): |
| return np.array([0,1]) | return np.array([0,1]) |
| ###Prediction step### | ###Vorhersageschritt### |
| def predict(x,P,dt): | def predict(x,P,dt): |
| x=F(dt).dot(x) | x=F(dt).dot(x) |
| P=F(dt).dot(P).dot(transpose(F(dt)))+Q(dt) | P=F(dt).dot(P).dot(transpose(F(dt)))+Q(dt) |
| return (x,P) | return (x,P) |
| ###Correction step### | ###Korrekturschritt### |
| def update (x,P,z,R): | def update (x,P,z,R): |
| y=z-H().dot(x) | y=z-H().dot(x) |
| S=H().dot(P).dot(transpose(H()))+R | S=H().dot(P).dot(transpose(H()))+R |
| K=P.dot(transpose(H))).dot(S++(-1)) | K=P.dot(transpose(H))).dot(S++(-1)) |
| x=x+K\*y | x=x+K\*y |
| P=(identity(2)-outer(K,H())).dot(P) | P=(identity(2)-outer(K,H())).dot(P) |
| return (x,P) | return (x,P) |

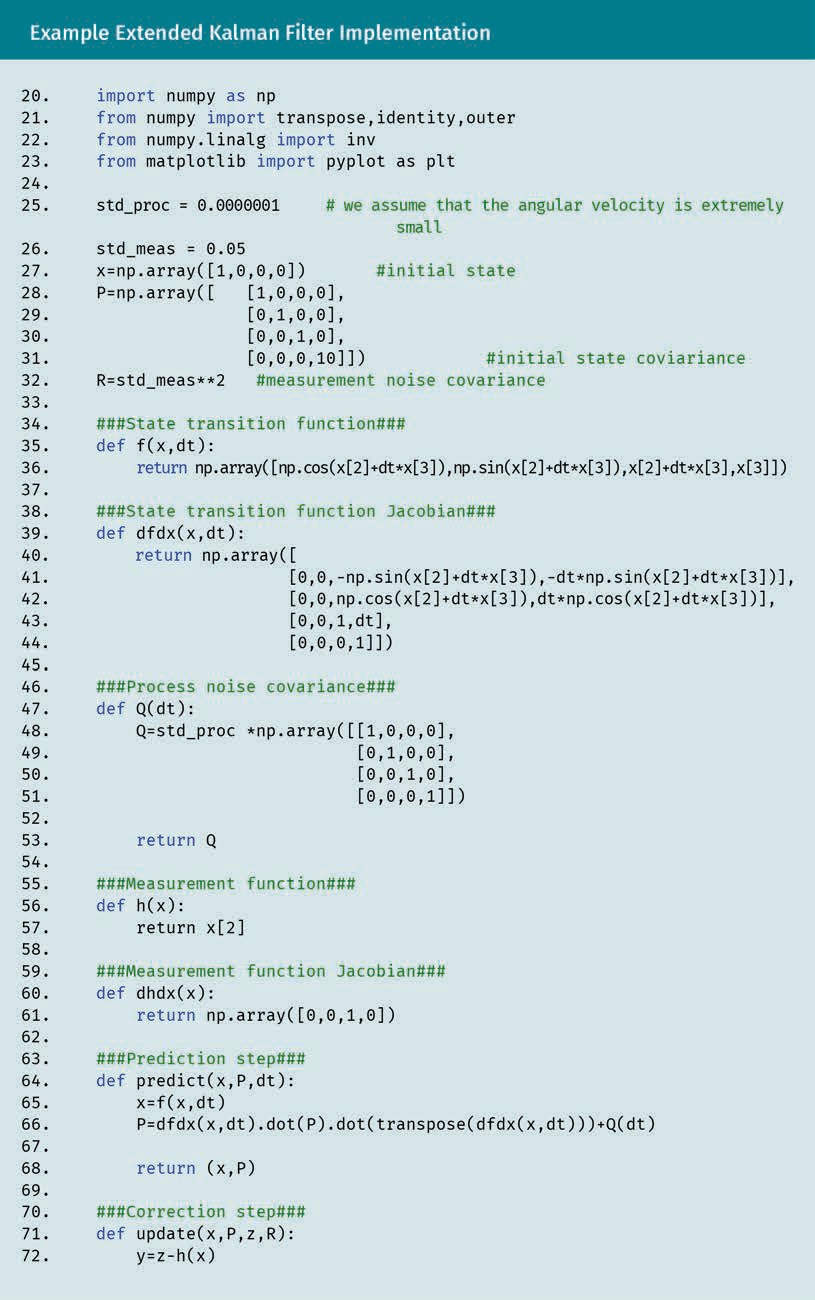
**Left: Noise Measurements and Filtered Signals, Right: Histogram for**

**Noisy Measurements and Filtered Signals**

**[Links: Rauschmessungen und gefilterte Signale. Rechts: Histogramm für verrauschte Messungen und gefilterte Signale]**

|  |  |
| --- | --- |
| Velocity [m/s] in x-direction | Geschwindigkeit [m/s] in x-Richtung |
| Velocity [m/s] in y-direction | Geschwindigkeit [m/s] in y-Richtung |
| Iteration | Iterationen |
| Frequency | Frequenz |

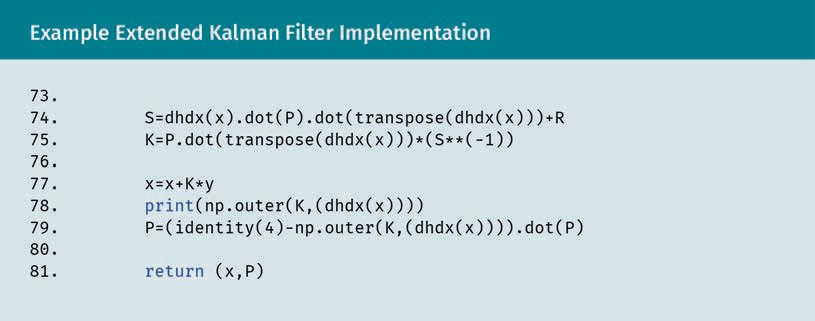
**Example Extended Kalman Filter Implementation**

**[Beispiel für die Implementierung eines erweiterten Kalman-Filters]**

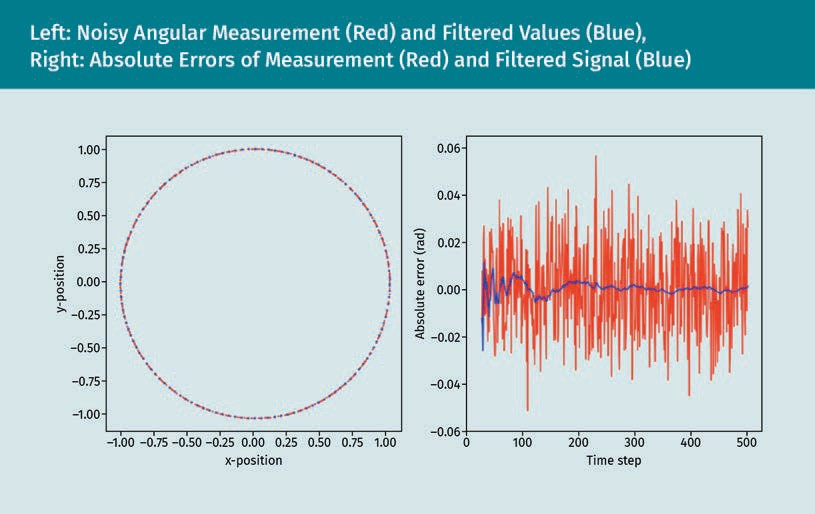
|  |  |
| --- | --- |
| **import** numpy as np | **import** numpy as np |
| **from** numpy **import** transpose,identity,outer | **from** numpy **import** transpose,identity,outer |
| **from** numpy.linalg **import** inv | **from** numpy.linalg **import** inv |
| **from** matplotlib **import** pyplot as plt | **from** matplotlib **import** pyplot as plt |
| std\_proc = 0.0000001 | std\_proc = 0.0000001 |
| # we assume that the angular  velocity is extremely small | # Wir gehen davon aus, dass die Winkelgeschwindigkeit  extrem gering ist |
| std\_meas = 0.05 | std\_meas = 0.05 |
| x=np.array([1,0,0,0])  #initial state | x=np.array([1,0,0,0])  #Anfangszustand |
| P=np.array([ [1,0,0,0], | P=np.array([ [1,0,0,0], |
| [0,1,0,0], | [0,1,0,0], |
| [0,0,1,0], | [0,0,1,0], |
| [0,0,0,10]])  #initial state coviariance | [0,0,0,10]])  #Kovarianz des Anfangszustands |
| R=std\_meas\*\*2  #measurement noise covariance | R=std\_meas\*\*2  #Kovarianz des Messrauschens |
| ###State transition function### | ###Zustandsübergangsfunktion ### |
| **def** f(x,dt): | **def** f(x,dt): |
| **return** np.array  ([np.cos(x[2]+dt\*x[3]),  np.sin(x[2]+dt\*x[3]),x[2]+dt\*x[3],x[3]]) | **return** np.array  ([np.cos(x[2]+dt\*x[3]),  np.sin(x[2]+dt\*x[3]),x[2]+dt\*x[3],x[3]]) |
| ###State transition function Jacobian### | ###Zustandsübergangsfunktion Jacobi-Matrix### |
| **def** dfdx(x,dt): | **def** dfdx(x,dt): |
| **return** np.array([ | **return** np.array([ |
| [0,0,-np.sin(x[2]+dt\*x[3]),  -dt\*np.sin(x[2]+dt\*x[3])], | [0,0,-np.sin(x[2]+dt\*x[3]),  -dt\*np.sin(x[2]+dt\*x[3])], |
| [0,0,np.cos(x[2]+dt\*x[3]),  dt\*np.cos(x[2]+dt\*x[3])], | [0,0,np.cos(x[2]+dt\*x[3]),  dt\*np.cos(x[2]+dt\*x[3])], |
| [0,0,1,dt], | [0,0,1,dt], |
| [0,0,0,1]]) | [0,0,0,1]]) |
| ###Process noise covariance### | ###Kovarianz des Prozessrauschens### |
| **def** Q(dt): | **def** Q(dt): |
| Q=std\_proc \*np.array([[1,0,0,0], | Q=std\_proc \*np.array([[1,0,0,0], |
| [0,1,0,0], | [0,1,0,0], |
| [0,0,1,0], | [0,0,1,0], |
| [0,0,0,1]]) | [0,0,0,1]]) |
| **return** Q | **return** Q |
| ###Measurement function### | ###Messfunktion### |
| **def** h(x): | **def** h(x): |
| **return** x[2] | **return** x[2] |
| ###Measurement function Jacobian### | ###Messfunktion, Jacobi-Matrix### |
| **def** dhdx(x): | **def** dhdx(x): |
| **return** np.array([0,0,1,0]) | **return** np.array([0,0,1,0]) |
| ###Prediction step### | ###Vorhersageschritt### |
| **def** predict(x,P,dt): | **def** predict(x,P,dt): |
| x=f(x,dt) | x=f(x,dt) |
| P=dfdx(x,dt).dot(P).dot  (transpose(dfdx(x,dt)))+Q(dt) | P=dfdx(x,dt).dot(P).dot  (transpose(dfdx(x,dt)))+Q(dt) |
| **return** (x,P) | **return** (x,P) |
| ###Correction step### | ###Korrekturschritt### |
| **def** update(x,P,z,R): | **def** update(x,P,z,R): |
| y=z-h(x) | y=z-h(x) |
| S=dhdx(x).dot(P).dot(transpose(dhdx(x)))+R | S=dhdx(x).dot(P).dot(transpose(dhdx(x)))+R |
| K=P.dot(transpose(dhdx(x)))\*(S\*\*(-1)) | K=P.dot(transpose(dhdx(x)))\*(S\*\*(-1)) |
| x=x+K\*y | x=x+K\*y |
| **print**(np.outer(K,(dhdx(x)))) | **print**(np.outer(K,(dhdx(x)))) |
| P=(identity(4)-np.outer(K,(dhdx(x)))).dot(P) | P=(identity(4)-np.outer(K,(dhdx(x)))).dot(P) |
| **return** (x,P) | **return** (x,P) |

**Example Extended Kalman Filter Implementation**

**[Beispiel für die Implementierung eines erweiterten Kalman-Filters]**

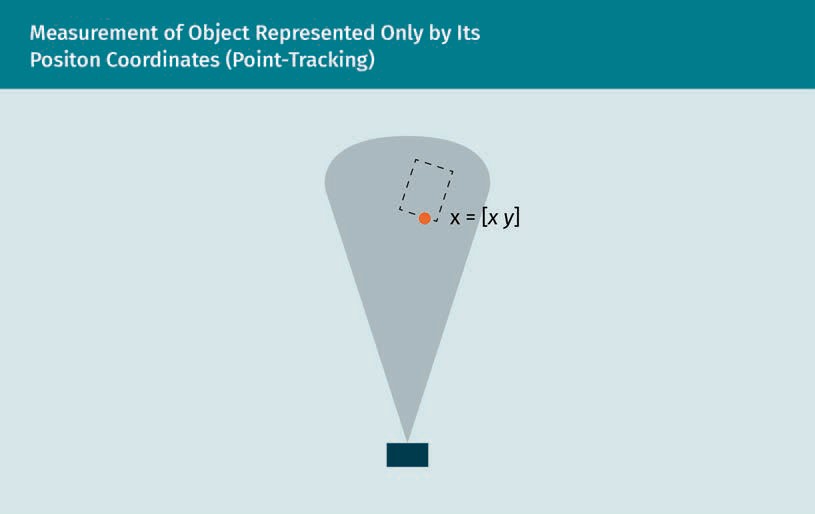


Le**ft: Noisy Angular Measurement (Red) and Filtered Values (Blue), Right: Absolute Errors of Measurement (Red) and Filtered Signal (Blue)**

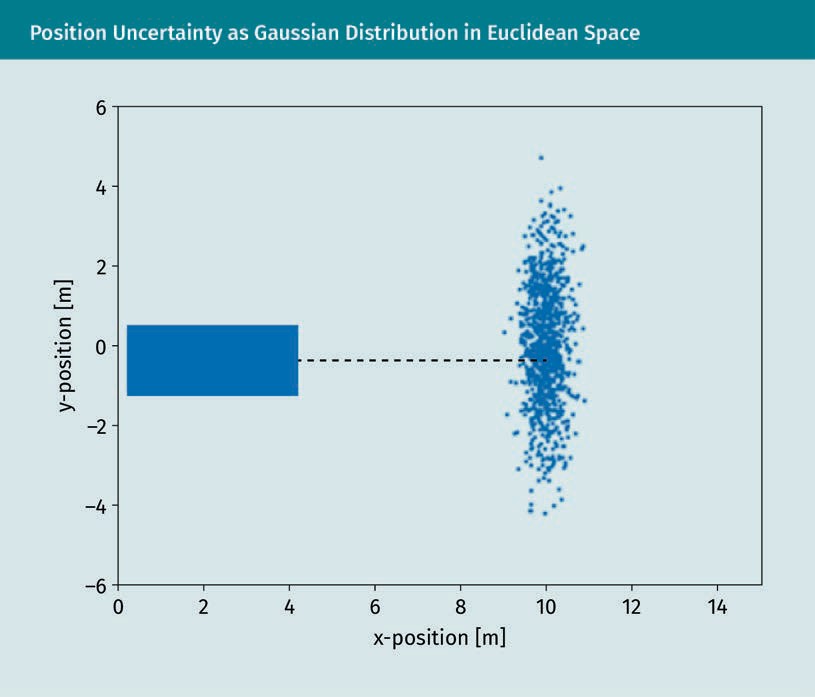
**[Links: Verrauschte Winkelmessung (rot) und gefilterte Werte (blau). Rechts: Absolute Messabweichungen (rot) und gefiltertes Signal (blau)]**

|  |  |
| --- | --- |
| y-position | y-Position |
| x-position | x-Position |
| Absolute error (rad) | Absoluter Fehler (rad) |
| Time step | Zeitschritt |

**Measurement of Object Represented Only by Its Position Coordinates (Point-Tracking)**

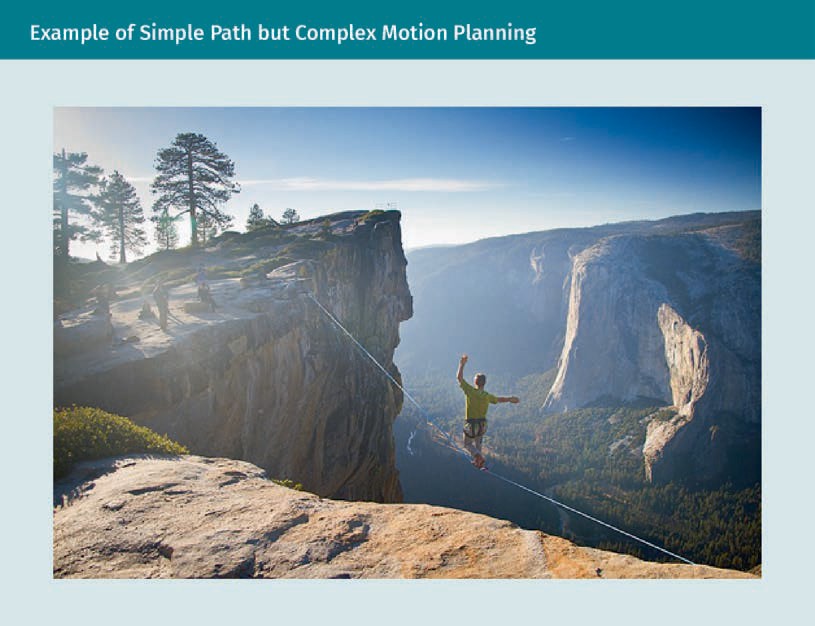
**[Messung eines Objekts, das nur durch seine Positionskoordinaten dargestellt wird (Punktverfolgung)]**

**Position Uncertainty as Gaussian Distribution in Euclidean Space**

**[Positionsunsicherheit als Gaußsche Verteilung im euklidischen Raum]**

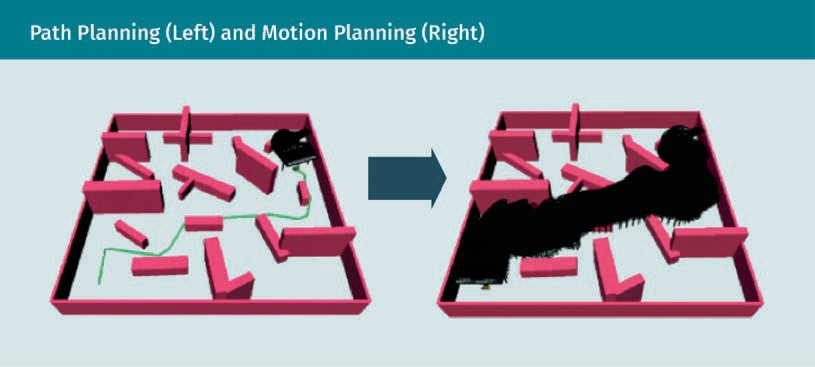
|  |  |
| --- | --- |
| y-position [m] | y-Position [m] |
| x-position [m] | x-Position [m] |

**Example of Simple Path but Complex Motion Planning**

**[Beispiel für einen einfachen Pfad aber komplexe Bewegungsplanung]**

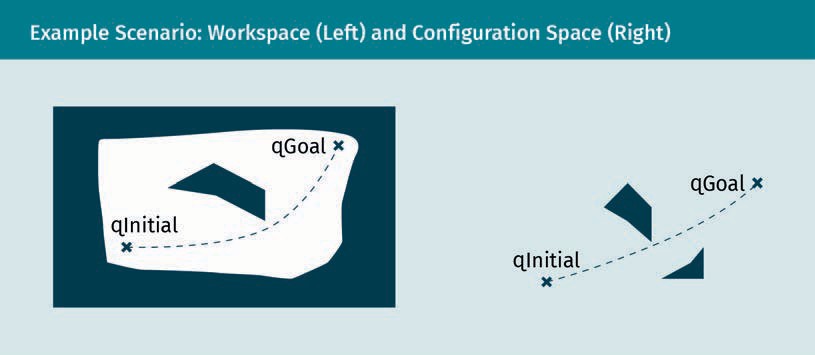
**Path Planning (Left) and Motion Planning (Right)**

**[Pfadplanung (links) und Bewegungsplanung (rechts)]**



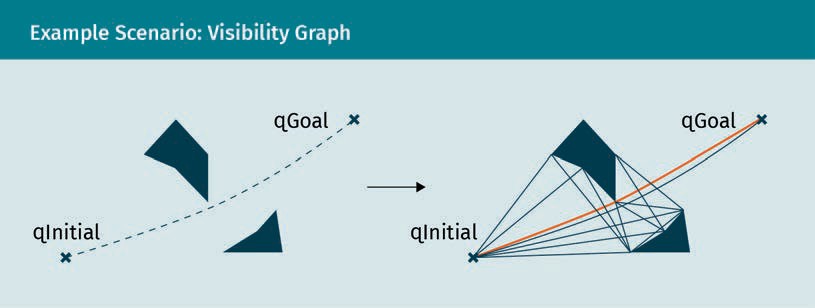
**Example Scenario: Workspace (Left) and Configuration Space (Right)**

**[Beispielszenario: Arbeitsraum (links) und Konfigurationsraum (rechts)]**

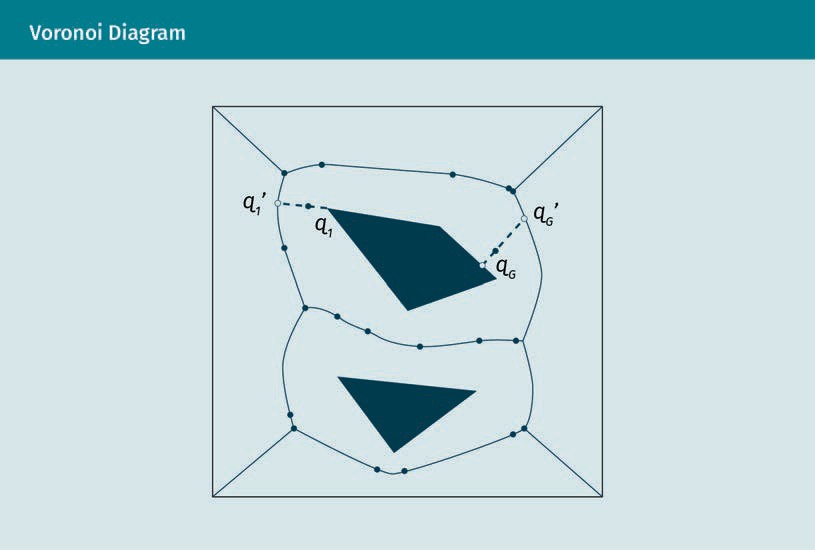


|  |  |
| --- | --- |
| qInitial | qInitial |
| qGoal | qGoal |

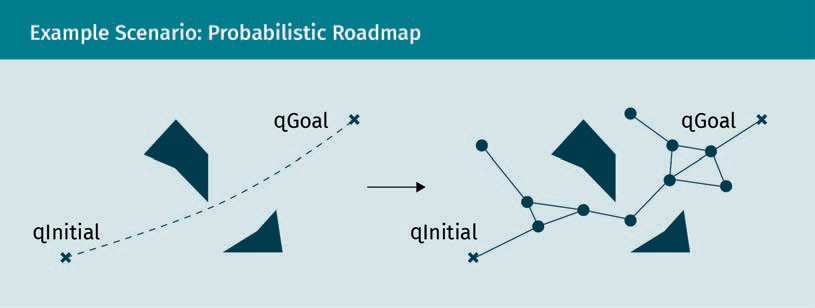
**Example Scenario: Visibility Graph**

**[Beispielszenario: Sichtbarkeitsdiagramm]**

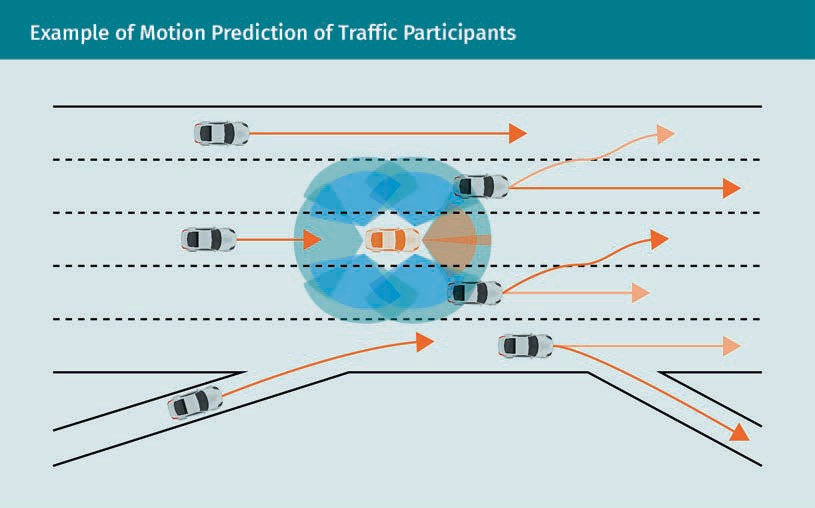
**[Voronoi-Diagramm]**



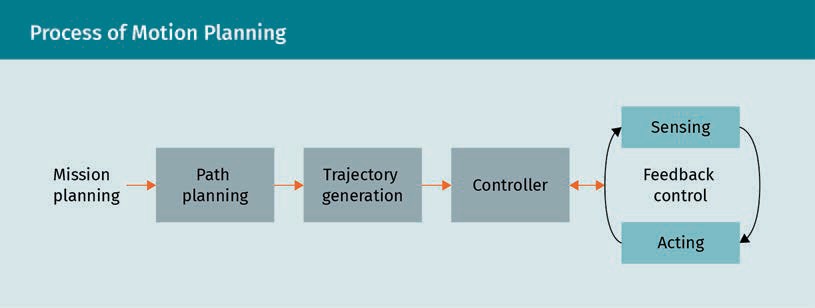
**Example Scenario: Probabilistic Roadmap**

**[Beispielszenario: Probabilistische Straßenkarte]**

**Example of Motion Prediction of Traffic Participants**

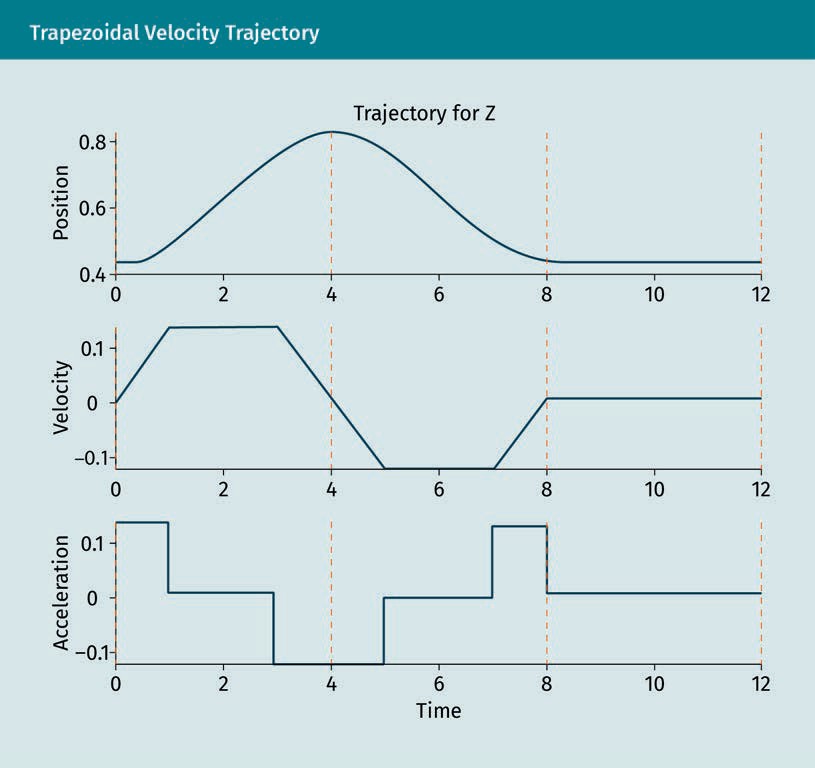
**[Beispiel für die Bewegungsvorhersage von Verkehrsteilnehmenden]**

**Process of Motion Planning**

**[Ablauf der Bewegungsplanung]**

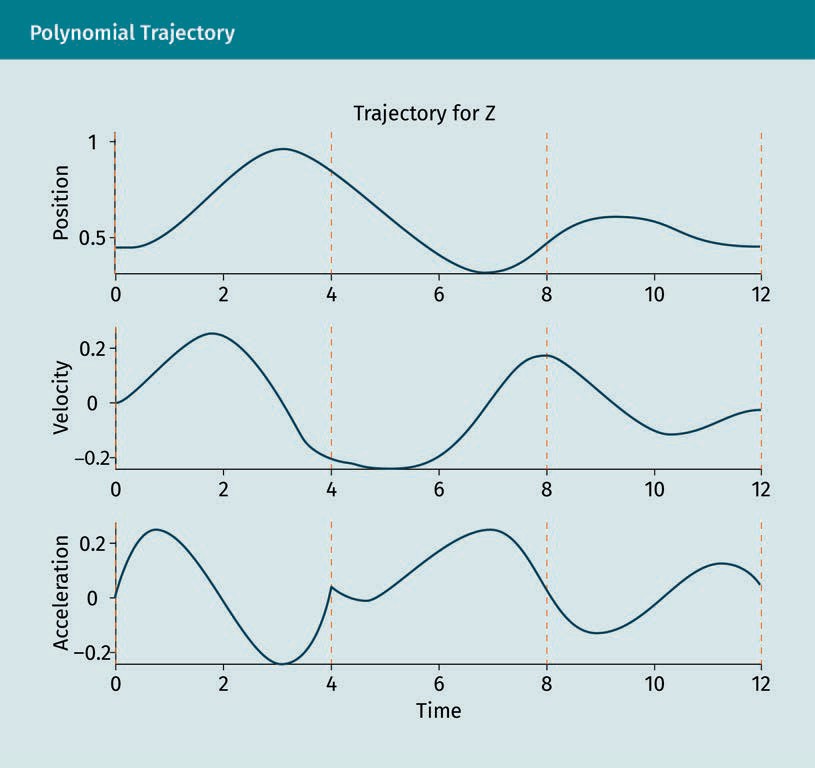
|  |  |
| --- | --- |
| Mission planning | Auftragsplanung |
| Path planning | Pfadplanung |
| Trajectory generation | Bahnkurvenermittlung |
| Controller | Kontrolle |
| Sensing | Sensorik |
| Feedback Control | Regelkreis |
| Acting | Aktionen |

**Trapezoidal Velocity Trajectory**

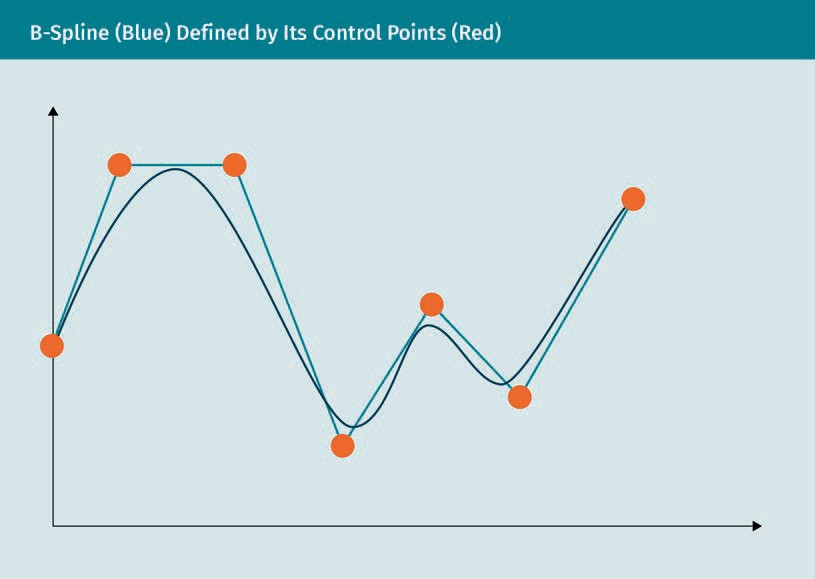
**[Bahnkurve mit trapezförmigem Geschwindigkeitsverlauf]**

|  |  |
| --- | --- |
| Trajectory for Z | Bahnkurve für Z |
| Acceleration | Beschleunigung |
| Velocity | Geschwindigkeit |
| Position | Position (Weg) |
| Time | Zeit |

**Polynomial Trajectory**

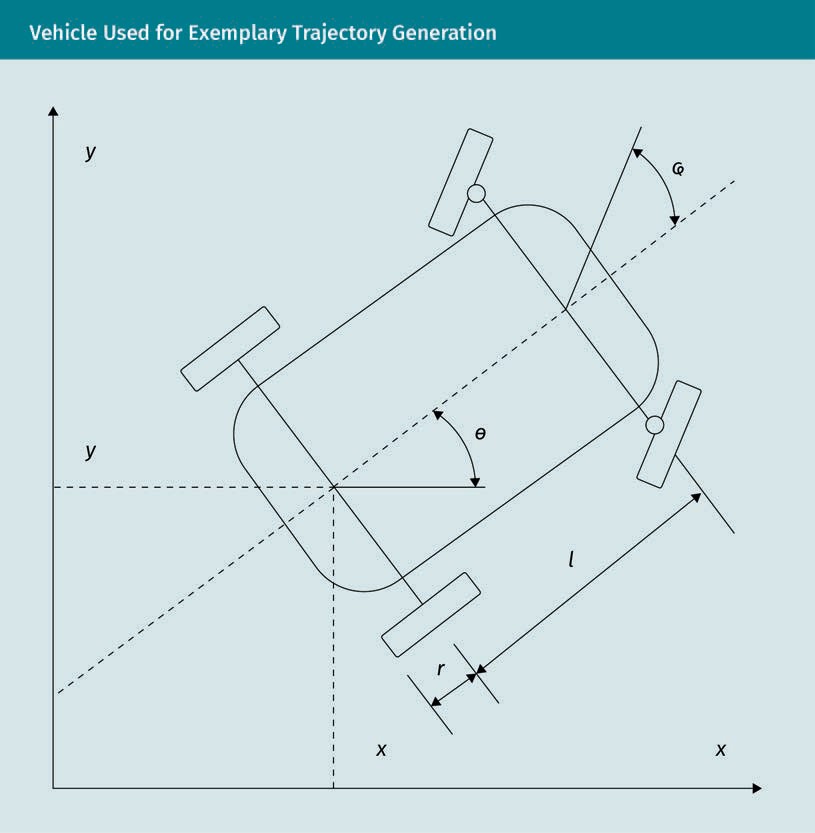
**[Polynomiale Bahnkurve]**

**B-Spline (Blue) Defined by Its Control Points (Red)**

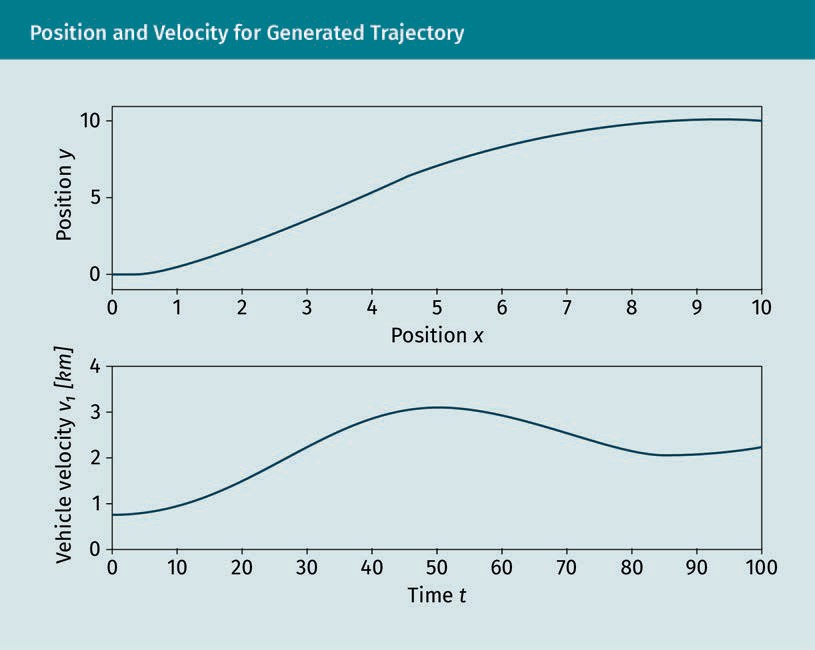
**[Durch seine Kontrollpunkte (rot) definierter B-Spline (blau)]**

**Vehicle Used for Exemplary Trajectory Generation**

**[Fahrzeug mit beispielhafter Bahnkurvenermittlung]**



**Position and Velocity for Generated Trajectory**

**[Position und Geschwindigkeit für eine ermittelte Bahnkurve]**

|  |  |
| --- | --- |
| Position *y* | Position *y* |
| Vehicle velocity v1 *[km]* | Fahrzeuggeschwindigkeit v1 *[km/h]* |
| Position *x* | Position *x* |
| Time *t* | Zeit *t* |