WOR-ROBOTS-DICTAAT_

Joost Kraaijeveld, Chris van Uffelen en Richard Holleman
1 oktober 2024

HAN_UNIVERSITY
OF APPLIED SCIENCES

Dit is het dictaat voor de course “Robots” van het semester “World of Robots” van het profiel Embedded Software Development (ESD) van AIM.

INHOUDSOPGAVE

LIJST VAN TABELLEN

LIJST VAN FIGUREN

1. INLEIDING

In dit document worden voor de meeste begrippen “informele” en “working knowledge” definities gegeven. Het gaat om het begrip en niet de exacte definitie. Dit is genoeg voor de toets en voor het dagelijks gebruik in WoR. Mocht je ooit de behoefte hebben om met echte wiskundigen over deze materie te praten dan raad ik je aan de meer formele definities uit de diverse Wikipedia-bronnen of de in die Wikipedia vermelde wetenschappelijke literatuur te gebruiken. Bovendien wordt er van uitgegaan dat je de lessen gevolgd hebt: dit stuk is naslagwerk, geen leerboek.

Her en der in het dictaat staan links naar met name Wikipedia-pagina’s. Soms staan ze er als achtergrond van het verhaal (vooral als het historische figuren zijn) en zijn ze ter vermaak. Maar meestal gaat het naar aanvullende uitleg van de stof. Als je het ondanks de lessen of dictaat niet begrijpt dan kan je daar beginnen met verder zoeken. De Wikipedia-lemma’s zijn bovedien vaak diepgravender en uitgebreider dan de korte uitleg in dit dictaat. Je kan de links dus ook gebruiken ter verdieping.

Meestal wordt er gelinkt naar de Engelstalige Wikipedia-lemma’s. Die zijn veelal uitgebreider, volledieger en daardoor vaak duidelijker. De meeste Wikipedia-pagina’s bevatten naast het lemma met de referenties zelf ook links naar relevante andere pagina’s of documenten (zie ook “Further reading” op de Wikipedia-pagina) of links naar externe websites. Deze kan je allemaal gebruiken bij de zoektocht naar meer kennis.

Soms wordt er naar een Nederlandstalig lemma gelinkt. In dat geval vond ik dat het Nederlandse lemma zoveel duidelijker was dat het mij het handigst leek dat je daar begint te lezen. Ga daarna eventueel verder met de Engelstalige pagina. YMMV. Veranderen van taal is niet heel ingewikkeld op Wikipedia.

2. DIFFERENTIËREN EN AFGELEIDEN

Zie ook Wikipedia:

- Afgeleide (Nederlands): Afgeleide
- Afgeleide (Engels): Derivative
- Functie (Nederlands): Wiskundige functie
- Functie (Engels): Mathematical function

2.1. FUNCTIES

Een functie is een relatie die het verband legt tussen één of meer onafhankelijke variabelen en één of meer afhankelijke variabelen. De onafhankelijke variabelen vormen de input van de functie en worden door een functievoorschrift omgezet/vertaald naar de afhankelijke variabelen. De onafhankelijke variabelen worden ook wel *het domein* van een functie genoemd, de afhankelijke variabelen *het codomein* van de functie. *Het bereik* van een functie bestaat uit die waarden uit het codomein die daadwerkelijk de uitkomst van de functie kunnen zijn. Praktisch gezien komen het codomein en het bereik van de meeste continue functies[1] met elkaar overeen.

Een ander manier om over functies te praten is in termen van verzamelingen. In dat geval worden het domein, codomein en bereik gezien als verzamelingen. Een functie wordt in dat geval gedefineerd door de relatie van een element in het domein en het bereik. Het koppelen van een element uit het domein aan het bereik wordt een afbeelding genoemd. Het element uit het domein heet dan een origineel en het element uit het bereik een beeld.

Een functie heet bijectief als ieder element uit het domein een uniek element in het bereik en ieder element uit het bereik een uniek element in het domein. Het bereik en codomein zijn gelijk.

Een functie heet surjectief als ieder element uit het bereik een afbeelding heeft maar sommige element hebben hetzelfde beeld. Het bereik en het codomein zijn gelijk.

Een functie heet injectief als ieder element uit het domein een uniek element element in het bereik heeft maar als niet alle elementen uit het codomein hebben een origineel. Het bereik en het codomein zijn dus *niet* gelijk: het codomein is groter.

Een functie van één onafhankelijke variabele X en één afhankelijke variabele Y wordt ook wel op de volgende twee manieren beschreven:

- $F : X \to Y$
- $y = f(x)$

Een functie van meerdere onafhankelijke variabelen en meerdere afhankelijke variabelen wordt ook wel op de volgende manier beschreven:

- $F : \mathbb{X}^m \to \mathbb{Y}^n$

Hierbij zijn X en Y meestal getallen uit de getallenverzamelingen $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ of $\mathbb{C}$. m en n zijn meestal getallen uit $\mathbb{N} > 0$ en geven de *dimensies* van de functie aan. De functie $F : \mathbb{R}^3 \to \mathbb{R}^2$ is een functie die 3 dimensies vertaalt naar 2 dimensies. Een praktisch voorbeeld hiervan is een functie die de 3 hoeken van een 3DOF–robot vertaalt naar de x– en y–coördinaten van het eindpunt.

1 De link gaat naar een formele definitie van een continue functie op Wikipedia. Een informele definitie van een continue functie is een functie die je zou kunnen tekenen door je pen op een papier te zetten en een lijn te tekenen zonder je pen van het papier te halen en zonder hoeken te gebruiken. Helaas gaat deze beschrijving aleen maar op voor 2D- of (met wat fantasie) 3D-tekeningen. Het idee blijft voor meer-dimensionale functies evenwel hetzelfde.

2.2. INVERSE FUNCTIES

Een inverse functie is een relatie die het verband tussen tussen één of meer afhankelijke variabelen en één of meer onafhankelijke variabelen, het omgekeerde (inverse) van een gewone functie.

Voor lineaire functies is de inverse functie makkelijk te bepalen:
$y = ax + b \Leftrightarrow$
$y - b = ax \Leftrightarrow$
$\frac{y-b}{a} = x \Leftrightarrow$
$x = \frac{y-b}{a} \Leftrightarrow$
$x = \frac{1}{a}y - \frac{b}{a}$.
Voor hogere–ordefuncties of functies met *sin* en *cos* is het vaak ingewikkelder om de inverse functie te bepalen.

2.3. AFGELEIDEN VAN FUNCTIES

De afgeleide van een functie is een maat voor de verandering van de output (afhankelijke variabelen) als de input (onafhankelijke variabelen) een (oneindig) klein beetje wijzingen.

De afgeleide van een functie met één variabele is gedefinieerd als $\lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$. Hierbij is $f(x+\Delta x) - f(x)$ de verandering in de afhankelijke variabele en Δx de (oneindig) kleine verandering van de onafhankelijke variabelen. In andere woorden, de afgeleide van een functie is de wijziging in de output/afhankelijke variabele y $(f(x+\Delta x) - f(x))$ per kleine verandering van de input/onafhankelijke variabele x (Δx).

De afgeleide van een functie wordt onder andere op de volgende wijzen genoteerd:

- $f'(x)$
- $\frac{dy}{dx}$
- $\frac{\Delta y}{\Delta x}$
- $\frac{df}{dx}$
- $\frac{\Delta f}{\Delta x}$

Voor het bepalen van afgeleiden van diverse functies zijn rekenregels opgesteld. Deze rekenregels zijn makkelijk te vinden op internet. De rekenregels voor functies van de volgende types moet je kennen:

- $y = b$, waarbij b een constante is, afgeleide: 0
- $y = ax + b$, afgeleide: a
- $y = ax^n$, afgeleide: nax^{n-1}
- $y = \sin(x)$, afgeleide: $\cos(x)$
- $y = \cos(x)$, afgeleide: $-\sin(x)$

Mocht je willen weten waarom de rekenregels zijn zoals ze zijn kan je hier kijken

Ook moet je de volgende regels met betrekking tot complexere afgeleiden kennen en kunnen gebruiken:

- lineariteit: $(af(x))' = af'(x)$
- somregel: $(f(x) + g(x))' = f'(x) + g'(x)$
- kettingregel: $f(g(x))' = f'(g(x))g'(x)$
- de productregel en quotiëntregel hoef je voor WoR niet te kennen of uit je hoofd toe te kunnen passen.

Er zijn nog twee regels maar die hoef je voor WoR niet te kennen:

- productregel: $(af(x)g(x))' = af'(x)g(x) + af(x)g'(x)$
- quotiëntregel: $(f'(x) = \frac{g(x)}{h(x)}) = \frac{g'(x)h(x) + g(x)h'(x)}{h(x)^2}$

2.4. GEBRUIK VAN AFGELEIDEN

Uit de definitie van de afgeleide volgt dat afgeleiden gebruikt kunnen worden om een voorspelling te doen over de verandering van de afhankelijke variabelen bij een kleine verandering van de onafhanke-

lijke variabelen. Let hierbij op dat dit in het algemeen slechts mogelijk is voor kleine veranderingen van de onafhankelijke variabelen.

Om een voorspelling te doen moet je, gegeven een functie $F : x \to y$, de volgende stappen uitvoeren:

1. Reken met behulp van de functie $F : x \to y$ voor een gekozen x_1 de bijbehorende y_1 uit.
2. Bepaal de afgeleide $\frac{dy}{dx}$ van de functie $F : x \to y$.
3. Gebruik een gekozen Δx en de afgeleide $\frac{dy}{dx}$ om de Δy uit te rekenen: $\Delta y = \frac{dy}{dx}\Delta x$.
4. Bereken de voorspelde y met behulp van $y_{voorspeld} = y_1 + \Delta y$.

Voorbeeld 1. Het gebruik van de afgeleide bij lineaire functies

De afgeleide van de lineaire functie $y = 42x$ is voor een $\Delta x = 2$ een perfecte voorspeller voor Δy.

Bewijs:

Op grond van de diverse notatievormen van de afgeleide geldt: $\frac{df}{dx} = \frac{dy}{dx} \approx \frac{\Delta f}{\Delta x}$

Voor kleine Δx geldt dan: $\frac{df}{dx} = \frac{dy}{dx} = \frac{\Delta f}{\Delta x} \Rightarrow \Delta y = \frac{dy}{dx}\Delta x$

Berekening:

1. Reken met behulp van de functie $F : x \to y$ voor een gekozen x_1 de bijbehorende y_1 uit:
   Stel $x_1 = 1$, dan $y_1 = 42 \times x_1 = 42$
2. Bepaal de afgeleide $\frac{dy}{dx}$ van de functie $F : x \to y$:
   De afgeleide $\frac{dy}{dx}$ van $y = 42x$ is 42.
3. Gebruik een gekozen Δx en de afgeleide $\frac{dy}{dx}$ om de Δy uit te rekenen: $\Delta y = \frac{dy}{dx}\Delta x$:
   Stel $\Delta x = 2$ dan $\Delta y = \frac{dy}{dx}\Delta x = 42 \times 2 = 84$
4. Bereken de voorspelde y met behulp van $y_{voorspeld} = y_1 + \Delta y$:
   Dus $y_{voorspeld} = y_1 + \Delta y = 42 + 84 = 126$

Controle met behulp van de functie zelf:
$x_2 = x_1 + \Delta x = 1 + 2 = 3$ dus
$y_2 = 42x_2 = 42 \times 3 = 126$

Conclusie:
$y_{voorspeld} = y_2$, de afgeleide is een perfecte voorspeller.

Voorbeeld 2. Het gebruik van de afgeleide bij kwadratische functies

De afgeleide van de kwadratische functie $y = 3x^2$ is voor een $\Delta x = 1$ een acceptabele[2] voorspeller voor Δy.

Bewijs:

Op grond van de diverse notatievormen van de afgeleide geldt: $\frac{df}{dx} = \frac{dy}{dx} \approx \frac{\Delta f}{\Delta x}$

Voor kleine Δx geldt dan: $\frac{df}{dx} = \frac{dy}{dx} = \frac{\Delta f}{\Delta x} \Rightarrow \Delta y = \frac{dy}{dx}\Delta x$

Berekening:

1. Reken voor een willekeurige x_1 de bijbehorende y_1 uit:
   Stel $x_1 = 1$, dan $y_1 = 3 \times x_1^2 = 3 \times 1^2 = 3$
2. Bepaal de afgeleide $\frac{dy}{dx}$ van de functie $F : x \to y$:
   De afgeleide $\frac{dy}{dx}$ van $y = 3x^2$ is $6x$, op $x = 1 \Rightarrow \frac{dy}{dx} = 6 * 1 = 6$
3. Gebruik een gekozen Δx en de afgeleide $\frac{dy}{dx}$ om de Δy uit te rekenen: $\Delta y = \frac{dy}{dx}\Delta x$:
   Stel $\Delta x = 1$ dan $df = dy = \frac{\Delta f}{\Delta x}dx = 6 * 1 = 6$
4. Bereken de voorspelde y met behulp van $y_{voorspeld} = y_1 + \Delta y$:
   Dus $y_{voorspeld} = y_1 + dy = 3 + 6 = 9$

Controle met behulp van de functie zelf:
$x_2 = x_1 + \Delta x = 1 + 1 = 2$ dus
$y_2 = 3x^2 = 3 \times 1^2 = 3$

2 Acceptabel is in dit verband afhankelijk van de omstandigheden en de doelstelling van de berekening. Dit voorbeeld is vooral gericht op het uitrekenen van de gegevens, niet op het daadwerkelijk acceptabel zijn...

Conclusie:
$y_{voorspeld} \approx y_2$, de afgeleide is een acceptabele voorspeller.

Voor een kleinere Δx zijn de voorspellingen gelukkig beter:

- $\Delta x = \frac{1}{2}$: $y_{voorspeld} - y_2 = 6 \cdot \Delta x - 3 \cdot x_2^2 = 6 \cdot \frac{1}{2} - 3 \cdot \frac{1}{2}^2 = \frac{786}{256} - \frac{192}{256} = \frac{576}{256}$
- $\Delta x = \frac{1}{4}$: $y_{voorspeld} - y_2 = 6 \cdot \Delta x - 3 \cdot x_2^2 = 6 \cdot \frac{1}{4} - 3 \cdot \frac{1}{4}^2 = \frac{384}{256} - \frac{48}{256} = \frac{336}{256}$
- $\Delta x = \frac{1}{8}$: $y_{voorspeld} - y_2 = 6 \cdot \Delta x - 3 \cdot x_2^2 = 6 \cdot \frac{1}{8} - 3 \cdot \frac{1}{8}^2 = \frac{192}{256} - \frac{12}{256} = \frac{180}{256}$
- $\Delta x = \frac{1}{16}$: $y_{voorspeld} - y_2 = 6 \cdot \Delta x - 3 \cdot x_2^2 = 6 \cdot \frac{1}{16} - 3 \cdot \frac{1}{16}^2 = \frac{96}{256} - \frac{3}{256} = \frac{93}{256}$

Het verschil tussen de feitelijke uitgerekende waarde en de voorspelde waarde is kleiner voor een kleinere Δx. Dit klopt met het feit dat de afgeleide is gedefinieerd met $\lim_{\Delta x \to 0}$

2.5. AFGELEIDEN VAN INVERSE FUNCTIES

Op dezelfde manier als bij de gewone functie is de afgeleide van een inverse functie een maat voor de verandering van de output (de onafhankelijke variabelen) als de input (afhankelijke variabelen) een (oneindig) klein beetje wijzingen. Bij een inverse functie geeft de afgeleide dus eigenlijk het antwoord op de vraag hoeveel je de onafhankelijke variabelen moet wijzingen om een bepaalde wijziging van de afhankelijke variabelen te krijgen.

De manier om de afgeleide van de inverse functie te bepalen is identiek aan die van de gewone afgeleide met dien verstande dat de x en y in het algemeen omgekeerd zijn.

Van de lineaire functie $y = ax + b$ is $x = \frac{1}{a}y - \frac{b}{a}$ de inverse functie. Hiervan is de afgeleide $\frac{1}{a}$.

3. MATRICES

Zie ook Wikipedia:

- Matrix (Nederlands)
- Matrix (Engels

Een matrix is een twee-dimensionale array met getallen, de elementen van de matrix. Traditiegetrouw worden de rijen met de letter m en de kolommen met de letter n aangeduid. Bij het specificeren van een matrix wordt eerst het aantal rijen en dan het aantal kolommen opgegeven. Het element op de i-de rij en j-de kolom van matrix A wordt aangegeven met A_{ij}. Een 4×3-matrix A is dus een matrix met 4 rijen en 3 kolommen en A_{23} is het element dat op de tweede rij en in de derde kolom staat. Als m en n aan elkaar gelijk zijn dan wordt de matrix een *vierkante* matrix genoemd. Een matrix met m rijen en 1 kolom wordt een kolom-vector (column vector) genoemd en een matrix met 1 rij en n kolommen een rij-vector (row vector).

Al naar gelang de vorm en inhoud van een matrix hebben sommige matrices een specifieke naam. De volgende matrices zijn hiervan de belangrijkste:

- Identiteitsmatrix
- Driehoeksmatrix

De uitleg bij de wikipedia-lemma's is voldoende. Bij de identiteitsmatrix en de getransponeerde matrix worden ook een aantal rekenregels/eigenschappen genoemd. Die hoef je niet allemaal uit je hoofd te kennen. Een aantal daarvan zullen later in het dictaat gebruikt gaan worden. Die moet je dan dus wel kennen.

3.1. REKENKUNDIGE OPERATIES

Met matrices kunnen de volgende rekenkundige operaties uitgevoerd worden:

- optellen (en aftrekken),
- vermenigvuldigen met (en delen door) een scalar,
- vermenigvuldigen met een andere matrix.

3.1.1. OPTELLEN

Optellen van matrices kan alleen als beide matrices dezelfde grootte $m \times n$ hebben. In dat geval worden de matrices elementgewijs opgeteld: de som van de $m \times n$ matrices A en B levert de $m \times n$ matrix C op waarvoor geldt $C_{ij} = A_{ij} + B_{ij}$. Dus:

$$C = A + B = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2n}+b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \dots & a_{mn}+b_{mn} \end{bmatrix}$$

Een praktisch voorbeeldje:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1+1 & 2+2 & 3+3 \\ 4+4 & 5+5 & 6+6 \\ 7+7 & 8+8 & 9+9 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$$

Aftrekken van twee matrices wordt analoog uitgevoerd: gewoon de $+$ vervangen door $-$.

3.1.2. VERMENIGVULDIGEN MET EEN SCALAR

Het vermenigvuldigen van een matrix met een scalar gebeurt door het elementgewijs vermenigvuldigen met die scalar: het product van de $m \times n$ matrix A en de scalar b levert de $m \times n$ matrix C op waarvoor

geldt $C_{ij} = A_{ij} \times b$. Dus:

$$C = A \times b = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \times b = \begin{bmatrix} a_{11} \times b & a_{12} \times b & \dots & a_{1n} \times b \\ a_{21} \times b & a_{22} \times b & \dots & a_{2n} \times b \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} \times b & a_{m2} \times b & \dots & a_{mn} \times b \end{bmatrix}$$

Een praktisch voorbeeldje:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times 2 = \begin{bmatrix} 1 \times 2 & 2 \times 2 & 3 \times 2 \\ 4 \times 2 & 5 \times 2 & 6 \times 2 \\ 7 \times 2 & 8 \times 2 & 9 \times 2 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$$

Delen door een scalar wordt analoog uitgevoerd: : gewoon de $\times$ vervangen door /.

3.1.3. VERMENIGVULDIGEN MET EEN MATRIX

Twee matrices kunnen met elkaar vermenigvuldigd worden als het aantal kolommen van de linker operand van de vermenigvuldiging gelijk is aan het aantal rijen van de rechter operand: i.e. de $m \times p$-matrix A en $p \times n$-matrix B kunnen met elkaar vermenigvuldigd worden. Het resultaat wordt de $m \times n$-matrix C waarvoor geldt $C_{mn} = \sum_{i=1}^{p} A_{mi} B_{in}$. Oftewel, de cellen van de resultaatmatrix worden gevuld door de optelling van de elementgewijze vermenigvuldiging van een rij uit matrix A en kolom uit matrix B. Hierbij wordt cel $(1,1)$ van de resultaatmatrix gevuld met behulp van rij 1 uit matrix A en kolom 1van matrix B, cel $(1,2)$ met behulp van rij 1 uit matrix A en kolom 2van matrix B, $(2,1)$ met behulp van rij 2 uit matrix A en kolom 1van matrix B enzovoorts. Dus:

$$C = A \times B = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mp} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{p1} & b_{p2} & \dots & b_{pn} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{p} A_{1i}B_{i1} & \sum_{i=1}^{p} A_{1i}B_{i2} & \dots & \sum_{i=1}^{p} A_{1i}B_{in} \\ \sum_{i=1}^{p} A_{2i}B_{i1} & \sum_{i=1}^{p} A_{2i}B_{i2} & \dots & \sum_{i=1}^{p} A_{2i}B_{in} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{p} A_{mi}B_{i1} & \sum_{i=1}^{p} A_{mi}B_{i2} & \dots & \sum_{i=1}^{p} A_{mi}B_{in} \end{bmatrix}$$

Een praktisch voorbeeldje van een vermenigvuldigen van twee matrices:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1\times1+2\times4+3\times7 & 1\times2+2\times5+3\times8 & 1\times3+2\times6+3\times9 \\ 4\times1+5\times4+6\times7 & 4\times2+5\times5+6\times8 & 4\times3+5\times6+5\times9 \\ 7\times1+8\times4+9\times7 & 7\times2+8\times5+9\times8 & 7\times3+8\times6+9\times9 \end{bmatrix} = \begin{bmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \\ 102 & 126 & 150 \end{bmatrix}$$

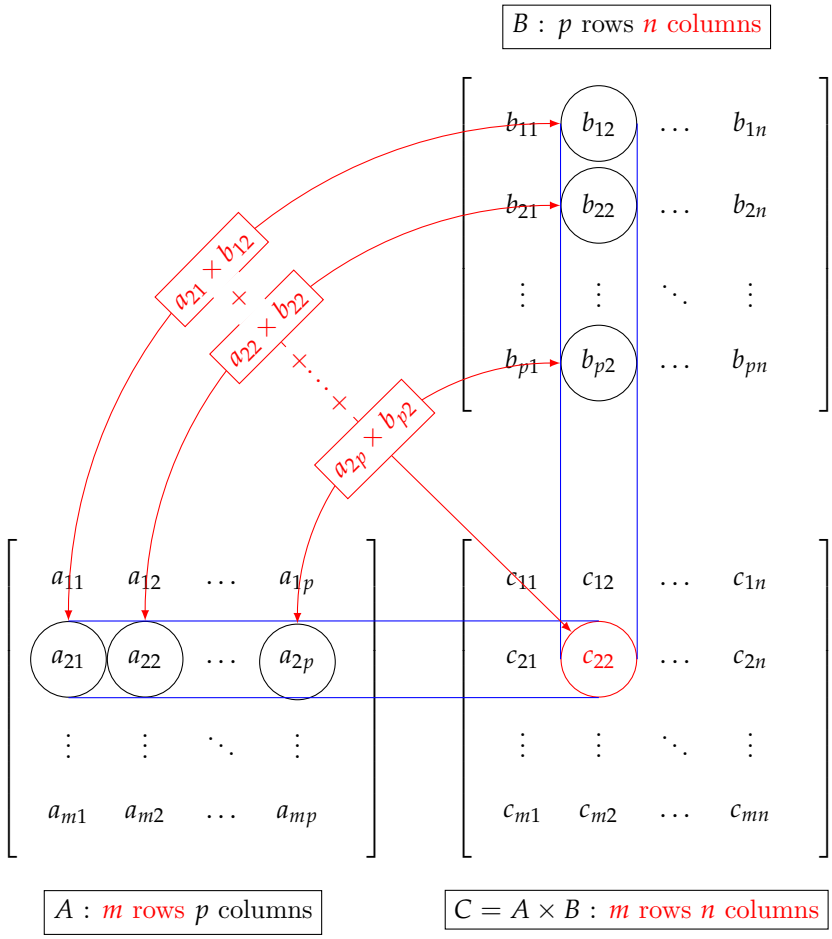
Een andere, en wellicht wat inzichtelijkere, uitleg hoe de waarde van een element in de matrix C tot stand komt staat in figuur 3.1. Hierbij wordt het element C_{22} uitgerekend door element 1 uit rij 2 te vermenigvuldigen met elementen 1 uit kolom 2, element 2 uit rij 2 te vermenigvuldigen met elementen 2 uit kolom 2 enzovoorts, en tot slot worden de resultaten van die vermenigvuldigingen bij elkaar opgeteld. Alle cellen van C worden uitgerekend door de rij en de kolom die samen die cel als snijpunt hebben.

In tegenstelling tot het vermenigvuldigen van gewone getallen is het het vermenigvuldigen van matrices niet commutatief: gegeven de matrices A en B is in het algemeen $AB \neq BA$.

Zo is $\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \times \begin{bmatrix} 4 & 2 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 13 & 5 \\ 20 & 8 \end{bmatrix}$ terwijl $\begin{bmatrix} 4 & 2 \\ 3 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 8 & 20 \\ 5 & 13 \end{bmatrix}$

3.1.4. DELEN DOOR EEN MATRIX

Matrices kunnen niet door elkaar gedeeld worden. Maar uit het gewone rekenen weten we dat delen hetzelfde is als vermenigvuldigen met het omgekeerde (inverse) en dat het vermenigvuldigen van een getal met zijn inverse 1 oplevert: $2 \cdot 2^{-1} = 2 \cdot \frac{1}{2} = 1$. Nu blijkt het dat het voor vierkante matrices (met wat mitsen en maren) mogelijk is om de inverse van die matrices te bepalen. Die matrices waarvoor een inverse matrix bestaat kunnen dus optreden als deler van een andere matrix. Uiteraard kan dit alleen als de daaruit resulterende twee matrices met elkaar te vermenigvuldigen zijn. Voor het inverteren van ma-



Figuur 3.1.: Visuele matrix-vermenigvuldiging

trices: zie paragraaf 3.2.2.

3.2. ANDERE OPERATIES

3.2.1. TRANSPONEREN

Het transponeren van een matrix is het verwisselen van de rijen en kolommen. De getransponeerde $m \times n$ matrix A is de $n \times m$ matrix A^T, waarbij de T uiteraard staat voor "transpose". Een voorbeeld van een getransoneerde matrix:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Als een getransponeerde matrix in een berekening voorkomt dan wordt eerst de transpose gedaan voordat de berekening wordt uitgevoerd. Voor meer uitleg over getransponeerde matrices zie het lemma op Wikipedia: Getransponeerde matrix.

3.2.2. INVERTEREN

De inverse van een matrix A is de matrix B waarvoor geldt dat $AB = BA = I$. Hierbij zijn A en B elkaars inverse en is I de identiteitsmatrix. De identiteitsmatrix functioneert hier als het getal 1, maar dan in matrixvorm.

3.2.2.1. VIERKANTE MATRICES

Voor een vierkante matrix kan, als de determinant van de matrix niet gelijk is aan 0[1], een inverse matrix bepaald worden (zie Wikipedia, Wolfram). Voor de berekening van een inverse matrix maken we onderscheid tussen kleine en grote vierkante matrices.

3.2.2.2. KLEINE VIERKANTE MATRICES

Als de matrix klein genoeg is (tot en met ongeveer 4×4) dan zijn er gesloten formules voor het bepalen van de inverse matrix (zie o.a. Wikipedia).

De inverse van een 2×2 matrix kan als volgt bepaald worden. Gegeven de matrix:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

is de inverse van deze matrix:

$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Hierbij staat $|A|$ voor de *determinant* van de matrix A die voor een 2×2 matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ dus $ad - bc$ is.

De inverse van een 3×3 matrix kan als volgt bepaald worden.

Gegeven de matrix: $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$

is de inverse van die matrix:

$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{13} & a_{12} \\ a_{33} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ \begin{vmatrix} a_{23} & a_{21} \\ a_{33} & a_{31} \end{vmatrix} & \begin{vmatrix} a_{13} & a_{12} \\ a_{33} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{13} & a_{11} \\ a_{23} & a_{21} \end{vmatrix} \\ \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{11} \\ a_{32} & a_{31} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}$$

1 Er zijn verschillende methoden om de determinant van een matrix te bepalen maar een praktische methode is het vegen van de matrix met behulp van Gauss tot een bovendriehoeksmatrix en dan is de determinant het product van de diagonaal.

Hierbij staat $\begin{vmatrix} a_{xx} & a_{xx} \\ a_{xx} & a_{xx} \end{vmatrix}$ voor de determinant van de gegeven 2×2 matrix. Deze 2×2 matrix wordt de *minor* genoemd. De minor M van een element A_{ij} is de determinant van de matrix die overblijft als alle elementen in dezelfde rij en kolom als dat element geschrapt worden. De determinanten van de 2 matrices in de grote matrix kunnen met de eerder genoemde gesloten formule voor determinanten uitgerekend worden.

Voor de berekening van de determinant van deze matrix zie Wikipedia en de links in dat lemma.

Van een 4×4 matrix kan op analoge wijze (recursief) een gesloten formule bepaald worden. Voor meer informatie hierover verwijs ik naar het Internet: daar zijn wel formules te vinden, o.a. op Wikipedia. Hoewel het zowel in theorie als in de praktijk mogelijk is om voor grotere dan 4×4 matrices gesloten formules op te stellen wordt dit meestal niet gedaan omdat er algoritmes bestaan met een kleiner complexiteit dan de uitgeschreven gesloten variant.

3.2.2.3. GROTE VIERKANTE MATRICES

Omdat het vinden van de determinant met behulp van de minor (en de minor van de minor en de minor daar weer van, enzovoorts) een recursieve functie is kan in beginsel voor iedere matrix de determinant in recursieve vorm gevonden en uitgeschreven worden. Behalve dat het een "leuke" en daarom populaire opdracht voor progammeer-cursussen is, is de complexiteit van een dergelijk functie $O(n!)$. Praktisch gezien is de bovenstaande methode daarom ook al vrij snel niet echt bruikbaar.

Voor grotere matrices wordt daarom Gauss-Jordan eliminatie, LU decompositie of Singular Value decompositie gebruikt. Deze hebben een complexiteit van $O(n^3)$. In dictaat wordt alleen Gauss-Jordan eliminatie behandeld.

3.2.2.4. NIET-VIERKANTE MATRICES

Van niet-vierkante matrices kan geen gewone inverse matrix bepaald worden. Voor niet-vierkante matrices kan wel een z.g. pseudo-inverse matrix bepaald worden waarvan de Moore–Penrose pseudoinverse een erg bekende en veel gebruikte is. Zie voor een verdere uitleg hoofdstuk 6.

3.2.3. INVERTEREN MET GAUSS-JORDAN ELIMINATIE

Als je de inverse van een matrix wil bepalen met behulp van Gauss-Jordan eleminatie[2] dan gaat dat als volgt. Zie paragraaf 4.3 voor een uitleg van Gauss-Jordan eleminatie.

Gegeven is een vierkante matrix:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{bmatrix}$$

Breid de matrix aan de rechterkant met de identiteitsmatrix uit:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 2 & 3 & 0 & 1 & 0 \\ 1 & 4 & 9 & 0 & 0 & 1 \end{bmatrix}$$

Pas vervolgens Gauss-Jordan net zolang toe totdat de identiteitsmatrix aan de linkerkant staat, zodat de matrix van de volgende vorm is, waarbij "*" voor een willekeurige getal staat:

$$\begin{bmatrix} 1 & 0 & 0 & * & * & * \\ 0 & 1 & 0 & * & * & * \\ 0 & 0 & 1 & * & * & * \end{bmatrix}$$

Als dat gelukt is dan staat nu aan de rechterkant de inverse matrix van de oorspronkelijke linkerkant. Dus eerst Gauss...

Stap 1: trek rij 1 van rij 2 en rij 3 af zodat die beide beginnen met 0.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 2 & -1 & 1 & 0 \\ 0 & 3 & 8 & -1 & 0 & 1 \end{bmatrix}$$

Stap 2: trek rij 2 van rij 1 en 3 keer rij 2 van rij 3 af zodat in de tweede kolom bij beide een 0 staat.

2 Het volgende is grotendeels een bewerking van de Wikipedia–lemma's Lineaire algebra, Lineaire vergelijkingen, Gauss eliminatie en Gauss-Jordan eliminatie. Voor meer uitleg over Gauss-Jordan eleminatie in dit dictaat zie sectie 4.3

$$\begin{bmatrix} 1 & 0 & -1 & 2 & -1 & 0 \\ 0 & 1 & 2 & -1 & 1 & 0 \\ 0 & 0 & 2 & 2 & -3 & 1 \end{bmatrix}$$

Stap 3: deel rij 3 door 2.

$$\begin{bmatrix} 1 & 0 & -1 & 2 & -1 & 0 \\ 0 & 1 & 2 & -1 & 1 & 0 \\ 0 & 0 & 1 & 1 & -\frac{3}{2} & \frac{1}{2} \end{bmatrix}$$

De matrix staat nu "in Gauss" (aangevulde matrix in rij-echelonvorm). Nu op naar Jordan...

Stap 4: tel rij 3 op bij rij 1 en trek 2 keer rij 3 af van rij 2.

$$\begin{bmatrix} 1 & 0 & 0 & 3 & -\frac{5}{2} & \frac{1}{2} \\ 0 & 1 & 0 & -3 & 4 & -1 \\ 0 & 0 & 1 & 1 & -\frac{3}{2} & \frac{1}{2} \end{bmatrix}$$

Nu staat de matrix "in Gauss-Jordan". Laten we dit controleren. Omdat $M \cdot M^{-1} = I$:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{bmatrix} \begin{bmatrix} 3 & -\frac{5}{2} & \frac{1}{2} \\ -3 & 4 & -1 \\ 1 & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Reken maar na...

4. LINEAIRE VERGELIJKINGEN

Zie ook Wikipedia:

- Lineaire vergelijkingen (Nederlands)
- Linear equations (Engels)

Lineaire vergelijkingen zijn vergelijkingen van de volgende vorm:

$a_1x_1 + a_2x_2 + a_3x_3 + \ldots + a_nx_n = y$

Hierbij zijn $x_1, x_2, x_3, \ldots, x_n$ de variabelen (maximaal tot de macht 1), $a_1 \ldots a_n$ de coëfficiënten van de vergelijking en y een constante.

Een stelsel van lineaire vergelijkingen is een verzameling van lineaire vergelijkingen en ziet er zo uit:

$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = y_1$
$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = y_2$
$\ldots$
$a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \ldots + a_{mn}x_n = y_m$

Het vinden van getallen voor x_1 tot en met x_n, *gegeven* alle coëfficiënten a_{mn} en constanten y_m, heet het oplossen van het stelsel. Als er net zoveel vergelijkingen zijn als onbekende variabelen dan is een dergelijk stelsel (mits überhaupt oplosbaar) triviaal op te lossen.

Hierna volgen twee methodes om een stelsel van lineaire vergelijkingen op te lossen: de substitutiemethode en Gauss-Jordan eliminatie.

4.1. SUBSTITUTIEMETHODE

De meest gebruikte methode om simpele stelsels op te lossen is de substititie-methode. Dit is ook de methode die de meesten op de middelbare school hebben gebruikt. Een voorbeeld van de subsititutie–methode van een simpel stelsel met 2 variabelen gaat als volgt.

Stel, gegeven is het stelsel:
$y = 5x + 1$
$y = 2x + 7$

Dan:
$5x + 1 = 2x + 7$, immers: $a = b \wedge b = c \Leftrightarrow a = c$ dus $5x + 1 = y = 2x + 7$
$5x - 2x + 1 = 2x - 2x + 7$, immers als $a = a \Leftrightarrow a + b = a + b$
$3x + 1 = 7$, HTKW (huis–tuin–en–keuken–wiskunde, rekenenen?)
$3x + 1 - 1 = 7 - 1$, immers als $a = a \Leftrightarrow a + b = a + b$
$3x = 6$, HTKW
$x = 2$, immers $a = a \Leftrightarrow a/b = a/b$
$y = 2 * 2 + 7$ of $y = 5 * 2 + 1$, substitutie $x = 2$
$y = 11$, HTKW

Een andere methode die je had kunnen volgen voor hetzelfde stelsel gaat als volgt. Stel, gegeven is wederom:
$y = 5x + 1$
$y = 2x + 7$

Dan:
$y - y = (5x + 1) - (2x + 7)$, immers $a = a \Leftrightarrow a + b = a + b$
$0 = 3x - 6$
$x = 2$, immers $a = b \Leftrightarrow b = a$ en $a = a \Leftrightarrow a/b = a/b$
$y = 2 * 2 + 7$ of $y = 5 * 2 + 1$, substitutie $x = 2$
$y = 11$, HTKW

De regels die hierboven staan en gebruikt worden zijn als volgt samen te vatten. Stel dat gegeven is $a =$

b met $a, b \in \mathbb{R}$. Dan zijn de volgende regels van toepassing:

- $a = b \Leftrightarrow b = a$
- $a = b \Leftrightarrow a + x = b + x,\ x \in \mathbb{R}$.
- $a = b \Leftrightarrow a * x = b * x,\ x \in \mathbb{R}$.

Waarschijnlijk overbodig maar de laatste bovenstaande regels werken natuurlijk ook voor aftrekken en delen. Aftrekken is immers het optellen van het tegenovergestelde en delen is het vermenigvuldigen met het omgekeerde.

Deze regels kunnen gebruikt worden om ingewikkeldere vergelijkingen op te lossen dan vergelijkingen met 2 onbekenden. Hoewel de subsitutiemethode ook altijd werkt is het bij het oplossen van vergelijkingen met veel onbekende variabelen nogal arbeidsintensief en foutgevoelig. Bovendien veronderstelt het (menselijk) nadenken en is de procedure lastig te automatiseren.

4.2. STELSELS VAN LINEAIRE VERGELIJKINGEN ALS MATRIX

4.2.1. AANGEVULDE MATRIX

Nogmaals de vorm van een uitgebreider stelsel van lineaire vergelijkingen:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = y_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = y_2$$
$$\ldots$$
$$a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \ldots + a_{mn}x_n = y_m$$

Als we afspreken dat iedere rij altijd netjes op dezelfde wijze op $x_1, x_2, x_3, \ldots x_n$ is gesorteerd (bijvoorbeeld op alfabet) en dat als er een x_* ontbreekt er daar 0 komt te staan dan kan een dergelijk stelsel in een matrix–vorm geschreven worden zonder dat we informatie verliezen:[1]

$$\left[\begin{array}{ccccc|c} a_{11}x_1 & a_{12}x_2 & a_{13}x_3 & \ldots & a_{1n}x_n & y_1 \\ a_{21}x_1 & a_{22}x_2 & a_{23}x_3 & \ldots & a_{2n}x_n & y_2 \\ \ldots & & & & & \\ a_{m1}x_1 & a_{m2}x_2 & a_{m3}x_3 & \ldots & a_{mn}x_n & y_m \end{array}\right]$$

Een dergelijke matrix wordt een *aangevulde matrix* genoemd (in het Engels: *augmented matrix*). Overigens wordt de verticale streep aan de rechterkant ook wel eens weg gelaten.

Een praktisch voorbeeldje. Gegeven het volgende stelsel van vergelijkingen:

$$x_1 + 3x_2 + 4x_3 = 8$$
$$x_1 + 2x_2 + 5x_3 = 13$$
$$3x_1 + x_2 - x_3 = 1$$

Dan wordt de bijbehorende aangevulde matrix:

$$\left[\begin{array}{ccc|c} 1 & 3 & 4 & 8 \\ 1 & 2 & 5 & 13 \\ 3 & 1 & -1 & 1 \end{array}\right]$$

4.2.2. AANGEVULDE MATRIX IN RIJ–ECHELONVORM

Een matrix staat in de zogenaamde *rij–echelonvorm* als de aangevulde matrix de volgende vorm heeft:

---

1 Een andere observatie is dat we het stelsel ook kunnen schrijven als de vermenigvuldiging van een matrix met een column vector die een bepaalde uitkomst moet hebben:
Vergelijk

$$2A - 10B + 6C = 18$$
$$2A - 6B + 4C = 22$$
$$-A + 9B - C = 27$$

maar eens met

$$\begin{bmatrix} 2 & -10 & 6 \\ 2 & -6 & 4 \\ -1 & 9 & -1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} 18 \\ 22 \\ 27 \end{bmatrix}$$

Schrijf de vermenigvuldiging maar uit en kijk wat er in cellen komt te staan.

$$\left[\begin{array}{cccc|c} 1 & * & * & * & * \\ 0 & 1 & * & * & * \\ 0 & 0 & 1 & * & * \\ 0 & 0 & 0 & 1 & * \end{array}\right]$$

Hierbij is "$*$" telkens een willekeurig getal. Een dergelijke matrix wordt ook wel een "bovendriehoeks-matrix" genoemd.

Hier gelden de volgende regels:

- Elk eerste niet–nul element in een rij gelijk aan 1.
- De niet-nulrijen zijn zo geordend dat elke volgende rij met meer nullen begint.
- Eventuele rijen met alleen maar nullen staan onderaan in de matrix.

Het leuke van een matrix in rij–echelon is dat op de rij die begint met allemaal nullen en die afgesloten wordt door een 1 gevolgd door een willekeurig getal eigenlijk de vergelijking $0 * \textit{de andere getallen} + 1 * x = \textit{getal}$ staat. Zodra je dat weet is de rest gewoon een kwestie van invullen...

4.2.3. AANGEVULDE MATRIX IN GEREDUCEERDE RIJ–ECHELONVORM

Een matrix staat in de zogenaamde *gereduceerde rij–echelonvorm* als de aangevulde matrix de volgende vorm heeft:

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & * \\ 0 & 1 & 0 & 0 & * \\ 0 & 0 & 1 & 0 & * \\ 0 & 0 & 0 & 1 & * \end{array}\right]$$

Hierbij is "$*$" telkens een willekeurig getal.

Hier gelden de volgende regels:

- De matrix staat in de rij-echelonvorm.
- In een kolom met een leidende 1 is elk ander element gelijk aan 0 (behalve het laatste element dat iedere willekeurige waarde mag hebben)

Het is natuurlijk onmiddelijk duidelijk dat het leuke van een matrix in gereduceerde rij–echelonvorm is dat ineens alle variabelen onmiddelijk aflezen kunnen worden. Nu staat immers op *iedere* regel de vergelijking $1 * x + 0 * \textit{een ander getal} = \textit{getal}$.

4.3. GAUSS-JORDAN ELIMINATIE

Gauss-Jordan [2] eliminatie is een systematische manier om een stelsel van lineaire vergelijkingen op te lossen. Hiervoor wordt het stelsel eerst geschreven als een matrix, waarna er met behulp van elementaire operaties op de matrix een oplossing gevonden wordt voor het stelsel.

4.3.1. OPERATIES OP EEN MATRIX

Op een matrix mogen de volgende bewerkingen uitgevoerd worden:

- Rijen verwisselen.
- Rijen bij elkaar optellen of van elkaar aftrekken.
- Rijen vermenigvuldigen met een constante.

De eerste operatie is triviaal: het verwisselen van de volgorde van rijen verandert natuurlijk de uitkomst van het stelsel niet. De tweede en derde operatie zijn variaties op de regels die we ook al gezien hebben bij de substitutie–methode. Je mag immers in een vergelijking altijd links en rechts dezelfde operatie uitvoeren (optellen/aftrekken met hetzelfde, vermenigvuldigen/delen met hetzelfde) en de rijen zijn vergelijkingen in vermomming.

2 Hoewel hij de methode niet heeft uitgevonden (dat hebben wat onbekende Chinezen in ongeveer 150 voor Christus al in meer of mindere mate gedaan) is de techniek om stelsels van lineaire vergelijkingen op te lossen naar Gauss vernoemd (ja, die van de kromme). Jordan heeft de techniek verder vervolmaakt, vandaar dat zijn naam er ook bij staat.

4.3.2. GAUSS–ELIMINATIE

Gaus–eliminatie[3] is een techniek om met behulp van de hierboven genoemde operaties een willekeurig matrix in rij–echelonvorm te krijgen. Nadat de matrix in rij–echelon staat is op de onderste rij onmiddelijk af te lezen wat de oplossing voor de meest rechter variabele in de vergelijkingen is. Daarna kan door subtitutie de overige variabelen uitgerekend worden.

Gegeven is het volgende stelsel:
$x + y + z = 6$
$x + 2y + 3z = 14$
$x + 4y + 9z = 36$

Daarbij hoort de volgende aangevulde matrix:

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 1 & 2 & 3 & 14 \\ 1 & 4 & 9 & 36 \end{array}\right]$$

Om de eerste rij in echelon–vorm te krijgen hoeven we niets te doen: de eerste rij begint immers al met een 1. Om op de eerste plaats van de tweede rij een 0 te krijgen trekken we rij 1 van rij 2 af. Ditzelfde doen we voor rij 3: ook hier trekken wij rij 1 van rij 2 af.

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & 1 & 2 & 8 \\ 0 & 3 & 8 & 30 \end{array}\right]$$

Rij 1 en rij 2 staan nu in echelon–vorm. Nu moet rij 3 nog. Eerst zorgen we dat er een 0 komt te staan op de plaats van de 3: we trekken 3 keer rij 2 van rij 3 af.

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & 1 & 2 & 8 \\ 0 & 0 & 2 & 6 \end{array}\right]$$

Tot slot delen we rij 3 door 2.

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & 1 & 2 & 8 \\ 0 & 0 & 1 & 3 \end{array}\right]$$

Op de onderste rij staat nu: $0x + 0y + 1z = 3$ dus $z = 3$.
Op de tweede rij staat nu: $0x + 1y + 2z = 8$, $z = 3$ dus $y = 8 - 2 * 3 = 2$.
Op de eerste rij staat nu $x + y + z = 6$, $y = 2', z = 3$ dus $x = 6 - 2 - 3 = 1$

4.3.3. GAUSS–JORDAN–ELIMINATIE

De Zwitserse wiskundige Jordan bedacht dat je niet hoefde te stoppen bij een matrix in rij–echelon. Als je de matrix in gereduceerde rij–echelonvorm brengt dan hoef je niet eens te substitueren maar kan je de oplossing van het stelsel onmiddelijk aflezen in de matrix. Een methode om dit te doen is, als je de matrix al in rij–echelonvorm hebt, gewoon verder gaan met vegen, maar dan van onderaf.

We hadden al de matrix in rij–echelonvorm:

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 6 \\ 0 & 1 & 2 & 8 \\ 0 & 0 & 1 & 3 \end{array}\right]$$

Om de tweede rij in gereduceerde rij–echelon te krijgen trekken we 2 keer rij 3 van rij 2 af. Om in kolom 3 van rij 1 een 0 te krijgen trekken we rij 3 van rij 1 af.

$$\left[\begin{array}{ccc|c} 1 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{array}\right]$$

Om de eerste rij tot slot in de juiste vorm te krijgen trekken we rij 2 van rij 1 af.

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{array}\right]$$

En onmiddelijk zijn de waardes van x, y en z af te lezen.

3 Het uitvoeren van een Gaus-eliminatie wordt ook wel "het vegen van een matrix" genoemd. Dat komt omdat er uiteindelijk een soort trap-vorm-matrix ontstaat waarbij de voorkant schoongeveegd is. Wie zegt dat wiskundigen geen humoristische fantasieën hebben. . . ?

Mocht je willen oefenen: het internet staat vol met voorbeelden. Je kan ook je eigen matrix verzinnen, oplossen en kijken of je het goed gedaan hebt op de volgende website: Matrix Refresh.

Als je je eigen matrix verzint dan is het het handigst dat je eerst de uitkomt bepaalt en daarbij vergelijkingen verzint.

5. FORWARD KINEMATICS

Zie ook Wikipedia:

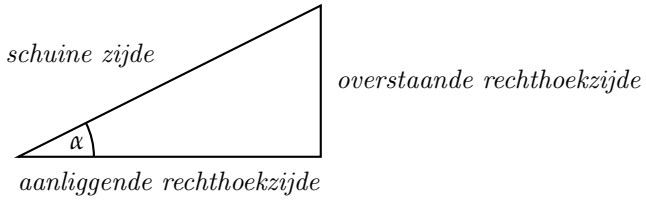
- Voorwaartse kinematics (Nederlands): Niet beschikbaar
- Forward kinematics (Engels)

In de context van de course Robots is "forward kinematics" het berekenen van de positie van het eindpunt van de arm van een robot, gegeven de hoeken van de gewrichten en de lengtes van de armdelen.

Op de middelbare school heb je het ezelsbruggetje "SOS/CAS/TOA" geleerd om de definities van sinus, cosinus en tangens te onthouden:

- SOS: $\sin = \frac{overstaande\ rechthoekzijde}{schuine\ zijde}$
- CAS: $\cos = \frac{aanliggende\ rechthoekzijde}{schuine\ zijde}$
- TOA: $\tan = \frac{overstaande\ rechthoekzijde}{aanliggende\ rechthoekzijde}$ [1].

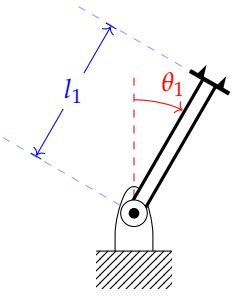
Gegegeven is de onderstaande rechthoekige driehoek waarvan de hoek α en de lengte van de schuine zijde bekend zijn.



Met behulp van SOS en CAS zijn dan de lengtes van de overstaande en aanligende rechthoekzijden eenvoudig te berekenen. Immers:

- $cos(\alpha) = \frac{aanliggende}{schuine}$ dus $aanliggende = schuine \times cos(\alpha)$
- $sin(\alpha) = \frac{overstaande}{schuine}$ dus $overstaande = schuine \times sin(\alpha)$

Deze kennis kan je gebruiken om de coördinaten van het eindpunt van een robot–arm te berekenen. Stel dat je een robotarm hebt met 1DOF zoals in de onderstaande tekening. De x– en y–coördinaten van de gripper (eindpunt, e) zijn dan vervolgens simpel te bepalen.



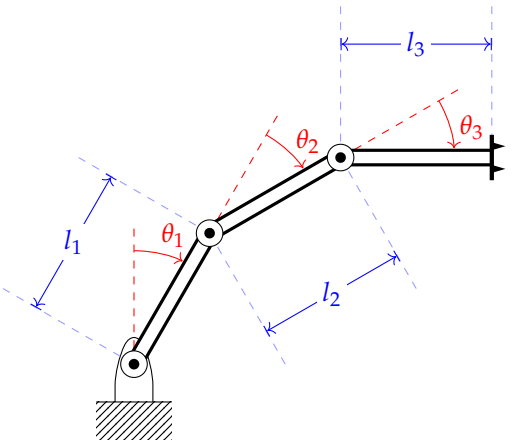
Stel dat $\theta_1 = 30$[2] en $l_1 = 5$. Dan zijn de coördinaten respektievelijk:
$x_{eindpunt} = l_1 \sin(\theta_1) = 5 \cdot \sin(30) = 2.5$
$y_{eindpunt} = l_1 \cos(\theta_1) = 5 \cdot \cos(30) = 4.3$

Hierbij is er van uit gegaan dat het draaipunt van het gewricht zich op $(0, 0)$ bevindt. Als dat niet het geval is dan wordt de formule als volgt:
$x_{eindpunt} = x_0 + l_1 \sin(\theta_1)$

1 TOA mag je meteen weer vergeten omdat deze voor informatici minder bruikbaar is. De tangens is namelijk niet gedefinieerd voor hoeken van $90°/270°$ ($\frac{\pi}{2}$ / $\frac{3\pi}{2}$) en is dus lastig te gebruiken in functies die het voor alle hoeken moeten doen...

2 Het maakt niet zoveel uit hoe je de hoeken kiest: rechts– of linksom gemeten, ten opzichte van de x–as of y–as: zo lang je maar consequent bent en een beetje nadenkt over de correcte formules. In dit dictaat gebruik ik altijd de rechtsom gemeten hoek ten opzichte van de y–as

$y_{eindpunt} = y_0 + l_1 \cos(\theta_1)$,
waarbij x_0 en y_0 de x– en y–coördinaten zijn van het gewricht.



Het uitrekenen van het eindpunt van een 3DOF–arm is een triviale uitbreiding op het uitrekenen van het eindpunt van een 1DOF–arm. Immers:
$x_{eindpunt} = x_0 + \Delta x \text{ door } armdeel1 + \Delta x \text{ door } armdeel2 + \Delta x \text{ door } armdeel3$
$y_{eindpunt} = y_0 + \Delta y \text{ door } armdeel1 + \Delta y \text{ door } armdeel2 + \Delta y \text{ door } armdeel3.$

Uitgedrukt in *sin/cos* wordt dit:
$x_{eindpunt} = x_0 + l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3)$
$y_{eindpunt} = y_0 + l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3)$

De bovenstaande wiskundige functies zijn triviaal naar C++ over te zetten.

5.1. AFGELEIDE VAN DE FORWARD KINEMATICS FUNCTIE

Stel dat je van de bovenstaande functies zou willen voorspellen wat de verandering van de x– en y-coördinaten is als één of meer van de hoeken verandert. Bij een één–dimensionale functie gebruikten we daar de afgeleide van de functie voor. Immers: $\Delta y = \frac{dy}{dx}\Delta x$ waarbij $\frac{dy}{dx}$ de afgeleide van de functie was[3]. Het (wellicht) verrassende is dat we dit ook bij een multidimensionale functie doen met behulp van de afgeleide. Maar wat is de afgeleide van een multidimensionale functie en hoe noteren we die?

Intuitief is de verandering van de x– en y-coördinaten als gevolg van de verandering één of meer van de hoeken het optellen van de invloeden van de verandering van θ_1, θ_2 en θ_3 op de beide coördinaten:
$\Delta x_{eindpunt} = \Delta x \text{ door } \theta_1 + \Delta x \text{ door } \theta_2 + \Delta x \text{ door } \theta_3$
$\Delta y_{eindpunt} = \Delta y \text{ door } \theta_1 + \Delta y \text{ door } \theta_2 + \Delta y \text{ door } \theta_3.$

Analoog aan $\Delta y = \frac{dy}{dx}\Delta x$, waarbij $\frac{dy}{dx}$ de afgeleide van de functie is, Δx de verandering van de onafhankelijke variabele x en Δy de voorspelde (berekende) verandering van de afhankelijke variabele y:

Voor Δx:

- $\Delta x \text{ door } \theta_1 = (afgeleide\ van\ multidimensionale\ functie\ voor\ x) \cdot \Delta\theta_1$
- $\Delta x \text{ door } \theta_2 = (afgeleide\ van\ multidimensionale\ functie\ voor\ x) \cdot \Delta\theta_2$
- $\Delta x \text{ door } \theta_3 = (afgeleide\ van\ multidimensionale\ functie\ voor\ x) \cdot \Delta\theta_3$

Voor Δy:

- $\Delta y \text{ door } \theta_1 = (afgeleide\ van\ multidimensionale\ functie\ voor\ y) \cdot \Delta\theta_1$
- $\Delta y \text{ door } \theta_2 = (afgeleide\ van\ multidimensionale\ functie\ voor\ y) \cdot \Delta\theta_2$
- $\Delta y \text{ door } \theta_3 = (afgeleide\ van\ multidimensionale\ functie\ voor\ y) \cdot \Delta\theta_3$

In dat geval moet dus de afgeleide van een multidimensionale functie worden bepaald. Een afgeleide van een multidimensionale functie is iets moeilijker dan van een gewone functie. De afgeleide van een multidimensionale functie is een matrix met net zoveel columns als er onafhankelijke variabelen zijn en net zoveel rows als er afhankelijke variabelen zijn. De bovenstaande forward kinematics functie is een functie van $\mathbb{R}^3 \rightarrow \mathbb{R}^2$. Dat betekent dat de afgeleide van die functie een $2x3$–matrix is: 2 rows, 3 columns. Een dergelijke matrix wordt een *Jacobi matrix* genoemd en wordt genoteerd als $J(e,\theta)$. Hierbij staat e voor de voorspelde e(ffecten), in ons geval coördinaten. De θ staat voor de hoeken in de formules. Een

3 Dit volgt uit: $f'(x) = \frac{df}{dx} = \frac{dy}{dx} = \frac{\Delta y}{\Delta x}$. Wat herschrijven van deze vergelijkingen en de klus is geklaard.

Jacobi–matrix is in deze context dus op te vatten als de afgeleide van de forward kinematics functie[4].

In de bovenste row van de matrix komen de factoren te staan die bijdragen aan het veranderen van de x-coördinaat, in de onderste row de factoren die bijdragen aan het veranderen van de y-coördinaat. In de eerste column komen de factoren te staan die θ_1 bijdraagt aan de verandering van beide coördinaten, in column 2 die van θ_2 en in column 3 die van θ_3. Iedere factor in een cel is een z.g. *partiële afgeleide*. Een partiële afgeleide is een gewone afgeleide van een multidimensionale functie waarbij telkens één van de variabelen gezien wordt als DE variabele en alle andere variabelen als constantes. Men differentieert dus naar een bepaalde variabele.

Voor de x-row betekent dit:

$\frac{dx}{d\theta_1} = l_1\cos(\theta_1) + l_2\cos(\theta_1+\theta_2) + l_3\cos(\theta_1+\theta_2+\theta_3)$
$\frac{dx}{d\theta_2} = l_2\cos(\theta_1+\theta_2) + l_3\cos(\theta_1+\theta_2+\theta_3)$
$\frac{dx}{d\theta_3} = l_3\cos(\theta_1+\theta_2+\theta_3)$

Voor de y-row betekent dit:

$\frac{dy}{d\theta_1} = -l_1\sin(\theta_1) - l_2\sin(\theta_1+\theta_2) - l_3\sin(\theta_1+\theta_2+\theta_3)$
$\frac{dy}{d\theta_2} = -l_2\sin(\theta_1+\theta_2) - l_3\sin(\theta_1+\theta_2+\theta_3)$
$\frac{dy}{d\theta_3} = -l_3\sin(\theta_1+\theta_2+\theta_3)$

Dus: $J(e,\theta) =$

$$\begin{bmatrix} \frac{dx}{d\theta_1} & \frac{dx}{d\theta_2} & \frac{dx}{d\theta_3} \\ \frac{dy}{d\theta_1} & \frac{dy}{d\theta_2} & \frac{dy}{d\theta_3} \end{bmatrix} =$$

$$\begin{bmatrix} l_1\cos(\theta_1) + l_2\cos(\theta_1+\theta_2) + l_3\cos(\theta_1+\theta_2+\theta_3) & l_2\cos(\theta_1+\theta_2) + l_3\cos(\theta_1+\theta_2+\theta_3) & l_3\cos(\theta_1+\theta_2+\theta_3) \\ -l_1\sin(\theta_1) - l_2\sin(\theta_1+\theta_2) - l_3\sin(\theta_1+\theta_2+\theta_3) & -l_2\sin(\theta_1+\theta_2) - l_3\sin(\theta_1+\theta_2+\theta_3) & -l_3\sin(\theta_1+\theta_2+\theta_3) \end{bmatrix}$$

En nu blijkt de intuitie van hierboven correct. Immers:

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \frac{dx}{d\theta_1} & \frac{dx}{d\theta_2} & \frac{dx}{d\theta_3} \\ \frac{dy}{d\theta_1} & \frac{dy}{d\theta_2} & \frac{dy}{d\theta_3} \end{bmatrix} \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix}$$

Waarna de matrix/column–vermenigvuldiging expandeert tot:

$\Delta x = \frac{dx}{d\theta_1}\Delta\theta_1 + \frac{dx}{d\theta_2}\Delta\theta_2 + \frac{dx}{d\theta_3}\Delta\theta_3$
$\Delta y = \frac{dy}{d\theta_1}\Delta\theta_1 + \frac{dy}{d\theta_2}\Delta\theta_2 + \frac{dy}{d\theta_3}\Delta\theta_3$

Wat dan inderdaad erg lijkt op:

$\Delta x_{eindpunt} = \Delta x \textit{ door } \theta_1 + \Delta x \textit{ door } \theta_2 + \Delta x \textit{ door } \theta_3$
$\Delta y_{eindpunt} = \Delta y \textit{ door } \theta_1 + \Delta y \textit{ door } \theta_2 + \Delta y \textit{ door } \theta_3.$

4 Meer in het algemeen is de matrix op te vatten als de afgeleide van een functie van $\mathbb{R}^m \rightarrow \mathbb{R}^n$ en wordt ook voor veel meer doeleinden dan forward kinematics gebruikt.

6. INVERSE KINEMATICS

Zie ook Wikipedia:

- Inverse kinematica (Nederlands))
- Inverse kinematics (Engels)

In de context van de course Robots is "inverse kinematics" het berekenen van de hoeken van de gewrichten van een robot, gegeven de positie van het eindpunt van de arm en de lengtes van de armdelen.

6.1. INVERSE KINEMATICS ALGORITME

Forward kinematics is makkelijk uit te rekenen, inverse kinematics in het algemeen niet. Voor simpele 1– en 2DOF (en zelfs voor 3DOF) armen zijn er wel gesloten, analytische formules. Voor 3DOF of hoger moet terug gevallen worden op numerieke wiskunde: het gebruiken van benaderingen om tot een antwoord te komen.

De Duitse wiskundige Carl Gustav Jacob Jacobi heeft een numeriek algoritme uitvonden.

Het volgende algoritme is een basis inverse kinematics algoritme waarbij gebruik gemaakt wordt van een Jacobi-matrix.

```
while ( e is too far from g)
  Compute J(e,θ) for the current pose θ
  Compute J^-1
  Δe = β(g − e)
  Δθ = J^-1 × Δe
  θ_nieuw = θ_oud + Δθ
  Compute new e vector
```

Hierbij betekenen de toverformules in het bovenstaande algoritme de volgende dingen:

- e is de huidige vector (coördinaten) van het eindpunt van de arm, i.e. *end effector*
- g is de vector (coördinaten) van het doel–eindpunt, i.e. *goal* ,
- $J(e,\theta)$ is de Jacobi–matrix met daarin de afgeleide van de forward kinematics functie, uitgerekend met de huidige configuratie–vector van de arm,
- J^{-1} is de inverse matrix van deze Jacobi–matrix,
- $\Delta\theta = \beta(g-e)$ is de volgende stap die we gaan nemen, waarbij
  1. $(g-e)$ het verschil is tussen de huidige en gewenste vector van het eindpunt van de arm,
  2. β correctie op de grootte van de stap, waarbij $0 < \beta < 1$
  3. Δe is de stap die we daadwerkelijk gaan nemen,
- $\Delta\theta = J^{-1} \times \Delta e$ berekent met behulp van de afgeleide van inverse Jacobi–matrix en de te nemen stap de verandering van de toekomstige waarde van de configuratie–vector van de arm,
- $\theta_{nieuw} = \theta_{oud} + \Delta\theta$ berekent de uiteindelijke nieuwe waarde van de configuratie–vector van de arm,
- Compute new e vector berekent met de forward kinematic functie de vector van het eindpunt als we deze stap daadwerkelijk nemen.

De inverse $J(e,\theta)^-$ van $J(e,\theta)$ kan op een aantal manieren worden bepaald.

Voor vierkante matrices is de procedure al beschreven in hoofdstuk 3.2.2.

Van niet-vierkante matrices kan geen gewone inverse matrix bepaald worden. Voor niet- vierkante matrices kan wel een z.g. pseudo-inverse matrix bepaald worden waarvan de MoorePenrose pseudoinverse een erg bekende en veel gebruikte is.

Tot slot kan J^T gebruikt worden als inverse matrix.

7. KANSREKENING

Zie ook Wikipedia:

- probability theory

7.1. INLEIDING

Kansrekening houdt zich binnen de wiskunde bezig met onzekerheid en toeval. Het gaat daarbij onder andere over de vraag of je iets zinnigs kan zeggen over de waarde van variabelen als de waarde van die variabelen door het toeval worden bepaald.

Diverse Arabische geleerden hebben tussen de achtste en dertiende eeuw al onderdelen van kansrekening gebruikt bij hun onderzoek naar cryptografie. In de zestiende en zeventiende was men vooral geïnteresseerd in kansrekening in het kader van kaart- en dobbelspelletjes. Onder andere de Nederlander Christiaan Huygens[1] heeft zijn steentje daaraan bijgedragen door er een heel boek over te schrijven. In het begin van de negentiende eeuw heeft de Franse markies Laplace[2] de basis gelegd voor Bayesiaanse statistiek en kansrekening. In essentie heeft hij de basis voor veel wiskunde die wij gebruiken in WoR gelegd. Maar uiteindelijk heeft de Russische wiskundige Kolmogorov pas in de eerste helft van de twintigste eeuw de kansrekening een echt wiskundig fundament gegeven.

Uit het bovenstaande korte historische overzicht blijkt dat het best wel lang geduurd heeft om grip te krijgen op kansrekening. In dit hoofdstuk gaan we in op die onderdelen van de kansrekening die we later nodig hebben voor diverse probabilistische filters.

7.2. INFORMEEL

Het is opvallend ingewikkeld om precies uit te leggen wat een kans is zonder een synoniem van kans, of iets dat daar sterk op lijkt, te gebruiken. Je vervalt al snel in zinnen als “een kans is de mogeljkheid dat . . . ” of “een kans is dat waarschijnlijkheid dat . . . .”. Toch een poging, dus. . .

Een kans is een getal dat aangeeft hoe vaak een bepaalde gebeurtenis optreedt binnen een verzameling gebeurtenissen.

Een kans op een bepaalde (verzameling van) gebeurtenis(sen) wordt berekend door het aantal gunstige gebeurtenissen te delen door het totaal aantal mogelijke gebeurtenissen. Een gebeurtenis is hierbij het (eenmalig) toewijzen of observeren van de waarde van één of meerdere variabelen. De variabelen die zo een waarde krijgen worden stochastische (random) variabelen genoemd.

Een praktisch voorbeeld van een gebeurtenis is het werpen van een dobbelsteen. Door het werpen van de dobbelsteen krijgt de stochastische variabele “worp” een bepaalde waarde. Die zal bijvoorbeeld 6 zijn als er 6 gegooid wordt. Een ander praktische voorbeeld is het trekken van 10 kaarten uit één spel kaarten waarbij 5 kaarten rood en 5 kaarten zwart zijn[3]. Hierbij bestaat de gebeurtenis dus uit de stochastische variable “1-greep” die zijn waarde krijgt door 10 kaarten uit een spel te trekken.

De vraag naar hoe vaak een gebeurtenis optreedt is soms makkelijk en soms wat moeilijker te bepalen. In het geval van de dobbelsteen ziet vrijwel iedereen onmiddelijk in dat er maar 1 manier is om 6 te gooien. Maar op hoeveel manieren je met een 10-greep 5 rode en 5 zwarte kaarten kunt trekken is niet triviaal. Met wat combinatoriek kan je het uitrekenen: dat kan op 4.327.008.400 manieren [4].

1 In 2012 is hij gekozen tot de grootste geleerde aller tijden van Nederland door het tijdschrift Quest.
2 Ja, die van het OpenCV-filter.
3 Het trekken van 10 elementen uit een verzameling wordt ook wel een 10-greep genoemd. En het trekken van 5 elementen uit een verzameling wordt ook wel een 5-greep genoemd. Kortom, het trekken van n elementen uit een verzamenling heet een n-greep.
4 Hoe gaat die berekening? Je doet eerst een 5-greep uit 26 rode kaarten en daarna een 5-greep uit 26 zwarte kaarten. Beiden kunnen op $\binom{26}{5}$ manieren (zie hier voor een hele korte uitleg). Volgens de productregel (als een eerste taak op m manieren kan worden uitgevoerd en een tweede taak op n manieren dan zijn er in totaal $m \cdot n$ manieren om alles achter

Hetzelfde geldt voor het bepalen van het totale aantal mogelijkheden: soms gaat dat gemakkelijk, soms wat moeilijker. De dobbelsteen heeft er 6, dat ziet iedereen: je kan ze gewoon tellen door een dobbelsteen te pakken. Maar het totaal aantal mogelijkheden om 10 kaarten uit een spel van 52 kaarten te trekken is niet onmiddelijk in te zien. Gelukkig kan je ook dat met wat combinatoriek uitrekenen: dat kan op 15.820.024.220 manieren [5]

Uit het voorgaande blijkt dat de kans om 6 te gooien dus $\frac{1}{6} = 0.17$ is: je kan op 1 manier 6 gooien op een totaal van 6 mogelijke uitkomsten. De kans om met 1 10-greep 5 zwarte en 5 rode kaarten te trekken is dus $\frac{4.327.008.400}{15.820.024.220} = 0.27$.

7.3. FORMELE DEFINITIES

De formele definitie van wat een kans is, is best wel ingewikkeld. In deze paragraaf worden voor de volledigheid de formele definities gegeven. De praktische consequenties komen vanzelf langs in de rest van het dictaat.

Een kansruimte is een triplet $\langle \Omega, \mathcal{F}, \mathcal{P} \rangle$ waarbij

- Ω een niet-lege verzameling van alle mogelijke uitkomsten is, die ook wel universum of sample-space wordt genoemd,
- $\mathcal{F}$ een collectie deelverzamelingen van Ω is, dat ook wel event-space wordt genoemd, waarbij de elementen *gebeurtenissen (events)* of *uitkomsten* worden genoemd, en
- $\mathcal{P}$ een kansverdelingsfunctie van $\mathcal{F} \rightarrow [0,1]$ is, een functie die aan ieder element uit $\mathcal{F}$ een kans toewijst.

Voor $\mathcal{F}$ geldt dat het een zogenaamde sigma-algebra moet zijn. Een collectie deelverzamelingen $\mathcal{F}$ van Ω is een sigma-algebra van verzamelingen over Ω als:

- $\Omega, \emptyset \in \mathcal{F}$, i.e. de gehele verzameling en de lege verzameling zitten in $\mathcal{F}$[6],
- $\mathcal{F}$ is gesloten onder complement, i.e. als $\forall f \in \mathcal{F}$ dan ook $f^c \in \mathcal{F}$ ($f^c = \Omega \setminus f$) [7], oftewel als het optreden van een gebeurtenis in $\mathcal{F}$ zit dan zit ook het niet-optreden van de gebeurtenis in $\mathcal{F}$,
- $\mathcal{F}$ is gesloten onder aftelbare vereniging, i.e. als $f_1 \ldots f_n \in \mathcal{F}$ dan ook $(\cup_n f_n) \in \mathcal{F}$, oftewel alle mogelijk deelverzamelingen die op basis van de elementen uit $\mathcal{F}$ gemaakt kunnen worden zijn deel van $\mathcal{F}$ zelf.

Dit komt er, heel in het kort, op neer dat we ook kunnen kansrekenen met aftelbaar oneindige verzamelingen. Hierbij is een aftelbare verzameling een verzameling van elementen die op een bepaalde volgorde gezet kunnen worden en die we vervolgens kunnen tellen, desnoods tot in het oneindige. Deze definitie lost het probleem op bij gevallen waarbij het niet mogelijk is om de mogelijke uitkomsten te tellen. Bijvoorbeeld bij dingen die met tijd of afstand van doen hebben: er zitten immers oneindig veel tijdstippen in een seconde of oneindig veel afstanden in een meter...

Voor de kansverdelingsfunctie $\mathcal{P}$ gelden de zogenaamde axioma's van Kolmogorov:

- $P(\Omega) = 1.0$, i.e alle kansen van de kansruimte bij elkaar opgeteld tellen op tot 1,
- Voor alle gebeurtenissen (events) $E \in \mathcal{F}$ geldt dat $P(E) \geq 0$, i.e. een kans is niet negatief,
- Voor alle gebeurtenissen (events) $E, F \in \mathcal{F}$ geldt dat als $E \cap F = \emptyset$ dan is $P(E \cup F) = P(E) + P(F) \geq 0$, i.e. gegeven twee of meer gebeurtenissen die niet tegelijk kunnen optreden [8] is de kans dat één van deze twee of meer gebeurtenissen optreden de som van de individuele kansen.

---

elkaar te doen) zijn er dan dus $\binom{26}{5} \cdot \binom{26}{5} = 65780 \cdot 65780 = 4.327.008.400$ mogelijkheden om twee dergelijke 5-grepen te doen. Voor de notatie $\binom{26}{5}$ (spreek uit als "26 boven 5" of "26 over 5": zie het Wikipedia-lemma "Binomiaalcoëficiënt"

5 Hoe gaat die berekening? Je doet een 10-greep uit 52 kaarten. Dat kan op $\binom{52}{10} = 15.820.024.220$ manieren (zie hier voor een hele korte uitleg).

6 Om universeel en ook van toepassing te zijn op oneindige verzamelingen moet de gehele sample-space Ω (alle mogelijke uitkomsten) onderdeel zijn van de event-space $\mathcal{F}$. Om vervolgens te voldoen aan de volgende regel (gesloten onder complement), moet de lege verzameling ook onderdeel zijn van $\mathcal{F}$. Het complement van de hele verzameling Ω is namelijk een lege verzameling.

7 f^c betekent het complement van de verzameling

8 Wat formeler: twee gebeurtenissen worden *disjunct* genoemd als ze niet gelijktijdig kunnen optreden: $A \cap B = \emptyset$. Iets uitgebreider, een rij gebeurtenissen is disjunct als voor $\forall(i,j), i \neq j \rightarrow A_i \cap A_j = \emptyset$.

Als we de bovenstaande definities toepassen op het werpen van een (eerlijke) dobbelsteen dan leidt dat tot de volgende invulling van het formele model:

- $\Omega = \{1,2,3,4,5,6\}$, de totale verzameling van gebeurtenissen,
- $\mathcal{F} = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\}\}$, waarbij iedere deelverzameling een elementaire gebeurtenis is,
- $\mathcal{P}$, de kansverdelingsfunctie:
  - $P(\Omega) = 1.0$: $p(1) + p(2) + p(3) + p(4) + p(5) + p(6) = 1.0$, waarbij de verdeling symmetrisch is: iedere kans heeft een waarde van $\frac{1}{6}$,
  - $P(1), P(2), P(3), P(4), P(5), P(6) \geq 0.0$ want alle kansen zijn $\frac{1}{6}$,
  - $P(1 \cup 2) = \frac{1}{6} + \frac{1}{6} = \frac{2}{6}$, de kans op 1 *of* 2 waarbij de beide gebeurtenissen disjunct zijn.

7.4. AANVULLENDE TERMEN EN NOTATIES

In deze paragraaf worden een aantal termen en notaties geïntroduceerd. De volgorde van introductie is meestal niet van belang. Je moet er gewoon bekend mee zijn.

De algemene notatie voor de kans op de uitkomst "$X = x$", de kans dat de stochastische variabele X de waarde x heeft, is $P(X = x)$, $p(X = x)$ of $P(x)$ dan wel $p(x)$. De laatste twee worden gebruikt als er geen verwarring kan bestaan over welke variabele nu bedoeld wordt. De veronderstelling is dan dat kleine x refereert aan de grote X.

Als de kans bekend is (er is een concreet getal dat de grootte van de kans weergeeft) dan wordt dat als volgt genoteerd (voor de stochastische variabele "worp van een dobbelsteen" met de naam X): $P(X = 1) = \frac{1}{6}$, $p(X = 1) = \frac{1}{6}$ of $P(x) = \frac{1}{6}$ dan wel $p(x) = \frac{1}{6}$.

Een gebeurtenis die slechts één uitkomst bevat, een *singleton*, noemen we een *elementaire gebeurtenis*.

Als de kansen alle mogelijke uitkomsten even groot is dan wordt de kansruimte *symmetrisch* genoemd. Dit is bijvoorbeeld het geval bij het gooien van een eerlijke dobbelsteen of een eerlijke munt. In het geval van de 10-grep uit het spel kaarten is dit duidelijk niet het geval.

7.5. JOINT DISTRIBUTION

Tot nu toe hebben we het vooral over 1 variabele gehad (ook de 10-greep uit het spel kaarten gaat eigenlijk over 1 variabele: het gaat immers over 1 10-greep). Het is natuurlijk ook mogelijk om over kansen en meerdere variabelen tegelijk te praten. In dat geval wordt gesproken over "gezamenlijke verdeling" ofwel "joint distribution". De kans dat de variabelen X en Y de waarde x en y hebben wordt genoteerd als $p(x,y)$, waarbij uiteraard alle eerder genoemde variaties ook toegestaan zijn. De komma heeft hier dus de betekenis van "en": de kans op x *en* y. Uiteraard kan dit ook voor meer dan twee variabelen: gewoon een kwestie van een extra komma en een nieuwe variabele erachter: $p(x,y,z)$.

7.5.1. UNCONDITIONAL PROBABILITY

Als de kansen op de waardes van X en Y niets met elkaar te maken hebben dan noemen we die kansen "onafhankelijk" of "independent". Als kansen independent zijn dan is de "gezamenlijke kans" ofwel "joint probability" het product van de kansen op x en y: $p(x,y) = p(x)p(y)$. Dit heb je waarschijnlijk vroeger wel gehad op de middelbare school: de kans om 2 keer achter elkaar 1 te gooien met één dobbelsteen is $\frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$

7.5.2. CONDITIONAL PROBABILITY

Soms (vaak?) hangen de kansen op de waardes van X en Y evenwel met elkaar samen. In dat geval is er sprake van "voorwaardelijke kans" oftewel "conditional probability". Conditional probability wordt als volgt genoteerd: $p(x|y)$, de kans op x *gegeven* y. In het geval van conditional dependency is de conditional probability:

$$p(x|y) = \frac{p(x,y)}{p(y)}$$

De formule komt ook vaak voor in een andere vorm: $p(x,y) = p(x|y) \cdot p(y)$ (gewoon van de eerste formule links en rechts vermenigvuldigen met $p(y)$ en de termen omdraaien).

Zo is de kans om uit een volledig spel een aas te trekken, gegegeven een hartenkaart:
$p(\text{aas}|\text{harten}) = \frac{p(\text{aas } en \text{ harten})}{p(\text{harten})} = \frac{\frac{1}{52}}{\frac{1}{4}} = \frac{1}{13}$

Als X en Y onafhankelijk zijn dan wordt de formule voor conditional probablity als volgt: $p(x|y) = \frac{p(x)p(y)}{p(y)} = p(x)$. Oftewel: als X en Y onafhankelijk zijn dan hangt de kans op x niet af van y.

Je hoeft de formule voor conditional probability niet zomaar als een gegeven aan te nemen. Hier volgt een afleiding van de formule.
Stel, je gooit met een eerlijke dobbelsteen. We zouden graag willen weten wat de kans is dat het aantal ogen groter of gelijk is aan 4, *gegeven* het feit dat er een even aantal ogen gegooid is. Dan is gebeurtenis Y het gooien van de dobbelsteen en het aantal ogen is even. Y bestaat dan uit de verzameling elementaire gebeurtenissen (singletons) $\{2,4,6\}$. Gebeurtenis X is dan de verzameling elementaire gebeurtenissen (singletons) $\{4,5,6\}$ dat het aantal ogen groter of gelijk is aan 4.
De kans op Y is dan:
$P(Y) = \frac{|Y|}{|S|} = \frac{|\{2,4,6\}|}{|\{1,2,3,4,5,6\}|} = \frac{3}{6} = \frac{1}{2}$ waarbij |iets| in de context van verzamelingen "het aantal elementen van iets" betekent.
Dan nu de kans dat X gebeurt, gegeven het feit dat Y gebeurt. Dat betekent dat zowel X als Y moeten gebeuren: $X \cap Y = \{4,5,6\} \cap \{2,4,6\} = \{4,6\}$
Uitgedrukt in een formule, uitgedrukt in aantallen:
$P(X|Y) = \frac{|X \cap Y|}{|Y|} = \frac{|\{4,6\}|}{|\{2,4,6\}|} = \frac{2}{3}$
De generalisatie van dit voorbeeld levert, als we zowel de teller als de noemer door $|S|$ (het totaal aantal mogelijkheden) delen zodat het kansen worden, het volgende op:
$P(X|Y) = \frac{|X \cap Y|}{|Y|} = \frac{\frac{|X \cap Y|}{|S|}}{\frac{|Y|}{|S|}} = \frac{P(X,Y)}{P(Y)}$

7.6. THEOREM OF TOTAL PROBABILITY

Stel dat dat de kans op x niet afhangt van 1 waarde van y maar van verschillende waardes van y. In dat geval is de kans op x de som van de kansen op x voor alle waardes van y:
$p(x) = \sum_y p(x|y)p(y)$.
Dit wordt de wet op de totale kans genoemd (law of theorem of total probability).

Een praktisch voorbeeld is het uitsorteren van rotte appels op een lopende band. Met behulp van een camera en OpenCV wordt iedere appel bekeken of deze rot is. Gemiddeld is 1 op de 100 appels rot. Maar helaas werkt het systeem niet helemaal foutloos. In 95% van de gevallen wordt een rotte appel correct gedetecteerd. In 1% van de gevallen wordt een niet-rotte appel helaas als rot gedetecteerd[9].

		Conditie	
		aanwezig	afwezig
Test	positief	True positive	False positive
	negatief	False negative	True negative

Tabel 7.1.: Sensitiviteit/specificiteit kruistable

De kans om een rotte appel te detecteren is dan dus de som van de kans om terecht een rotte appel te detecteren die daadwerkelijk rot is en de kans om onterecht een rotte appel te detecteren die eigenlijk niet rot is. Gegeven is dus:

- $p(rot) = 0.01$
- $p(detectie|rot) = 0.95$
- $p(detectie|\neg rot) = 0.01$

Dan wordt de berekening:

9 Het terecht detecteren van een uitkomst wordt sensitiviteit genoemd en het onterecht detecteren van een uitkomst specificiteit. Zie ook Wikipedia

$p(detectie) = p(detectie, rot) + p(detectie, \neg rot)$
Op grond van conditional probability kan dat worden geschreven als:
$p(detectie) = p(detectie|rot)p(rot) + p(detectie|\neg rot)p(\neg rot)$
Invullen van de waardes:
$p(detectie) = 0.95 * 0.01 + 0.01 * 0.99 = 0.02$

In totaal wordt dus 2% van de appels als rot gedetecteerd terwijl eigenlijk maar 1% daadwerkelijk rot is. Of dit erg is hangt maar erg van de omstandigheden af.

7.7. BAYES RULE

Thomas Bayes was een begin achtiende-eeuwse dominee in Engeland. Aan het einde van zijn leven ontwikkelde hij een interesse in statistiek naar aanleiding van een opmerking van de Engelse filosoof David Hume. Die stelde namelijk in een van zijn beroemste werken (An Enquiry Concerning the Principles of Morals) dat de kans dat mensen ten onrechte zeiden[10] dat ze de wederopstanding van Jezus hadden gezien veel groter was dan de kans dat de wederopstanding daadwerkelijk had plaats gevonden. Dat kon hij als dominee natuurlijk niet over zijn kant laten gaan en dus ging hij zich bezig houden met kansrekening en wel met het idee of je iets zinnigs kon zeggen van een kans op iets wat je niet direct zien maar waar je wel wat dingen van weet omdat je ze kan zien. Helaas ging hij eerder dood dan dat hij zijn manuscript kon publiceren. Gelukkig werd dat manuscript na zijn dood voorgelezen zodat het niet verloren is gegaan. Nadeel is wel dat het wereldberoemd is geworden en dat nu overal ter wereld studenten aan het zwoegen zijn op zijn theorie...

Zijn werk is beroemd geworden onder de naam "Theorema van Bayes" (Bayes' theorem ofwel Bayes rule)[11].

De Bayes rule luidt als volgt: $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$

Hierbij is

- $p(x|y)$ de voorwaardelijke kans, "posterior probability", op x gegeven y, het bijgestelde geloof naar aanleiding van bewijs/metingen.
- $p(y|x)$ de voorwaardelijke kans, "likelihood", op y gegeven x,
- $p(x)$ de zogenaamde "prior probability distribution", het initiële geloof in X,
- $p(y)$ de data, het bewijs/de metingen.

De formuler van Bayes is op basis van de conditional probability overigens makkelijk af te leiden.

1. $p(x|y) = \frac{p(x,y)}{p(y)} \Leftrightarrow p(x|y)p(y) = p(x,y) \Leftrightarrow p(x,y) = p(x|y)p(y)$
2. $p(y|x) = \frac{p(y,x)}{p(x)} \Leftrightarrow p(y|x)p(x) = p(y,x) \Leftrightarrow p(y,x) = p(y|x)p(x)$
3. Omdat $p(x,y) = p(y,x)$:
   $p(x|y)p(y) = p(y|x)p(x) \Leftrightarrow p(x|y) = \frac{p(y|x)p(x)}{p(y)}$

Een praktisch voorbeeldje van Bayes rule.

Drugs op het werk zijn vaak een gevaar voor de gebruiker en zijn omgeving. Daarom hanteren veel bedrijven een zero-tolerance beleid op het gebied van drugs: als je onder invloed van drugs op je werk verschijnt word je ontslagen. Daarom ontwikkel je een apparaat waarmee je kunt testen of een werknemer drugs heeft gebruikt: gat in de markt! Het apparaat blijkt in de de praktijk een sensitiviteit van 97% en een specificiteit van 95% te hebben. Met andere woorden, in 97% van de gevallen wordt iemand die onder invloed is van drugs positief getest op drugs, in 95% van de gevallen wordt iemand die niet onder de invloed is van drugs daadwerkelijke negatief getest op drugs. Uit grootschalig onderzoek blijkt dat 0.5% van de werknemers regelmatig onder invloed van drugs op het werk verschijnt. Nu meet je bij een werknemer dat hij onder invloed is van drugs. Hoe groot is nu de kans dat de werknemer daadwerkelijk onder invloed van drugs is?

Dat leidt tot de volgende berekening.

Gevraagd: $p(high|detectie)$.

10 Als de bijbel een historisch kloppende beschrijving van de gebeurtenissen is dan is op Wikipedia een overzicht van mensen te vinden die in ieder geval de wederopstanding lijken te hebben gezien.

11 Overigens is de naamgeving, net als bij Gauss-eliminatie, wat merkwaardig: de regel zelf is geformuleerd door Pierre-Simon Laplace (daar is hij weer!) die pas twaalf was toen Bayes dood ging en niet door Bayes zelf...

Gegeven:

- $p(high) = 0.005$
- $p(detectie|high) = 0.97$
- $p(\neg detectie|\neg high) = 0.95$ (oftewel $p(detectie|\neg high) = 0.05$)

Op grond van the law of total probability is de kans op detectie:
$p(detectie) = p(detectie, high) + p(detectie, \neg high)$
$p(detectie) = p(detectie|high)p(high) + p(detectie|\neg high)p(\neg high)$
$p(detectie) = (0.97 \times 0.005) + (0.05 \times 0.995) = 0.055$

De regel van Bayes zegt dan:
$p(high|detectie) = \frac{p(detectie|high) \cdot p(high)}{p(detectie)} = \frac{0.97 \times 0.005}{0.055} = 0.088$
Of je iemand op basis van deze test mag ontslaan is maar zeer de vraag...

7.8. TRANSITION MATRIX

Tot nu toe hebben we ons bezig gehouden met simpele voorwaardelijke kansen: wat is de kans op X gegeven Y. Maar vaak willen we iets ingewikkelders weten: wat is de kans op X gegeven de achtereenvolgende gebeurtenissen $Y_0, Y_1, Y_2, Y_n \ldots$. Een andere manier om dit te beschrijven is: wat is de kans op X als gevolg van de toestandsverandering door Y_0, gevolgd door de toestandsverandering van Y_1, gevolgd door de toestandsverandering van Y_1, enzovoorts.

In dit hoofdstuk gebruiken we een voorbeeld van een simpel weermodel. Het weer kent slechts drie varianten: het is zonnig, bewolkt of het regent. Wat het weer morgen wordt hangt af van het weer van vandaag, het weer van overmorgen van het weer van morgen en zo verder. Als het vandaag zonnig is dan is de kans dat het morgen weer zonnig tamelijk groot, 0.8. De kans dat het morgen bewolkt is is aanzienlijk kleiner, 0.2 terwijl de kans dat het morgen regent nihil (0.0) is. Als het vandaag bewolkt is dan is de kans op morgen zon 0.4, weer bewolkt 0.4 en regen 0.2. Voor vandaag regen geldt zon 0.0, bewolkt 0.6 en weer regen 0.2.

Het probleem dat we gaan bekijken is: wat is het weer over n dagen als het vandaag zonnig, bewolkt of regenachtig is.

7.8.1. TRANSITIE-TABEL

Het weermodel is uitstekend in een transitietabel zoals tabel 7.2 weer te geven.

		morgen		
		zonnig	bewolkt	regen
vandaag	zonnig	0.8	0.2	0.0
	bewolkt	0.4	0.4	0.2
	regen	0.2	0.6	0.2

Tabel 7.2.: Weermodel: tabel

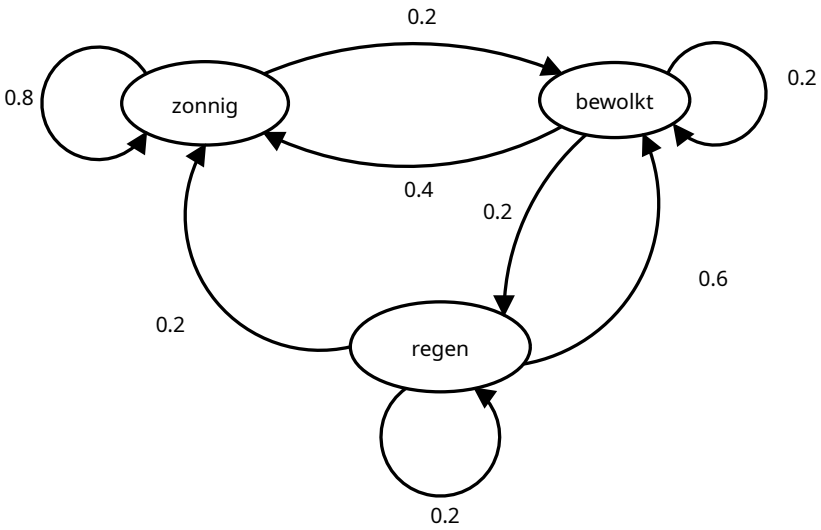
In de tabel 7.2 staat van boven naar beneden de huidige toestand (de "van toestand") en staan van links naar rechts de toestanden waar naar toe gegaan kan worden (de "naar toestand"). In de cellen staan vervolgens de kansen dat die transities genomen worden. Praktisch gezien staat er dus bijvoorbeeld dat als het vandaag regent de kans dat het morgen bewolkt is 0.6 is, of dat als het vandaag zonnig is dat de kans op morgen regen 0.0 is. De kansen tellen horizontaal op tot 1.0.

Uiteraard kunnen de assen van de tabel ook omgewisseld worden. In dat geval wordt morgen vandaag, vandaag morgen , de rijen kolommen, de kolommen rijen [12] en tellen de kansen verticaal op tot 1.0.

7.8.2. TRANSITIE-TABEL ALS FSM

In figuur 7.1 wordt het weermodel weergegeven met behulp van een zogenaamde "finite state machine"(FSM) waarbij de transities voorzien zijn van de kans dat die transitie genomen wordt.

12 Dit alles vrij naar Drs. P's geweldige lied "Veerpont"



Figuur 7.1.: Weermodel: finite state machine

7.8.3. TRANSITIE-TABEL ALS MATRIX

Zowel de tabel als de FSM kunnen in een matrix weergegeven worden zonder verlies van informatie. Een dergelijke matrix wordt een transitiematrix (transition matrix)[13] genoemd. Een dergelijk matrix kan op twee manieren worden opgeschreven.

Een “right stochastic matrix” is een transitiematrix waarbij verticaal (van boven naar beneden) de “van”-toestanden en horizontaal de “naar”-toestanden staan. In dat geval is de som van iedere rij 1.0 is.

Een “left stochastic matrix” is een transitiematrix waarbij verticaal (van boven naar beneden) de “naar”-toestanden en horizontaal de “van”-toestanden staan. In dat geval is de som van iedere kolom 1.0 is.

Van ieder right stochastic matrix is een left stochastic matrix te maken door de transpose van de eerste te nemen. En omgekeerd.

Voor ons weermodel ziet de right stochastic matrix (transitiematrix) (die in een dergelijk geval vaak P wordt genoemd (van **P**robability matrix)), er als volgt uit:

$$P = \begin{bmatrix} 0.8 & 0.2 & 0.0 \\ 0.4 & 0.4 & 0.2 \\ 0.2 & 0.6 & 0.2 \end{bmatrix}$$

De left stochastic matrix (transitiematrix) (ook hier P genoemd) is dan als volgt:

$$P = \begin{bmatrix} 0.8 & 0.4 & 0.2 \\ 0.2 & 0.4 & 0.6 \\ 0.0 & 0.2 & 0.2 \end{bmatrix}$$

Volgens Wikipedia wordt in de Engelstalige literatuur veelal gebruik gemaakt van een right stochastic matrix. En hij leest ook wat makkelijker: de matrix staat in een logische “van-naar”-vorm. Een left stochastic matrix levert wel een compactere schriftelijke notatie op (zie verderop).

13 Om het feest compleet te maken worden ook de volgende benamingen gebruikt: kansmatrix (probability matrix), stochastische matrix (stochastic matrix) of Markov matrix.

7.8.4. REKENEN MET HET WEER

Stel dat het vandaag regent en je wil weten wat de kans is dat het over 3 dagen zonnig is:
p("zonnig over 3 dagen"|"vandaag regent het").
Hoe zouden we dit kunnen berekenen?

7.8.4.1. MET EEN GRAFISCHE PADEN-BOOM

Het "theorem of the total probability" zegt in dit geval dat de kans op over 3 dagen zonnig, gegeven het feit dat het vandaag regent, de som van de kansen is van alle mogelijke paden die van vandaag regent het naar het is zonnig over 3 dagen leiden. In figuur 7.2 staan alle mogelijke paden vanuit vandaag regen die in 3 stappen gedaan kunnen worden. Uit de paden en de bijbehorende kansen blijkt dat de kans op zonnig in 3 stappen gegeven het feit dat het nu regent 0.544 is. Voor alle mogelijk paden die beginnen in "vandaag regent het" en die eindigen "vandaag is het zonnig" wordt de kansen van een dergelijk pad uitgerekend door de kansen van ieder onderdeel van het pad te vermenigvuldigen en tot slot opgeteld.

7.8.4.2. MET EEN MATRIX

Bij grote FSM-en of lange paden in een FSM is een formalisme van het uittekenen van alle mogelijke paden onbegonnen werk. Gelukkig is er een makkelijkere manier om de kansen uit te rekenen: met behulp van matrices (Duh! Waar zouden we zijn zonder matrices...). Zonder ons verder te vermoeien met een afleiding of bewijs volgt hier de manier om dat te doen.

Zoals al eerder gezegd wordt traditiegetrouw in veel literatuur de transitiematrix aangegeven met P. Een state vector wordt opvallend vaak met π aangegeven. Als de matrix P een right stochastic matrix is dan is π een rij-vector waarbij de elementen in de vector in dezelfde volgorde staan als de kolommen van de transitiematrix. Als de matrix P een left stochastic matrix is dan is π een kolom-vector waarbij de elementen in de vector in dezelfde volgorde staan als de rijen van de transitiematrix. Gelukkig zijn de meest transitie-matrices vierkant waarbij we de rijen en kolommen in dezelfde volgorde houden.

Voor een right stochastic matrix geldt: $\pi_n = \pi_0 P^n$

Voor een left stochastic matrix geldt: ${\pi_n}^T = (P^T)^n {\pi_0}^T$[14]

De state vector na n transities is dus de state vector van het begin vermenigvuldigd met de transitiematrix tot de macht van het aantal stappen.

Hierbij is de π een kolom- of rij-vector met daarin de kans voor ieder state dat je in die state bent. Stel dat er drie mogelijke states A, B en C zijn. In dat geval betekent de row-vector $\{0.0, 0.0, 1.0\}$ dat de kans dat we in state C zijn 1.0 is: we zijn er dus voor 100% van overtuigd dat we in state C zijn. Een row-vector van $\{0.33, 0.33, 0.33\}$[15] betekent dus dat we volstrekt niet weten in welke state we zijn (een uniforme of homogene kansverdeling).

Voor ons weervoorbeeld met een right stochastic transitiematrix levert dit het volgende op:

$$\begin{bmatrix}0.0 & 0.0 & 1.0\end{bmatrix} \begin{bmatrix}0.8 & 0.2 & 0.0\\ 0.4 & 0.4 & 0.2\\ 0.2 & 0.6 & 0.2\end{bmatrix}^3 = \begin{bmatrix}0.544 & 0.344 & 0.112\end{bmatrix}$$

En met een left stochastic transitiematrix:

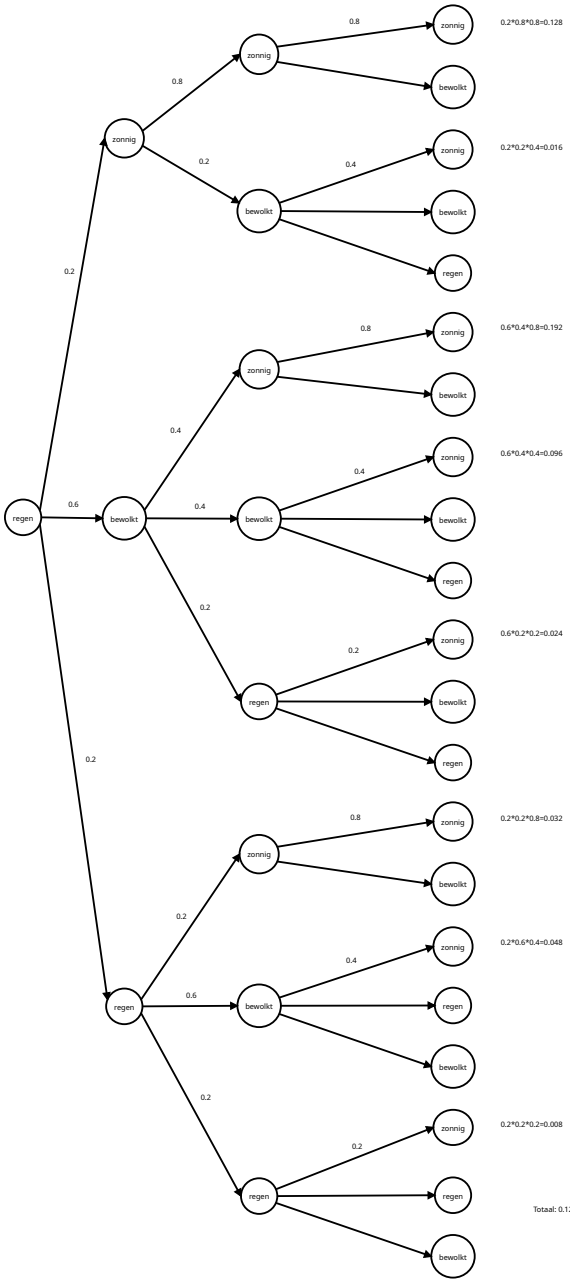
$$\begin{bmatrix}0.8 & 0.4 & 0.2\\ 0.2 & 0.4 & 0.6\\ 0.0 & 0.2 & 0.2\end{bmatrix}^3 \begin{bmatrix}0.0\\ 0.0\\ 1.0\end{bmatrix} = \begin{bmatrix}0.544\\ 0.344\\ 0.112\end{bmatrix}$$

De kans op zonnig is dus inderdaad 0.544. En voor de pessimisten onder ons: de kans dat het nog steeds regent is in verhouding tamelijk klein: 0.112. Na regen komt dus inderdaad zonneschijn!

7.8.5. STEADY-STATE

Vaak wil je weten wat de kans op een toestand op de (erg) lange termijn is. Als de toestand op de lange termijn niet meer (wezenlijk) verandert dan wordt dat de "steady state" genoemd. Een logische interpretatie van de steady-state is "de gemiddele state op de lange termijn". Voor een systeem in steady-state geldt:

14 Dit volgt uit de matrix-rekenregel $(AB)^T = B^T A^T$
15 Ja, ja, er ontbreekt 0.01.


regen
0.2
zonnig
0.8
zonnig
0.8
zonnig
bewolkt
0.2
zonnig
0.4
bewolkt
bewolkt
regen
0.6
bewolkt
0.4
zonnig
0.8
zonnig
bewolkt
0.4
zonnig
0.4
bewolkt
bewolkt
regen
0.2
zonnig
0.2
regen
bewolkt
regen
0.2
zonnig
0.8
zonnig
bewolkt
0.2
zonnig
0.4
regen
regen
bewolkt
0.6
bewolkt
0.2
zonnig
0.2
regen
regen
bewolkt
0.2*0.8*0.8=0.128
0.2*0.2*0.4=0.016
0.6*0.4*0.8=0.192
0.6*0.4*0.4=0.096
0.6*0.2*0.2=0.024
0.2*0.2*0.8=0.032
0.2*0.6*0.4=0.048
0.2*0.2*0.2=0.008
Totaal: 0.128+0.016+0.192+0.096+0.024+0.032+0.048+0.008 = 0.544

$\pi_n = \pi_n P$.
Uit de formule blijkt dus dat een extra transitie geen verandering in de state-vector oplevert.

Opvallend vaak is die steady-state te berekenen maar soms niet. In dit dictaat wordt verder niet op ingegaan op gevallen dat de steady-state niet te berekenen is.

De steady-state kan zowel analytisch als numeriek berekend worden.

7.8.5.1. ANALYTISCHE STEADY-STATE

Als je dit analytisch wil uitrekenen, dan gaat dat als volgt. Ik doe dit voor met een transitiematrix van 3×3. Voor matrices van andere groottes gaat het op een vergelijkbare manier.

Gegeven zijn:

- de right stochastic transitiematrix, $P = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$
  waarbij $a, b, c, d, e, f, g, h, i \in \mathbb{R}$ de bekende transitie-kansen zijn,
- de state rij-vector $\pi = \{X, Y, Z\}$,
  waarbij X, Y en Z de onbekende waardes van de steady state variabelen zijn.

Zoals al eerder aangegeven, in de steady state geldt dat de state vector niet wijzigt als gevolg van een nieuwe transitie, dus dan geldt: $\pi_n = \pi_n P$. Uitgeschreven in de volledige matrix/state vector-vorm wordt dat: $\begin{bmatrix} X & Y & Z \end{bmatrix} = \begin{bmatrix} X & Y & Z \end{bmatrix} \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$

Als we de rechterkant uitvermenigvuldigen dan ontstaat het volgende stelsel van lineaire vergelijkingen:

$$aX + dY + gZ = X$$
$$bX + eY + hZ = Y$$
$$cX + fY + iZ = Z$$

Bovendien geldt ook nog eens dat $X + Y + Z = 1.0$. De kansen van de toestanden tellen immers op tot 1.0.

Nu hebben we een stelsel van 4 lineaire vergelijkingen met slechts 3 variabelen. Een dergelijk stelsel is in beginsel op te lossen: zie hoofdstuk 4.

7.8.5.2. NUMERIEKE STEADY STATE

Je kunt de steady state ook numeriek berekenen. Bij een numerieke berekening voeren we gewoon een of ander algoritme uit, net zolang tot we op een gewenste graad van nauwkeurigheid zitten (of tot de uitkomst niet meer (merkbaar) verandert.

Als je het numeriek wil uitrekenen dan ziet dat er ongeveer zo uit:

```
Matrix transitieMatrix;
Matrix powerMatrix = identity;
Vector startState;
Vector currentState;
Vector previousState;

while true
        powerMatrix = powerMatrix * transitieMatrix;
        currentState  =  startState * powerMatrix;
        if (previousState == currentState) break;
        previousState = currentState;

currentState is nu de steady state
```

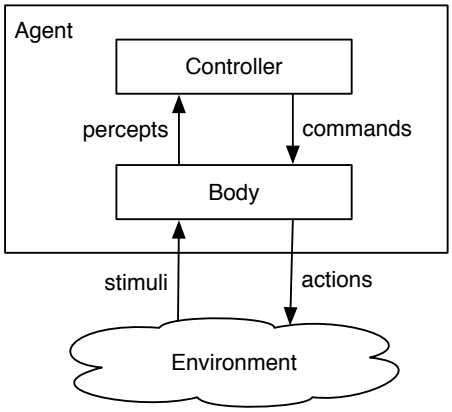
De steady state vector van het weer-voorbeeld is $\begin{bmatrix} 0.643 & 0.286 & 0.071 \end{bmatrix}$ na 8 iteraties op 3 cijfers achter de komma nauwkeurig.

8. ROBOT-MODEL

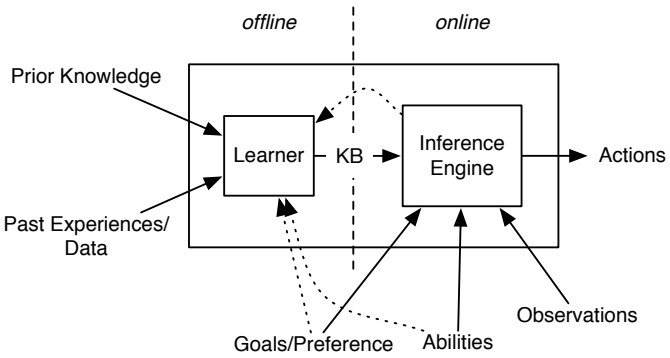
Veel van de inhoud van dit hoofdstuk komt uit de volgende boeken:

- "Artificial Intelligence: Foundations of Computational Agents (second edition)" (Poole en Mackworth 2017) waarvan een gratis versie hier online te lezen is.
- "Probabilistic robotics" (Thrun, Burgard en Fox 2005) waarvan helaas geen gratis versies beschikbaar zijn.

8.1. INLEIDING



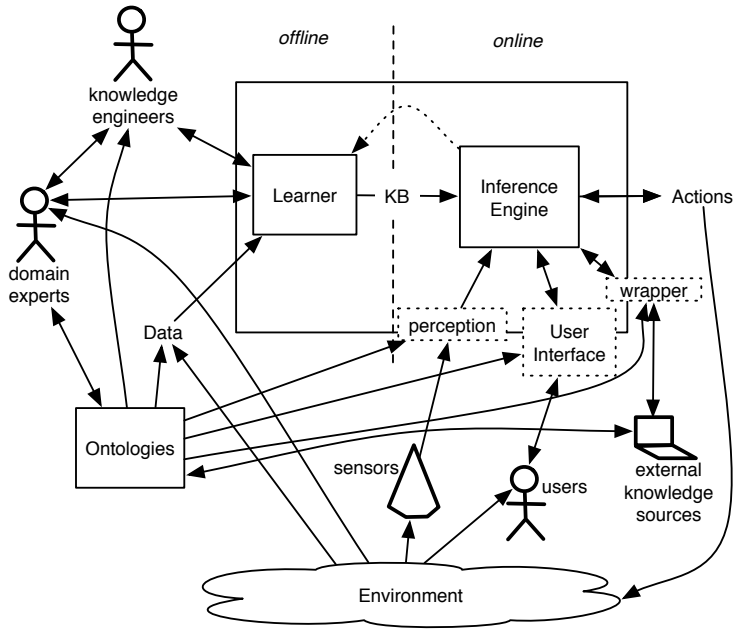
Figuur 8.1.: Algemene robotmodel



Figuur 8.2.: Algemene robotmodel: controller

8.2. TERMEN EN NOTATIES

In deze paragraaf worden een aantal termen en notaties geïntroduceerd die veel gebruikt worden in het robotica-veld. Helaas is het wel zo dat veel termen "overloaded" zijn. Robotica is een samenraapsel van een veelheid van vakgebieden en veel van die vakgebieden gebruiken dezelfde termen maar met een andere betekenis. Soms zijn de verschillen vooral interessant voor de fijnproever maar soms verschillen ze fundamenteel. Maar zoals in de inleiding van dit dictaat al is aangegeven: in dit document worden voor



Figuur 8.3.: Algemene robotmodel: details

de meeste begrippen "informele" en "working knowledge" definities gegeven. Het gaat vooral om het begrip en niet de exacte definitie. Veel van de wat technischere termen en notaties in deze paragraaf komen uit Thrun, Burgard en Fox.

8.2.1. STATE

De wereld met daarin de robot bevindt zich op een bepaald moment in tijd in een bepaalde toestand, de *state* x_t, waarbij x voor de toestand staat en t voor het moment in tijd. Een state wordt bepaald door variabelen en hun waardes in de wereld. De state is daarom een (kolom-) vector, de *state vector*. Welke concrete variabelen relevant zijn voor de state van een wereld is niet zo maar te zeggen: dit hangt sterk af van de aard van de wereld. Twee states zijn verschillend als de variabelen die de state beschrijven (voldoende) verschillende waarden hebben.

8.2.1.1. WAARNEMEN VAN DE STATE DOOR DE ROBOT

De robot neemt de *world* of *environment* waar door middel van sensoren. Deze zijn *noisy*. Noisy wil zeggen dat de metingen van de sensoren niet absoluut juist zijn maar dat ze er naast kunnen zitten. Hoeveel ze er naast kunnen zitten hangt van de sensoren af. De nauwkeurigheid van sensoren wordt meestal gegeven in termen van gemiddeldes en standaard-deviaties: een sensor meet "iets" met een bepaalde gemiddelde en standaard-deviatie. Eén meting waarbij stimuli vertaald worden naar percepts wordt een *measurement* of *observation* genoemd.

De data van de measurements in de tijd wordt de *measurement data* genoemd en wordt genoteerd als z_t, waarbij z staat voor de (kolom)vector met measurement data en t staat voor het tijdstip waarop de measurement is gedaan.

8.2.1.2. VERANDEREN VAN DE STATE DOOR DE ROBOT

De robot beïnvloedt de *environment* door middel van actuatoren. Ook deze zijn in zekere mate *noisy*, *unpredictable*. Ook hiervoor geldt dat de noise in de verandering van de wereld door de actuatoren wordt gegeven in termen van gemiddelden en standaard-deviaties: een actuator doet "iets" zodat een effect met een gemiddelde en een bepaalde standaard-deviatie wordt bereikt. Eén actie waarbij commands vertaald worden naar actions en de actions worden uitgevoerd wordt een *control action* genoemd.

De data met betrekking tot de control actions in de tijd wordt de *control data* genoemd en wordt geno-

teerd als u_t, waarbij u staat voor de (kolom)vector met control data en t staat voor het tijdstip waarop de control action is gedaan.

8.2.1.3. HET BELIEF VAN DE ROBOT

De consequentie van het voorgaande is dat er dus zowel een *feitelijke toestand* is als een *perceptie van de toestand* door de robot. Idealiter zijn de feitelijke toestand en de perceptie van de toestand identiek. In de praktijk is dit evenwel niet het geval. Dat betekent dat de robot voor wat betreft zijn eigen handelen uit moet gaan van zijn eigen model van de (toestand van de) wereld: de robot heeft immers niet anders. Het model van de robot wordt *belief* genoemd: de robot "gelooft" dat de toestand van de wereld is zoals deze de toestand via sensormetingen, actuator updates en inference heeft opgebouwd.

8.2.1.4. STATE VARIABELEN

Hoewel er dus niet zo maar een uitputtende verzameling variabelen te bepalen is die relevant is voor het beschrijven van de state van de wereld is er wel iets meer over te zeggen. De variabelen die de state van de wereld beschrijven zijn onder te verdelen in twee soorten: *statische variabelen* en *dynamische variabelen*. De statische variabelen zijn die variabelen die in de tijd niet veranderen. De dynamische variabelen veranderen in de tijd wel.

De statische variabelen kunnen ook weer onderverdeeld worden in twee categorien: statische variabelen die offline als prior knowledge door de robot-ontwikkelaar "in de robot gestopt" worden en statische variabelen waarvan de robot online via inference besluit dat ze statisch zijn. Een typische statische variabele is een kaart van de wereld met daarin objecten als muren, deuren, wegen, obstakels enzovoorts.

Dynamische variabelen zijn vooral die objecten die door sensoren gedetecteerd worden: objecten als andere robots, mensen, auto's enzovoorts. Van dergelijke objecten zijn veelal de grootte, de locatie, de snelheid en de richting van bewegen relevant. Sommige variabelen zijn dynamisch omdat actoren ze doelbewust veranderen, e.g. de plaats van een robot of mens verandert omdat deze dat willen. Maar sommige variabelen veranderen autonoom: tijd is hiervan een voorbeeld, maar ook de valsnelheid van een vallend object.

8.2.2. CONFIGURATION

De state van een robot wordt in het algemeen beschreven in met behulp van een *configuration*. De configuration van een robot beschrijft de toestand van alle relevante componenten van de robot zelf, in meestal een vector met daarin getallen als representatie voor de toestand. Welke componenten van de robot relevant zijn hangt natuurlijk van de robot af. Voor een robot-arm zijn dit bijvoorbeeld alle hoeken van de arm en de toestand van de gripper. Voor een zelfrijdende robot zou dit, naast iets triviaal als de positie van de robot in de wereld, de resterende rij-tijd relevant kunnen zijn. De verzameling van alle mogelijke configurations wordt *configuration space* of korter, *C-space* genoemd.

De kern van intelligente autonome robots bestaat eigenlijk uit het bepalen van de toestand van de wereld en de eigen configuration om vervolgens een pad van configurations in de C-space te zoeken zodat de goals/preferences gerealiseerd worden.

Binnen de robotica is er ook een wat beperkter begrip van een configuration en C-space die uit de klassieke mechanica stamt. Een configuration is hierbij een beschrijving van alle posities die de componenten van een robot beschrijft waarbij kinematica een rol speelt. Hierbij spelen begrippen als plaats, snelheid en versnelling een belangrijke rol.

8.2.3. POSE

De verzameling variabelen die je nodig hebt om de plaats en richting van een robot in de wereld vast te leggen wordt *pose* genoemd.

In theorie heeft een (star) lichaam in een 3D-wereld 6 vrijheidsgraden: het heeft in een zeker assenstelsel een positie (x, y, z) en een rotatie om respectievelijk om de lengte-as, de breedte-as en staande-as.[1]. Je hebt dus 6 variabelen nodig om *alles* met betrekking tot de plaats en richting van een robot in de wereld

1 Een rotatie om respectievelijk om de lengte-as, de breedte-as en staande-as heten in het Nederlands *rollen*, *stampen* en *gieren* en in het Engels *roll*, *pitch* en *yaw*. De Wikipedia-links gaan naar mooie bewegende plaatjes voor een vliegtuig.

vast te leggen. Hoeveel van die 6 variabelen je nodig hebt voor een specifieke robot hangt van de robot af. Voor een vliegende drone heb je ze alle 6 nodig. Maar ook voor een eindeffector (gripper) van een robot heb je ze alle 6 al vrij snel nodig. Voor een eenvoudige robot die zich in het platte vlak bevindt en voorbeweegt kan evenwel volstaan worden met 3 variabelen: de positie met (x, y) en een richting van zijn neus, de yaw.

8.3. STATE ONTWIKKELING

De states van de wereld en de robot ontwikkelen zich in de tijd: de dynamische variabelen veranderen immers in tijd. Sommige variabelen veranderen autonoom (e.g. de tijd), terwijl andere doelbewust door acties van de robot veranderen (e.g. zijn plaats). De robot bepaalt zijn acties in beginsel door de toestand en zijn goals/preferences: "als dit de state is dan is die action noodzakelijk om dat goal te bereiken".

Als een state x_t alle informatie bevat die noodzakelijk is voor het voorspellen van de toekomst, i.e. de volgende state, dan wordt een dergelijke state een *complete state* genoemd. De toekomst mag stochastisch of deterministisch zijn. Als er geen waardes van variabelen uit eerdere states x_{t-n} gebruikt worden bij de voorspeling[2] dan wordt een dergelijk proces een Markov chains genoemd. In dit dictaat behandelen we alleen Markov chains. Markov chains worden grafisch weergegeven als een FSM waarbij de transities voorzien zijn van de kans dat die transitie wordt genomen. Die hebben we al eerder gezien: zie figuur 7.1 voor een voorbeeld. Zie voor meer informatie over Markov chains in de volgende paragraaf.

In de praktijk is het niet doenlijk en niet noodzakelijk om een Markov chain van complete states te hebben. Veelal is het mogelijk om een subset van de state variabelen te kiezen die toch voldoende voorspellend vermogen hebben voor de volgende state, vooral ook omdat het systeem toch stochastisch is. In dat geval worden de states, heel verrassend, *incomplete states* genoemd. En hoewel Markov chains van/met incomplete states niet voldoen aan de theoretische voorwaarden voor berekeningen blijken ze toch opvallend goed te werken.

Een andere versimpeling die we aannemen is dat we te maken hebben met een *discrete state space*. Voor iedere state variabele is een eindig aantal mogelijkheden, in tegenstelling tot een *contineous state space* of eventueel een *hybrid state space*. Een belangrijke gediscretiseerde variabele hierbij is de *tijd*, i.e alle interessante *events* doen zich voor op een *discrete time step* $t = 1, 2, 3, \cdots$.

De laatste versimpeling van de state ligt bij de interacties. Measurements en control actions vinden in het echt tegelijkertijd plaats, maar om er over te kunnen redeneren doen we alsof deze niet tegelijk plaatsvinden. Hierbij gaan we er meestal van uit dat er eerst een update gedaan wordt voordat er een measurement gedaan wordt. Als dit alternerend gedaan wordt, i.e. telkens om en om een update en een measurement, dan kunnen we er vanuit gaan dat een verandering in measurement het gevolg is van een gedane update. En die kennis kunnen we weer gebruiken bij een volgende update.

8.3.1. MARKOV CHAINS

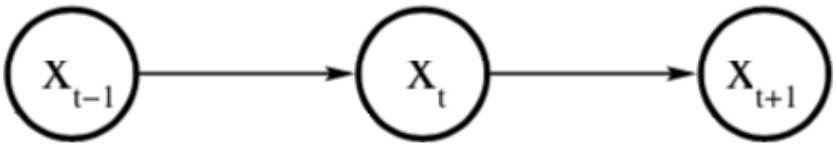
Zoals gezegd is een Markov-keten (Markov chain) een stochastisch model dat een systeem van sequentiële toestanden beschrijft waarbij de kans op een transitie van de toestand x_t naar de toestand x_{t+1} alleen maar afhangt van de toestand x_t. Een Markov chain met een aftelbare (en eventueel oneindige) sequentie van states waarbij de transities in discrete tijdstappen gaan wordt een discrete-time Markov chain (DTMC) genoemd. Een continue-tijd Markov chain wordt een continuous-time Markov chain (CTMC) genoemd. Hoewel de meeste robot-problemen feitelijk CTMCs zijn worden deze meestal gediscretiseerd tot DTMCs. De berekeningen voor een DTMC hebben een lagere complexiteit en zijn daarom in real-time veelal bruikbaarder. In dit dictaat gaan we verder alleen in op DTMCs.

Het gegeven dat voor een Markov chain geldt dat de transitie van state x_t naar de toestand x_{t+1} alleen maar afhangt van de toestand x_t wordt ook wel "Markov property" genoemd. Een andere term die hiervoor gebruikt wordt is dat Markov chains *memoryless* zijn: alleen het heden telt en niet het verleden.

Markov chains kunnen grafisch weergegegeven worden. In de figuren 8.4 en 8.6 staan twee varianten van een Markov chain weergegeven.

In figuur 8.4 staat een gewone Markov chain in een fully observable world. In dit model zien we dat een Markov chain zich vanuit x_{t-1} van x_t naar x_{t+1} ontwikkelt. De toestand van de wereld van de wereld is

2 Als er toch gegevens uit het verleden nodig zijn dan is een praktische oplossing deze op te nemen in de variabelen van de huidige staat.

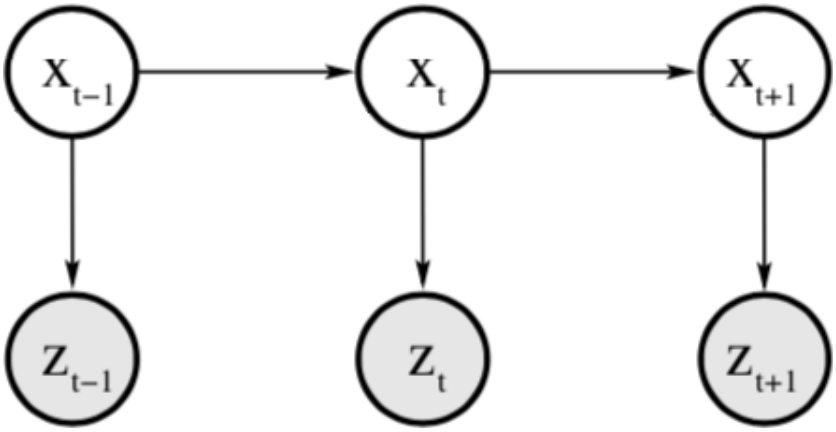


Figuur 8.4.: Markov-keten

direct en volledig zichtbaar. Dat wil in dit verband zeggen dat weliswaar de *transitie* naar de toestand probabilistisch is maar de toestand voor 100% zichtbaar en niet probabilistisch is: je weet zeker dat je in die toestand bent.

De kans dat je in een toestand komt is dus $p(x_t|x_{t-1}, P)$: de kans dat je op tijdstip t daadwerkelijk in toestand x komt is afhankelijk van de vorige toestand en de transition matrix P. Let erop dat de kans op een toestand op tijdstip t genoteerd wordt $p(x_t)$ maar dat die eigenlijk ook uitgedrukt kan worden als $p(\pi_t)$: de kans dat de state vector $\pi_t = [a, b, c, \ldots]$ een bepaalde waarde heeft. Dit geldt uiteraaard ook voor de vorige toestand x_{t-1}.

Een Markov chain waarbij P onafhankelijk is van t wordt *homogeen* genoemd. Het is ook mogelijk dat de transition matrix P wel afhankelijk is van t. In dat geval is de kans $p(x_t = \pi_{t-1}, P_{t-1})$. Een voorbeeld van een Markov chain die afhankelijk van de tijd is, zou een Markov chain kunnen zijn die de kans op kapot gaan van een apparaat modelleert: het is aannemelijk dat de kans op kapot gaan bij oude apparaten anders is dan bij nieuwe apparaten.



Figuur 8.5.: Hidden Markov model

In figuur 8.5 staat een *Hidden Markov Model* (HMM) of *dynamic Bayes network* (DBN) weergegeven [3] in een partially observable world. De concrete toestanden zijn niet direct zichtbaar maar moeten worden afgeleid van measurements van de state. In een HMM zijn niet alleen de transities probabilistisch maar zijn ook de states probabilistisch.

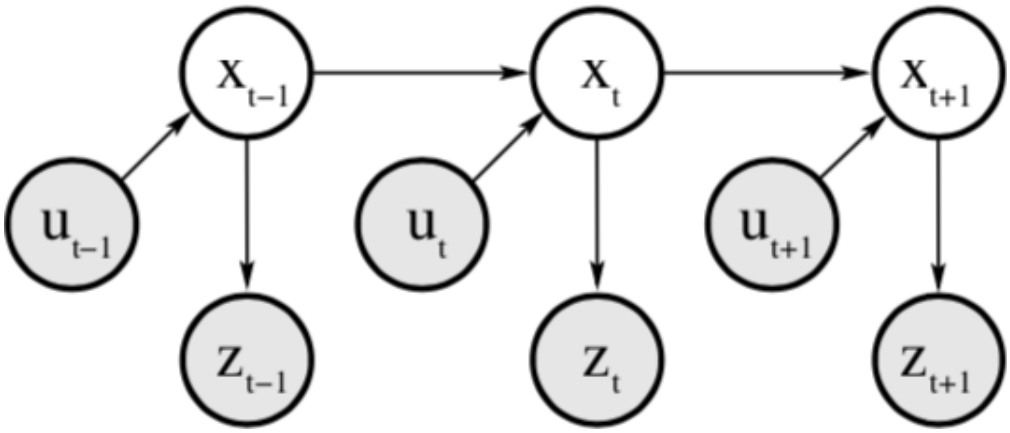
De kans dat je in een toestand bent is hier namelijk $p(x_t) = p(z_t|x_t)$: de kans dat je op tijdstip t daadwerkelijk in toestand x_t bent is de kans dat je measurements gegeven de toestand correct zijn. Met andere woorden, hoe waarschijnlijk is het dat je die measurements doet terwijl je in die toestand bent.

In figuur 8.6 staat een weer HMM in een partially observable world maar nu aangevuld met control updates. Dit sluit goed aan bij het algemene robotmodel waarbij de robot de wereld (en daarmee dus de state) beïnvloedt met control actions.

Hoewel er voor de updates u en measurements z verschillende tijdstippen gekozen had kunnen worden is er in dit formalisme voor gekozen dit niet niet te doen: de state x_t, de update u_t en measurements z_t hebben hetzelfde tijdstip. Maar omdat het een directed graph is moet je de keten als volgt lezen: eerst wordt op tijdstip t de update u_t gedaan, dan wordt de toestand x_t bereikt en tot slot wordt de meting z_t gedaan. Dit had ook allemaal met verschillende tijdtippen t aangegeven kunnen worden maar dat is nu niet gedaan.

De kans om in x_t te komen wordt bepaald door de voorgaande state x_{t-1} en door de update u_t: $p(x_t|x_{t-1}, u_t)$.

3 Op zich zijn deze twee sterk gerelateerde concepten niet identiek. Een HMM is een bijzondere vorm van een DBN waarin de states lineair in de tijd geindexeerd zijn.



Figuur 8.6.: Hidden Markov model met controls

Deze kans wordt de *state transition probability* genoemd.

Als je in x_t een measurement z_t doet dan is die measurement afhankelijk van de state van de wereld op dat moment: $p(z_t|x_t)$, de kans dat je een measurement z_t doet gegeven de state x_t. Deze kans wordt de *measurement probability* genoemd. Opgesloten in de measurement probability zit het idee van complete state: je hebt voldoende aan de state om de measurement te voorspellen. Dit veronderstelt natuurlijk wel dat je alle relevante variabelen kunt meten (of berekenen gegeven de metingen).

Eerder is al uitgelegd dat er een verschil zit tussen de daadwerkelijk state en het belief van de robot in wat de toestand is. Dat kwam door de probabilistische aard van zowel de measurements als control actions. Het belief van de robot wordt als volgt genoteerd: $bel(x_t) = p(x_t|u_t, z_t)$. Hier staat dus dat het belief van de robot afhangt van de measurement en update-action: "als de robot de action u_t gedaan heeft en als de measurement z_t gemeten wordt dan is de toestand waarschijnlijk x_t".

In een aantal gevallen proberen de algoritmes die gebruikt worden om de state in een HMM te berekenen rekening te houden met de noise in de sensoren. In dat geval is het nuttig om een meting te voorspellen. Het onderliggende idee is dat het verschil tussen de voorspelde measurement en de feitelijke measurement gebruikt kan worden om een fout in de measurement te corrigeren. In dat geval wordt een belief voorspeld (berekend) op grond van een theoretisch (natuuurkundig of wiskundig) model. Een voorspeld (berekend) belief wordt genoteerd als $\overline{bel}(x_t) = p(x_t|u_t, z_{t-1})$. Hoe het verschil tussen de voorspelde measurement en de feitelijke measurement wordt verwerkt staat in de volgende hoofdstukken over de diverse filters beschreven.

8.4. FILTERING

Het proces waarin het belief van een robot over de wereld wordt bijgesteld heet een *filter*. In dit dictaat zullen we de volgende filters bekijken:

- Bayes-filter: het meest algemene filter dat dient als basis voor de andere filters. Kan feitelijk alleen in hele simpele situaties gebruikt worden.
- Kalman-filter: een filter dat vooral gebruikt wordt om de nauwkeurigheid van metingen te verbeteren.
- Particle-filter; een filter dat door ons wordt gebruikt om de plaats van de robot te bepalen.

8.5. WAAR BEN IK?

In dit hoofdstuk wordt een intuitieve beschrijving gegeven van hoe een robot op grond van prior knowledge, measurements, control-actions en inference erachter komt waar hij is. In de hoofdstukken hierna wordt de intuitie vervangen door de bijbehorende wiskunde.

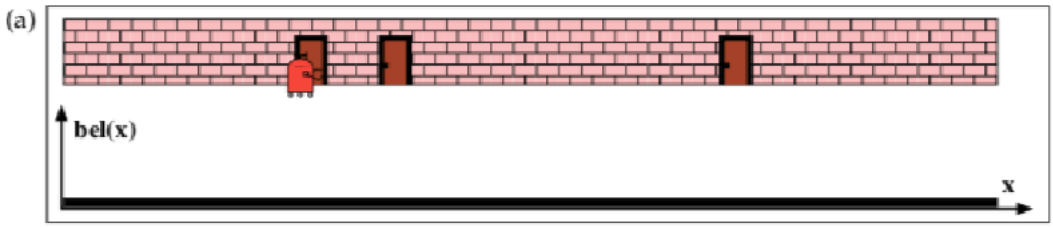
Het voorbeeld gaat uit van een twee-dimensionaal beeld van een wereld met daarin een robot en drie deuren. De robot kan door de wereld rijden en hij kan meten of hij voor een deur staat. In het voorbeeld rijdt de robot van links naar rechts door de wereld. Met het rijden als control action wordt in dit voor-

beeld niet zoveel gedaan. Maar met het meten wel. Verder heeft de robot kennis van de wereld: hij heeft een kaart van de wereld meegekregen als prior knowledge.

Onder aan ieder plaatje staat in het zwart het belief van de robot met betrekking tot zijn locatie (x). De hoogte van de grafiek geeft de mate van belief dat de robot op die plaats is weer. Als de robot een measurement doet dan staat in het rood de kans op de measurement, gegeven de state. Hier geeft de hoogte van de grafiek de kans weer dat de measurement gegeven de plaats (state) correct is.

8.5.1. ROBOT WEET VAN NIETS

Het begint met dat de robot van niets weet en leuk aan het rijden is zonder zijn locatie te kennen (figuur 8.7). Dit komt tot uitdrukking dat de overtuiging dat de robot op locatie x is even hoog is voor iedere locatie: er is sprake van een vlakke belief-verdeling.

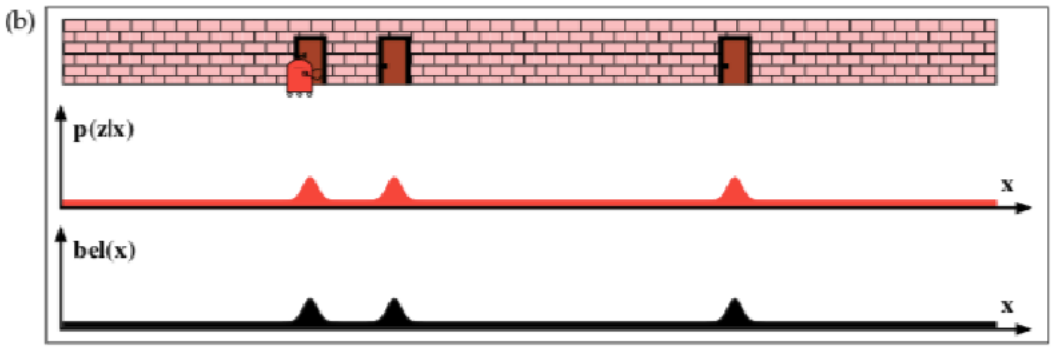


Figuur 8.7.: Robot weet van niets

8.5.2. ROBOT MEET EEN DEUR

Maar dan meet de robot een deur (figuur 8.8). Op grond van de kaart van de wereld kan de robot de conclusie trekken (inference) dat de kans op een dergelijke meting het hoogst is als de robot zich bevindt voor een deur. Dit komt tot uiting in de rode lijn: die piekt ter hoogte van ieder deur. De piek is voor iedere deur even hoog: de *meting* kan geen onderscheid maken tussen de deuren. Op grond van de meting en de kaart wordt het belief van de robot dat deze zich voor één van de deuren bevindt. Welke van de drie deuren weet de robot niet: hij heeft immers maar 1 measurement.

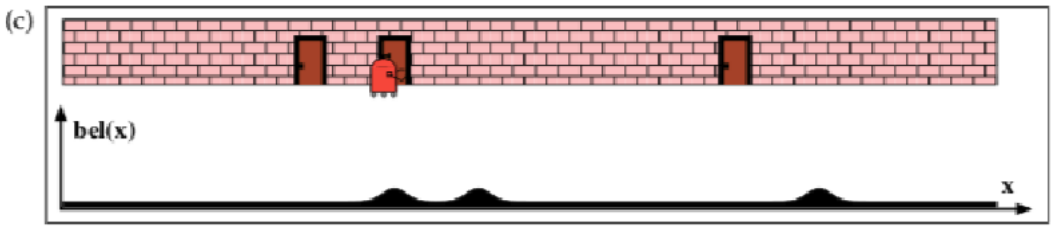
Merk op dat zowel de rode als de zwarte lijn in de gehele wereld zichtbaar is. Weliswaar heeft de robot nu een measurement waaruit zou kunnen blijken dat hij voor een deur staat maar die meting kan foutief zijn. En dat leidt tot een niet-nul belief over de de gehele wereld.



Figuur 8.8.: Robot meet een deur

8.5.3. ROBOT VERPLAATST

De robot gaat verder met rijden (figuur 8.9). Doordat alle update actions met onzekerheid gepaard gaan wordt het belief van de robot over waar die is minder sterk: de pieken worden minder hoog en de basis van de pieken breder.

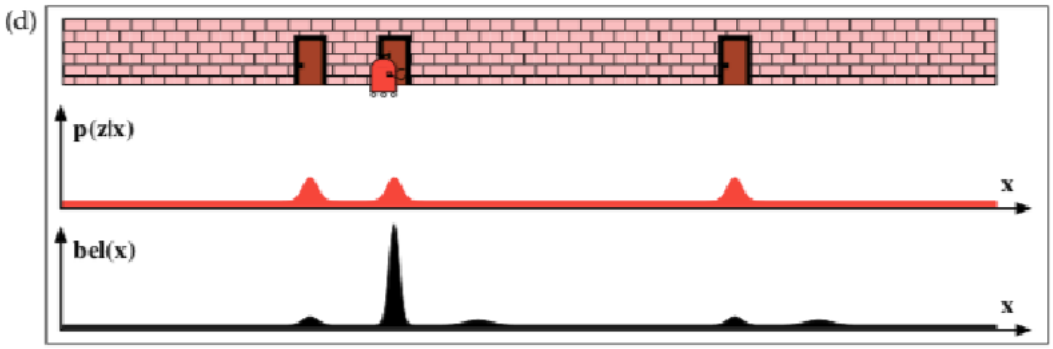


Figuur 8.9.: Robot verplaatst

8.5.4. ROBOT MEET WEER EEN DEUR

Maar dan meet de robot wederom een deur (figuur 8.10). En weer trekt de robot op grond van de kaart van de wereld de conclusie (inference) dat de kans op een dergelijke meting het hoogst is als de robot zich bevindt voor een deur. Dit komt tot uiting in de rode lijn: die piekt ter hoogte van ieder deur. En ook is de piek is voor iedere deur weer even hoog: de *meting* kan geen onderscheid maken tussen de deuren. Maar de kans dat hij voor de tweede deur staat piekt nu veel hoger dan alle andere mogelijke plaatsen waar hij zich zou kunnen bevinden. Dit zou kunnen zijn dat hij in combinatie van slim rekenen en de interne kaart kan uitdokteren dat de combinatie van twee keer een deur en een bepaalde rijafstand tussen die twee metingen zijn plaats wel voor de tweede deur moet zijn. Maar hij kan ook gewoonweg zijn nieuwe belief op basis van de nieuwe meting optellen bij zijn voorgaande belief: ook in dat geval wordt het belief dat hij voor de tweede deur staat het grootst.

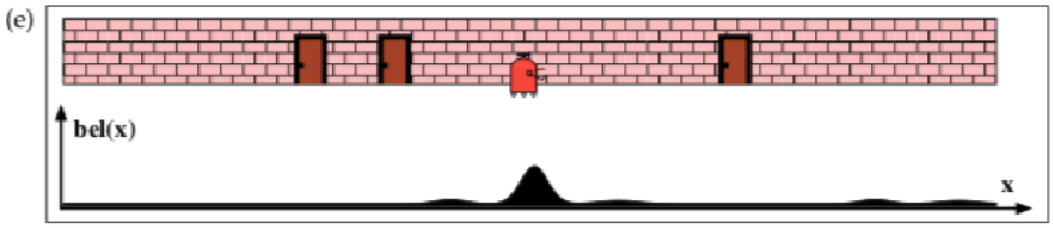
Ook in dit geval blijft zijn belief in alternatieve locaties niet nul: de kans is misschien wel klein maar er kunnen gewoon twee foute metingen achter elkaar gedaan zijn. Of de eerste was fout en de tweede correct. Of omgekeerd.



Figuur 8.10.: Robot meet een weer deur

8.5.5. ROBOT VERPLAATST ZICH VERDER

De robot gaat weer verder met rijden (figuur 8.11). En net als de vorige keer wordt het belief van de robot over waar die is minder sterk doordat alle update actions met onzekerheid gepaard gaan: de pieken worden minder hoog en de basis van de pieken qbreder.



Figuur 8.11.: Robot verplaatst

9. BAYES-FILTER

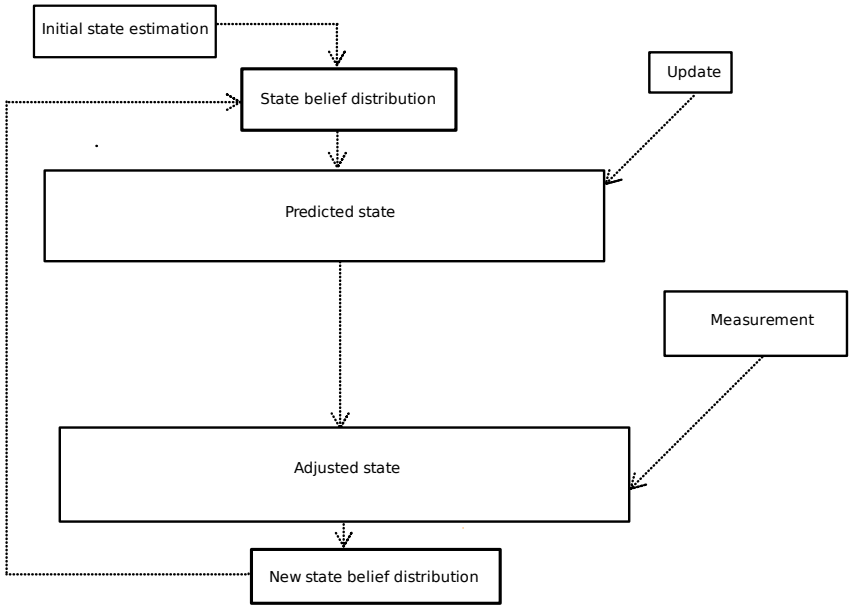
9.1. INLEIDING

Via Google Translate, de introductie van Recursive Bayesian estimation:

In waarschijnlijkheidstheorie, statistiek en machine learning: recursieve Bayesiaanse schatting, ook bekend als een Bayes-filter, is een algemene probabilistische benadering voor het recursief schatten van een onbekende kansdichtheidsfunctie (PDF) in de loop van de tijd met behulp van inkomende metingen en een wiskundig procesmodel. Het proces is sterk afhankelijk van wiskundige concepten en modellen die worden getheoretiseerd in een studie van eerdere en posterieure waarschijnlijkheden die bekend staan als Bayesiaanse statistiek.

9.2. HET BAYES-FILTER ALGORITME

Een Bayes-filter berekent het geloof (belief) op basis van het vorige belief, waarnemingen (measurements) en besturing (control actions). In dit dictaat gaan we in op het *recursive state estimation Bayes-filter*. Het recursieve van de filter zit in het feit dat het filter het vorige belief meeneemt in zijn berekening. Het Bayes-filter is het basis filter dat in (sterk) aangepaste vorm ook gebruikt wordt bij het Kalman-filter en het particle-filter. Hoewel er zowel discrete als continue varianten van Bayes-filters bestaan gaan we in dit hoofdstuk alleen maar in op de discrete variant. In figuur 9.1 staat een schematisch overzicht van het Bayes-filter.

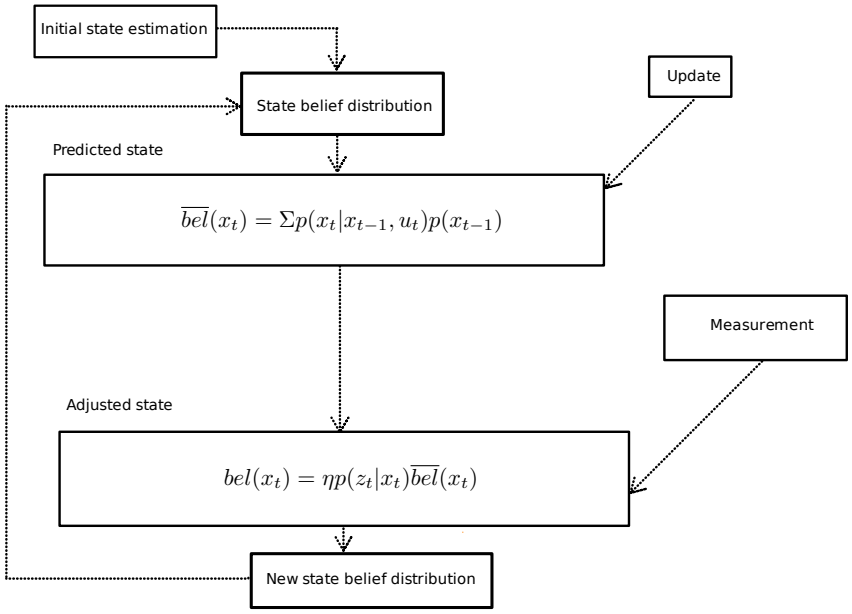


Figuur 9.1.: Overzicht Bayes-filter

Het algoritme voor een Bayes-filter is als volgt:

```
input bel(x_{t-1}), u_t, z_t
for all x_t do
    // Control update
    bel̄(x_t) = ∑ p(x_t|u_t, x_{t-1}) p(x_{t-1})
    // Measurement update
    bel(x_t) = η p(z_t|x_t) bel̄(x_t)
endfor
return bel(x_t)
```

In figuur 9.2 staat een schematisch overzicht van het Bayes-filter met de formules van het algoritme ingevuld.



Figuur 9.2.: Overzicht Bayes-filter met formules

In Nederlands staat hier het volgende:

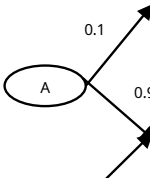
Doe voor alle mogelijke toestanden:

1. de *control update*: voorspel (het geloof in) de nieuwe toestand x_t, gegeven de vorige toestand x_{t-1} en de update u_t door de som te nemen van de kansen van alle updates die die toestand kunnen veroorzaken. (theorem of total probability).
2. de *measurement update*: berekenen (het geloof in) de nieuwe toestand, gegeven de de waarneming z_t en (het geloof in) de voorspelde nieuwe toestand.

Opmerking: de normalisatie constante η wordt veroorzaakt doordat de kansen in het systeem gegeven worden in termen van kansen om *uit* een toestand weg te gaan. Door de manier waarop het algoritme werkt kunnen er bij het berekenen van de total probability getallen ongelijk aan 1.0 als de som van de kansen uitkomen. Dit wordt gecorrigeerd met behulp van η.

In figuur 9.3 staat een FSM. Stel je bent met een kans van 0.5 in toestand A of B. Vanuit state A kom je met een kans van 0.9 in toestand C en met een kans van 0.1 in de toestand X. Vanuit state B kom je met een kans van 0.9 in ook toestand C en met een kans van 0.1 in de toestand Y. Dan is het duidelijk dat de kans om in C te komen niet $0.9 + 0.9 = 1.8$ is. Het is immers ook mogelijk om in X of Y te komen, afhankelijk van de vraag of je in A dan wel B was op het moment van het uitvoeren van update U. Intuitief is de kans om in X terecht te komen $\frac{1}{0.1+0.9+0.1+0.9} = 0.5$. De constante zorgt ervoor dat de kansen wel optellen tot 1.0.

Het discrete Bayes-filter algoritme berekent de kansverdeling over *alle* states het systeem. Dat betekent praktisch gezien dat het filter in deze vorm alleen maar bruikbaar is bij systemen met een eindige state space. Dit zijn typisch systemen waarvoor je een FSM kunt tekenen.



Figuur 9.3.: Uitleg η

9.3. VOORBEELD APPEL-PLUKKENDE BAYES ROBOT

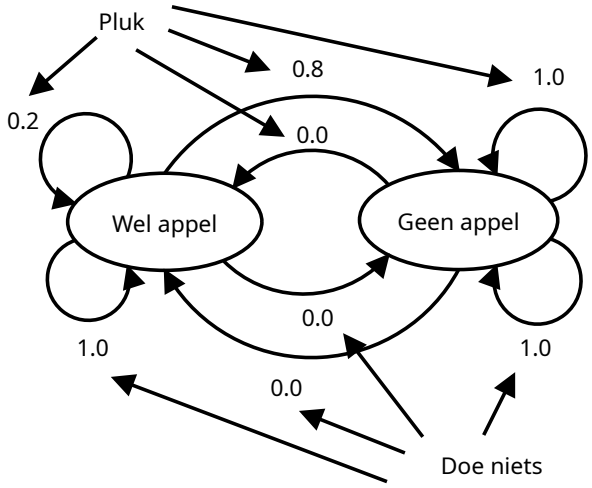
We zijn in het gelukkige bezit een appel-plukkende robot. Deze robot kan naar een boom rijden en daar met behulp van een sensor kijken of er een appel in de boom zit. Als de robot denkt dat er een appel in de boom zit dan kan de robot uiteraard proberen de appel te plukken (en uiteindelijk op de lopende band van sectie 7.6 te deponeren). De robot kan natuurlijk ook altijd besluiten om niets te doen

9.3.1. DE UPDATE- EN MEASUREMENTKANSEN

Zowel de actuatoren als sensoren van de robot zijn noisy, i.e. probabilistisch.

9.3.1.1. UPDATE-KANSEN

Als robot kan bij de appelboom staat dan kan hij op zich twee dingen doen: hij kan de appel proberen te plukken of hij kan niets doen.



Figuur 9.4.: De appel-plukkende robot

Als de robot besluit tot het plukken van de appel dan verandert er wellicht iets aan de toestand van de appel. Helaas voor de robot is het plukken niet geheel deterministisch:

$p(x_t = no\_apple\_present | u_t = grab\_apple, x_{t-1} = apple\_present) = 0.8$
$p(x_t = apple\_present | u_t = grab\_apple, x_{t-1} = apple\_present) = 0.2$
$p(x_t = no\_apple\_present | u_t = grab\_apple, x_{t-1} = no\_apple\_present) = 1.0$
$p(x_t = apple\_present | u_t = grab\_apple, x_{t-1} = no\_apple\_present) = 0.0$

Als er een appel aan de boom hangt dan lukt het plukken in veel gevallen: in 80% van de gevallen lukt het. Helaas mislukt het ook wel eens: dat gebeurt in 20% van de gevallen.

Niet geheel verrassend is het plukken van een appel die er niet is ($x_{t-1} = no\_apple\_present$) verbazingwekkend voorspelbaar effectief. Het plukken van een niet bestaande appel lukt namelijk in alle gevallen: na het plukken van een niet bestaande appel hangt er geen appel meer aan de boom. En wonder boven wonder leidt het plukken van een niet aanwezige appel niet tot het materialiseren van een appel.

Als de robot niets doet dan verandert er natuurlijk niets aan de beschikbaarheid (toestand) van de appel:

$p(x_t = no\_apple\_present | u_t = do\_nothing, x_{t-1} = no\_apple\_present) = 1.0$
$p(x_t = apple\_present | u_t = do\_nothing, x_{t-1} = no\_apple\_present) = 0.0$
$p(x_t = no\_apple\_present | u_t = do\_nothing, x_{t-1} = apple\_present) = 0.0$
$p(x_t = apple\_present | u_t = do\_nothing, x_{t-1} = apple\_present) = 1.0$

Niets doen verandert niet echt veel aan de kans op een aanwezige appel. Als er op $t-1$ geen appel was dan verwachten we dat er op tijdstip t niet echt heel veel veranderd is: er is, ongeacht wat we doen, nog steeds geen appel. Hetzelfde geldt voor het omgekeerde: als er op $t-1$ wel een appel was dan zal die er bij niets doen er op t nog steeds wel zijn.

9.3.1.2. MEASUREMENT-KANSEN

De robot kan met behulp van de sensor meten of er wel of geen appel aanwezig aanwzig is. Helaas is de sensor noisy. Wat de sensor meet is afhankelijk van de aanwezigheid van appels.

Als er daadwerkelijk een appel aan de boom hangt en de robot meet met de sensor dat er wel een appel aan de boom hangt dan zijn de kansen als volgt:

$p(z_t = see\_no\_apple|x_t = apple\_present) = 0.2)$
$p(z_t = see\_apple|x_t = apple\_present) = 0.8)$

Als er evenwel geen appel aan de boom hangt en de robot meet met de sensor dan zijn de kansen als volgt:

$p(z_t = see\_no\_apple|x_t = no\_apple\_present) = 0.6)$
$p(z_t = see\_apple|x_t = no\_apple\_present) = 0.4)$

Uit het bovenstaande blijkt dus dat de sensor van de robot relatief goed is in het detecteren van appels en relatief slecht is in het detecteren van geen appels.

9.3.2. INITIAL STATE ESTIMATION

Als een robot onder een boom staat en hij weet verder van niets, dan rest de robot niets anders dan aan te nemen dat de kans op een bepaalde toestand uniform verdeeld is over de toestanden. Een uniforme kansverdeling is een kansverdeling waarbij alle toestanden evenveel kans hebben. Oftewel, de kans dat er wel of geen appel hangt is even groot:

$bel(x_{t=0} = no\_apple\_present) = 0.5$
$bel(x_{t=0} = apple\_present) = 0.5$

9.3.3. NIETS DOEN MAAR WEL DE APPEL UIT DE BOOM KIJKEN

Stel dat de robot eerst maar eens besluit om niets te doen en maar eerst eens te gaan kijken of er een appel hangt. Wat gebeurt er met het berekende/voorspelde belief van de robot als deze besluit niets te doen maar gewoon eerst eens de appel uit de boom te kijken? Als de robot niets doet dan is zijn verwachting van het niets doen het volgende.

9.3.3.1. UPDATE

Eerst het belief ervan uitgaande dat er er daadwerkelijk *wel* een appel is en dat de robot niets doet:

$\overline{bel}(x_{t+1} = apple\_present, u_{t+1} = do\_nothing) =$
$p(x_{t+1} = apple\_present|u_{t+1} = do\_nothing, x_t = no\_apple\_present)bel(x_t = no\_apple\_present)+$
$p(x_{t+1} = apple\_present|u_{t+1} = do\_nothing, x_t = apple\_present)bel(x_t = apple\_present)$
$= 0.0 \cdot 0.5 + 1.0 \cdot 0.5 = 0.5$

Dan het belief ervan uitgaande dat er er daadwerkelijk *geen* appel is en dat de robot niets doet:

$\overline{bel}(x_{t+1} = no\_apple\_present, u_{t+1} = do\_nothing) =$
$p(x_{t+1} = no\_apple\_present|u_{t+1} = do\_nothing, x_t = no\_apple\_present)bel(x_t = no\_apple\_present)+$
$p(x_{t+1} = no\_apple\_present|u_{t+1} = do\_nothing, x_t = apple\_present)bel(x_t = apple\_present)$
$= 1.0 \cdot 0.5 + 0.0 \cdot 0.5 = 0.5$

Deze uitkomsten zijn natuurlijk niet verrassend. Gegeven de kans van niets doen (als je niets doet verandert er ook niets) zal de robot niet verwachten dat zijn inschatting van de situatie verandert. Kortom, op grond van zijn belief en niets doen verandert er niet zo heel veel (niets) aan zijn berekende/theoretische belief.

9.3.3.2. MEASUREMENT

Maar wat gebeurt er als de robot rekening gaat houden houden met zijn waarneming? Wat is volgens de robot zijn belief als hij een waarneming doet? Laten we aannemen dat de robot een measurement doet waaruit blijkt dat er *wel* een appel aan de boom hangt.

Het algoritme zegt:

$bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$

Eerst het $bel(x_{t+1})$ als er wel appel aanwezig is:

$$bel(x_{t+1} = apple\_present)$$
$$= \eta \cdot p(z_{t+1} = see\_apple|x_{t+1} = apple\_present)\overline{bel}((x_{t+1} = apple\_present)$$
$$= \eta \cdot 0.8 \cdot 0.5$$
$$= \eta \cdot 0.4$$

Dan het $bel(x_{t+1})$ dat er geen appel aanwezig is:

$$bel(x_{t+1} = no\_apple\_present)$$
$$= \eta \cdot p(z_{t+1} = see\_apple|x_{t+1} = no\_apple\_present)\overline{bel}(x_{t+1} = no\_apple\_present)$$
$$= \eta \cdot 0.4 \cdot 0.5$$
$$= \eta \cdot 0.2$$

De berekening van η is dan:

$bel(x_{t+1} = no\_apple\_present) + bel(x_{t+1} = apple\_present) = \eta \cdot 0.2 + \eta \cdot 0.4 = 1.0$

Dus $\eta = \frac{1.0}{0.2+0.4} = 1.67$

De berekening uiteindelijke belief wordt dan:

$$bel(x_{t+1} = no\_apple\_present) = 1.67 \cdot 0.2 = 0.33$$
$$bel((x_{t+1} = apple\_present) = 1.67 \cdot 0.4 = 0.67$$

Dit betekent dus dat de robot zekerder is geworden dat er een appel hangt op basis van de waarneming. Wat ook te zien is, is dat de overtuiging niet gelijk is aan de zekerheid die de waarneming als zodanig biedt. Klaarblijkelijk houdt de robot in het achterhoofd dat de meting fout zou kunnen zijn.

9.3.4. PLUK DE APPEL!

Stel dat de robot hierna besluit om de appel te plukken. De control action is dus “pluk appel”.

9.3.4.1. UPDATE

Eerst het belief in het geval dat het plukkken lukt, i.e. de appel hangt er niet meer:

$$\overline{bel}(x_{t+2} = no\_apple\_present, u_{t+2} = grab\_apple)$$
$$p(x_{t+2} = no\_apple\_present|u_{t+2} = grab\_apple, x_{t+1} = apple\_present)bel(x_{t+1} = apple\_present) +$$
$$p(x_{t+2} = no\_apple\_present|u_{t+2} = grab\_apple, x_{t+1} = no\_apple\_present)bel(x_{t+1} = no\_apple\_present)$$
$$= 0.8 \cdot 0.67 + 1.0 \cdot 0.33 = 0.87$$

Dan het belief in het geval het plukken niet lukt, i.e. de appel hangt er nog steeds:

$$\overline{bel}(x_{t+2} = apple\_present, u_{t+2} = grab\_apple)$$
$$p(x_{t+2} = apple\_present|u_{t+2} = grab\_apple, x_{t+1} = apple\_present)bel(x_{t+1} = apple\_present) +$$
$$p(x_{t+2} = apple\_present|u_{t+2} = grab\_apple, x_{t+1} = no\_apple\_present)bel(x_{t+1} = no\_apple\_present)$$
$$= 0.2 \cdot 0.67 + 0.0 \cdot 0.33 = 0.13$$

Hieruit blijkt dat de robot denkt dat de control action van het plukken van de appel gaat leiden tot een grotere kans op de state “geen appel”, maar er blijft een kans op de state “appel aanwezig”.

9.3.4.2. MEASUREMENT

Wat is volgens de robot nu zijn belief als deze de waarneming doet dat er geen appel meer is.

Eerst het belief op “geen appel”, gegeven de waarneming “geen appel”:

$$bel(x_{t+2} = no\_apple\_present)$$
$$= \eta \cdot p(z_{t+2} = see\_no\_apple|x_{t+2} = no\_apple\_present)\overline{bel}(x_{t+2} = no\_apple\_present)$$
$$= \eta \cdot 0.6 \cdot 0.87$$
$$= \eta \cdot 0.52$$

Vervolgens het belief op “appel aanwezig”, gegeven de waarneming “geen appel”:

$$
\begin{aligned}
&bel(x_{t+2} = apple\_present) \\
&= \eta \cdot p(z_{t+2} = see\_no\_apple | x_{t+2} = apple\_present) \overline{bel}(x_{t+2} = apple\_present) \\
&= \eta \cdot 0.2 \cdot 0.13 \\
&= \eta \cdot 0.026
\end{aligned}
$$

De berekening van η is dan:

$$bel(x_{t+2} = no\_apple\_present) + bel(x_{t+2} = apple\_present) = \eta \cdot 0.52 + \eta \cdot 0.026 = 1.0$$

Dus $\eta = \frac{1.0}{0.52+0.026} = 1.83$

De berekening uiteindelijke belief wordt dan:

$$
\begin{aligned}
&bel(x_{t+1} = no\_apple\_present) = 1.83 \cdot 0.52 = 0.95 \\
&bel(x_{t+1} = apple\_present) = 1.83 \cdot 0.026 = 0.05
\end{aligned}
$$

Op dit moment denkt de robot dat hij met een kans van 0.95 de appel geplukt heeft en dat het met een kans van 0.05 mislukt is. Of deze kansen voor de boer goed genoeg zijn om de robot in dienst te houden is open voor discussie...

10. KALMAN-FILTER

Zie ook Wikipedia:

- Kalman-filter (Nederlands): Kalman-filter
- Kalman-filter (Engels): Kalman-filter

Dit hoofdstuk is gebaseerd op twee bronnen: het boek "Probabilistic robotics" (Thrun, Burgard en Fox 2005) en een zeer goede video op ILecture Online. Helaas hanteert de docent in de video een iets andere terminologie dan Thrun. Gelukkig kan er laagdrempelig een vertaaltabel gemaakt worden.

	Thrun e.a.	ILectureOnline
Input	$\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$	$X_{k-1}, P_{k-1}, u_k, Y_k$
Voorspelling	$\overline{\mu}_t = A_t\mu_{t-1} + B_t u_t + \epsilon_t$ $\overline{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$	$X_{k_p} = AX_{k-1} + Bu_k + w_k$ $P_{k_p} = AP_{k-1}A^T + Q_k$
Kalman gain	$K_t = \overline{\Sigma}_t C_t^T (C_t\overline{\Sigma}_t C_t^T + Q_t)^{-1}$	$K = \frac{P_{k_p}H}{HP_{k_p}H^T+R}$
Bijstelling	$\mu_t = \overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t)$ $\Sigma_t = (I - K_tC_t)\overline{\Sigma}_t$	$X_k = X_{k_p} + K(Y_t - HX_{k_p})$ $P_k = (I - KH)P_{k_p}$
Output	μ_t, Σ_t	X_k, P_k

Tabel 10.1.: Vertaling Thrun/ILectureOnline

In de colleges en dit dictaat wordt de terminologie van Thrun e.a. gebruikt.

10.1. INLEIDING

Een Kalman-filter is een iteratieve rekenmethode die gebruik maakt van een theoretisch(wiskundig) model van een proces en van meet-gegevens uit bijvoorbeeld sensoren om de fouten/onzekerheid/variatie af te handelen. Kalman-filters gaan er van uit dat de wereld Gaussisch is: alle fouten, onzekerheden en variatie zijn Gaussisch. Een Kalman-filter is een unimodale kansverdeling over de mogelijke toestanden: slechts één van de mogelijke werelden heeft de grootste waarschijnlijkheid.

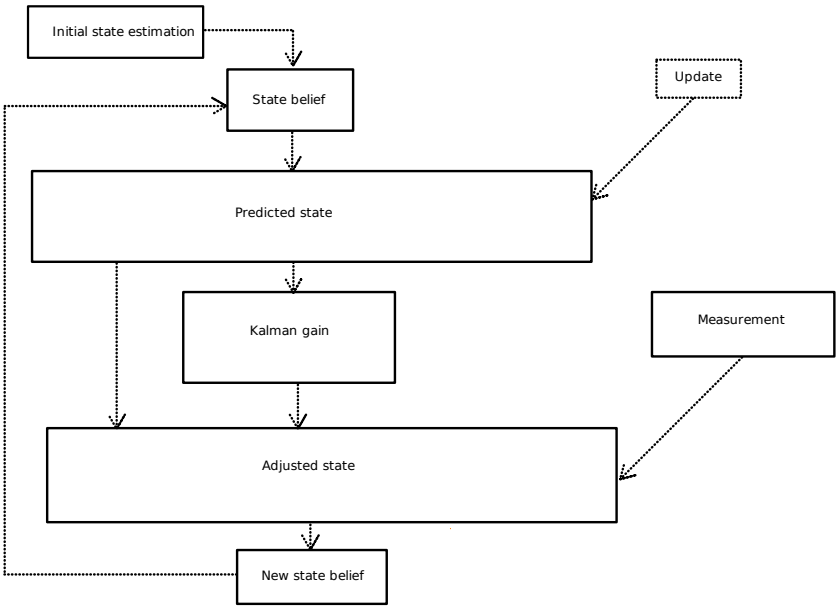
Kalman-filters worden heel veel toegepast in de meet- en regeltechniek. In de bovenstaande Wikipedia-artikelen worden heel veel toepassingen genoemd. Er bestaan diverse varianten van het filter maar die delen allemaal hetzelfde basis-principe. In figuur 10.1 staat de werking van het Kalman-filter schematisch weergegeven.

Zoals een vergelijking van figuur 10.1 en figuur 9.1 laat zien gaat het Kalman-filter van dezelfde twee fundamentele stappen uit als het Bayes-filter maar voegt een extra stap, het berekenen van de *Kalman gain*, toe:

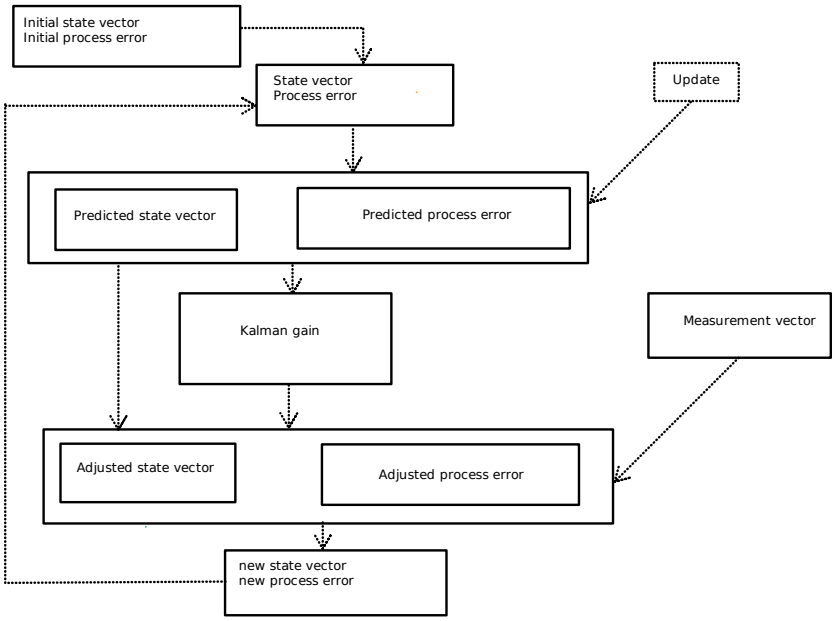
1. de *control update*: voorspel (het geloof in) de nieuwe toestand ($\overline{\mu}_t$ en $\overline{\Sigma}_t$) op basis van de vorige toestand (μ_{t-1} en Σ_{t-1}) en de update u_t,
2. de *Kalman gain*: bepaal het belang dat we hechten aan de (berekende) process error of aan de measurement error,
3. de *measurement update*: berekenen (het geloof in) de nieuwe toestand (μ_t en Σ_t), gegeven de waarneming (z_t), de Kalman gain (K_t) en (het geloof in) de voorspelde nieuwe toestand ($\overline{\mu}_t$ en $\overline{\Sigma}_t$).

Behalve het toevoegen van een extra stap is er ook nog een verschil in de representatie van de toestanden. Daar waar het Bayes-filter uit gaat van kansverdeling over concrete toestanden wordt er in het Kalman-filter een toestand beschreven door twee variabelen: een state vector en een covariantiematrix[1]. De state vector bevat alle variabelen die noodzakelijk zijn om de state te beschrijven. De covariantiematrix bevat de onzekerheid over de variabelen in de state vector. Dat betekent praktisch gezien, zoals in

1 Een covariantiematrix is de variantie van een vector.



Figuur 10.1.: Overzicht Kalman-filter



Figuur 10.2.: Detailoverzicht Kalman-filter

figuur 10.2 te zien is, dat zowel in de control update als in de measurement update de state vector en de covariantiematrix moeten worden bijgewerkt.

10.2. HET KALMAN-FILTER ALGORITME

Het algoritme van het Kalman-filter is als volgt:

input $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$
// Control update
$\overline{\mu}_t = A_t\mu_{t-1} + B_t u_t + \epsilon_t$
$\overline{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$
// Kalman gain
$K_t = \overline{\Sigma}_t C_t^T (C_t\overline{\Sigma}_t C_t^T + Q_t)^{-1}$
// Measurement update
$\mu_t = \overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t)$
$\Sigma_t = (I - K_t C_t)\overline{\Sigma}_t$
// Done
return μ_t, Σ_t

Het algoritme heeft feitelijk dus 5 berekeningen verdeeld over 3 fases:

1. Control update:
   a) Bereken de voorspelde state vector.
   b) Bereken de voorspelde process error[2].
2. Kalman gain:
   a) Bereken de Kalman gain.
3. Measurement update:
   a) Bereken de bijgestelde state vector.
   b) Bereken de bijgestelde process error.

In figuur 10.3 is de plaats van de diverse formules weergegeven in de flow van het algoritme.

Zoals al eerder vermeld is het Kalman-filter een iteratieve rekenmethode. De frequentie van de iteraties is in het algemeen gekoppeld aan de combinatie van de informatiebehoefte en de frequentie waarmee nieuwe measurements beschikbaar komen. Hierbij is de frequentie van de measurements sowieso de beperkende factor: de frequentie van interaties kan niet sneller dan de frequentie van de measurements.

10.2.1. CONTROL UPDATE

In de control update wordt met behulp van een theoretisch/wiskundig model een nieuwe toestand voorspeld. Dit model moet door materie-deskundigen uit vakgebieden als werktuigbouw, electrotechniek en natuurkunde worden opgesteld. Het model bestaat uit twee delen: een deel dat de ontwikkeling van de state afhandeld en een deel dat de ontwikkeling van de errors (onzekerheid) afhandelt. De nieuwe toestand wordt voorspeld op basis van een tweetal berekeningen: een voor de state vector en een voor de process covariantiematrix.

10.2.1.1. PREDICTED STATE VECTOR

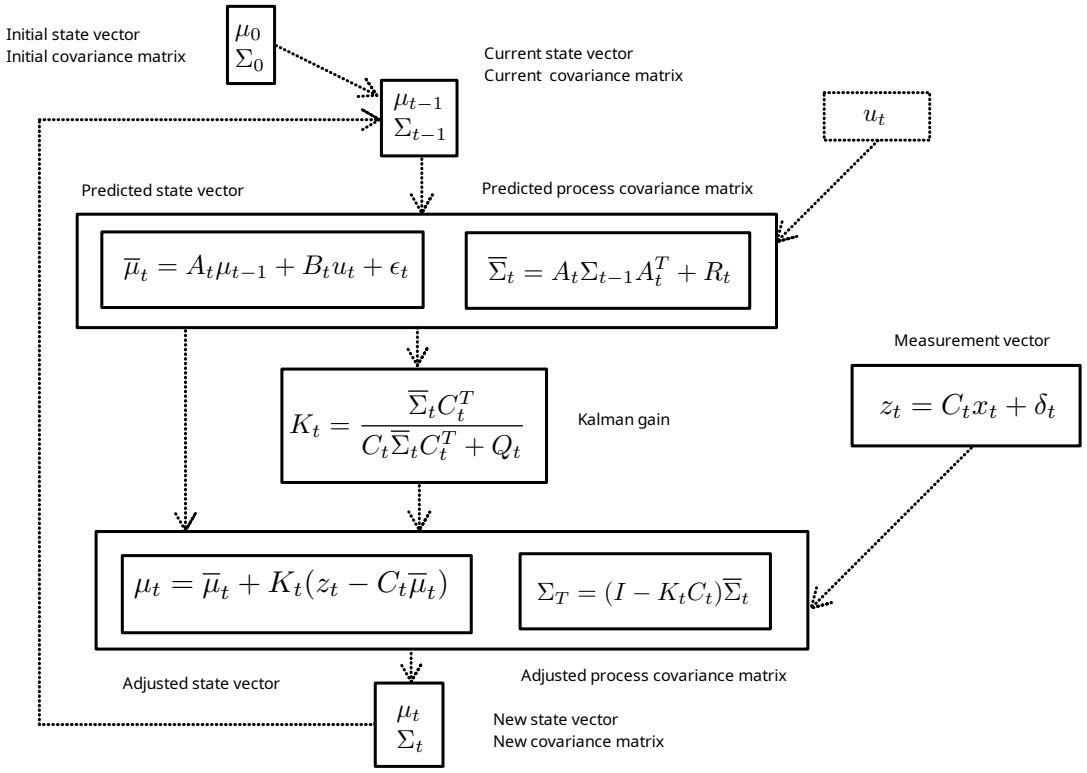
De voorspelling van de predicted state vector komt tot stand door de natuurlijke ontwikkeling van de state, i.e. de verandering van de state, op te tellen bij de ontwikkeling van de state als gevolg van de update en de noise die in het proces zit.

$\overline{\mu}_t = A_t\mu_{t-1} + B_t u_t + \epsilon_t$

Hierbij is:

– $\overline{\mu}_t$: de voorspelde state vector,

2 In de meeste Kalman-filter literatuur wordt er over “errors” gesproken. Een wellicht wat meer geschikte term is “onzekerheid”: het gaat immers over de onzekerheid van de uitkomst van het proces.



Figuur 10.3.: Overzicht Kalman-filter met formules

- A_t: de transitiematrix die de ontwikkeling van de state *zonder updates* modelleert,
- μ_{t-1} : de state vector van de vorige iteratie,
- B_t: de transitiematrix die de ontwikkeling van de state *als gevolg van de updates* modelleert,
- u_t: de control vector,
- ϵ_t: de predicted process noise vector.

$\overline{\mu}_t$ wordt berekend door de eigen verandering van de state, de verandering als gevolg van de update en de ruis in de state bij elkaar op te tellen.

$\overline{\mu}_t$ en μ_{t-1} zijn even grote state vectoren met de variabelen die de state beschrijven.

De transitiematrix A_t modelleert de ontwikkeling van de state *zonder updates*. De oude state vermenigvuldigd met de transitiematrix A_t levert een nieuwe matrix[3] op met de grootte van de state vectoren. A_t is hierbij een vierkante matrix met een grootte van $m \times m$ waarbij m de grootte van de state vector(en) is[4].

De transitiematrix B_t modelleert de ontwikkeling van de state *als gevolg van de updates*. De update vector vermenigvuldigd met de transitiematrix B_t levert een nieuwe vector op met de grootte van de state vectoren. De grootte van de matrix B_t is $m \times n$ waarbij m de grootte van de state vectoren is en n de grootte van de update vector.

De predicted process noise vector ϵ_t is een vector van ruis-waarden van de state variabelen. De vector is net zo groot als μ_t. De ruis-waarden worden getrokken uit een Gaussische kansverdeling met een gemiddelde van 0 en een gegeven standaarddeviatie.

10.2.1.2. PREDICTED PROCESS COVARIANCE

$\overline{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$

Hierbij is:

- $\overline{\Sigma}_t$: de voorspelde process covariantiematrix,
- Σ_{t-1}: de process covariantie matrix van de vorige iteratie,

3 Dit is goed te vergelijken met een transitiematrix bij kansrekening.
4 Zoals in hoofdstuk3 is vermeld is een $m \times n$ matrix een matrix met m rijen en n kolommen.

- A_t en A_t^T: de transitiematrix die de ontwikkeling van de state *zonder updates* modelleert (zie predicted state vector),
- R_t: de process noise covariantiematrix.

$\overline{\Sigma}_t$ wordt berekend door de process covariantie aan de vorige iteratie aan te passen met de verandering van de state en daar process noise aan toe te voegen.

De proces covariantiematrix R_t bevat de onzekerheid met betrekking tot de process error. R_t is de covariantiematrix van gebruikte standaarddeviaties bij ϵ_t, de process noise.

10.2.2. KALMAN GAIN

De Kalman gain wordt gebruikt om te bepalen in welke mate de (theoretische) voorspelling of de (praktische) measurement zwaarder weegt bij het bijstellen van de voorspelde state vector en process covariantiematrix. De Kalman gain is een getal ≥ 0 en ≤ 1. Naarmate de Kalman gain groter is wordt er meer belang gehecht aan de measurement, naarmate de Kalman gain kleiner is wordt er meer belang gehecht aan de voorspelling.

$$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1} = \frac{\overline{\Sigma}_t C_t^T}{C_t \overline{\Sigma}_t C_t^T + Q_t}$$

Hierbij is:

- K_t: de Kalman gain,
- $\overline{\Sigma}_t$: de berekende process covariantie matrix,
- C_t^T en C_t: een matrix en haar transpose om grootte-verschillen in matrices op te lossen,
- Q_t: de sensor covariantiematrix.

De sensor covariantiematrix Q_t bevat de onzekerheid met betrekking tot de meetnauwkeurigheid van de sensoren. Q_t is de covariantiematrix van gebruikte standaarddeviaties bij δ_t, de measurement noise.

In de berekening van de Kalman gain is te zien dat als Q_t klein is ten opzichte van $\overline{\Sigma}$ $\frac{\overline{\Sigma}}{\overline{\Sigma}}$ overheerst in de deling. De Kalman gain zal dan dus (ongeveer) 1.0 worden. Als Q_t groot is ten opzichte van $\overline{\Sigma}$ dan zal $\frac{1}{Q_t}$ gaan overheersen. De Kalman gain zal dan dus (ongeveer) 0.0 worden.
De Kalman gain wordt uiteindelijk in de measurement update bij het berekenen van de bijstelling van de state en covariantie gebruikt. Bij een grote Kalman gain (hoge error(onzekerheid) in de voorspelling, weinig error (onzekerheid) in de meting) zal het verschil tussen measurement en voorspelde state veel bijdragen aan de nieuwe state. Bij een kleine Kalmain gain (weinig error (onzekerheid) in de voorspelling, veel error (onzekerheid) in de meting) dan zal het verschil tussen measurement en voorspelde state weinig bijdragen aan de nieuwe state. Hetzelfde geldt voor de update in de covariantie.

10.2.3. MEASUREMENT UPDATE

In de measurement update worden de voorspelde state vector en process covariantiematrix bijgesteld op grond van de Kalman gain en de measurement.

10.2.3.1. MEASUREMENT

Een essentieel onderdeel van een Kalman-filter is de mogelijkheid om een daadwerkelijke meting te doen met behulp van sensoren.

$z_t = C_t x_t + \delta_t$

Hierbij is:

- z_t: de measurement vector
- C_t: een matrix om grootte-verschillen in matrices op te lossen,
- x_t: de state vector,
- δ_t: de measurement noise.

De measurement noise δ_t is een vector van ruis-waarden van de sensoren. De vector is net zo groot als z_t. De ruis-waarden worden getrokken uit een Gaussische kansverdeling met een gemiddelde van 0 en de voor de sensor gegeven standaarddeviatie.

Als er net zoveel metingen gedaan kunnen worden als er elementen in de measurement vector z_t en dus

als de grootte van de state vector x_t gelijk is aan de grootte van de measurement vector z_t dan is de matrix C_t een identiteitsmatrix ter groote van x_t. Anders is het een matrix met net zoveel rijen als de grootte van de measurement vector z_t en net zoveel kolommen als er sensoren zijn. En met het goed plaatsen van 0 in de matrix wordt dan de ontbrekende sensorwaarde op 0 gezet.

Normaal gesproken is C_t de identiteitsmatrix: alle elementen in z_t kunnen gemeten worden. Uitzonderingen hierop zijn typisch uitvallende sensoren of een verschil in de frequentie van de metingen.

10.2.3.2. ADJUSTED STATE VECTOR

Het uitrekenen van de bijgestelde state vector gebeurt door de voorspelde state vector te corrigeren voor de Kalman gain en het verschil tussen de measurement en de voorspelling.

$\mu_t = \overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t)$

Hierbij is:

- μ_t: de bijgestelde state vector
- $\overline{\mu}_t$: de voorspelde state vector,
- K_t: de Kalman gain,
- z_t: de waarneming,
- C_t: een matrix om grootte-verschillen in matrices op te lossen,
- $\overline{\mu}_t$: de voorspelde state vector.

De matrix C_t is analoog aan de gelijknamige matrix bij de measurement een matrix die eventuele grootte-verschillen tussen matrices oplost. Als er geen grootte-verschillen zijn is ook deze matrix een identiteitsmatrix.

10.2.3.3. ADJUSTED PROCESS COVARIANCE

Het uitrekenen van de bijgestelde process covariantie matrix gebeurt door de voorspelde process covariantie matrix te corrigeren voor de Kalman gain.

$\Sigma_t = (I - K_tC_t)\overline{\Sigma}_t$

Hierbij is:

- Σ_t: de bijgestelde process covariantie matrix,
- I: identiteitsmatrix,
- K_t: Kalman gain,
- C_t: een matrix om grootte-verschillen in matrices op te lossen,
- $\overline{\Sigma}_t$: de voorspelde process covariantie matrix.

De matrix C_t is analoog aan de gelijknamige matrix bij de measurement een matrix die eventuele grootte-verschillen tussen matrices oplost. Als er geen grootte-verschillen zijn is ook deze matrix een identiteitsmatrix.

10.2.4. OVERIGE DETAILS

In figuur 10.4 is de flow van de variabelen in de berekening weergegeven.

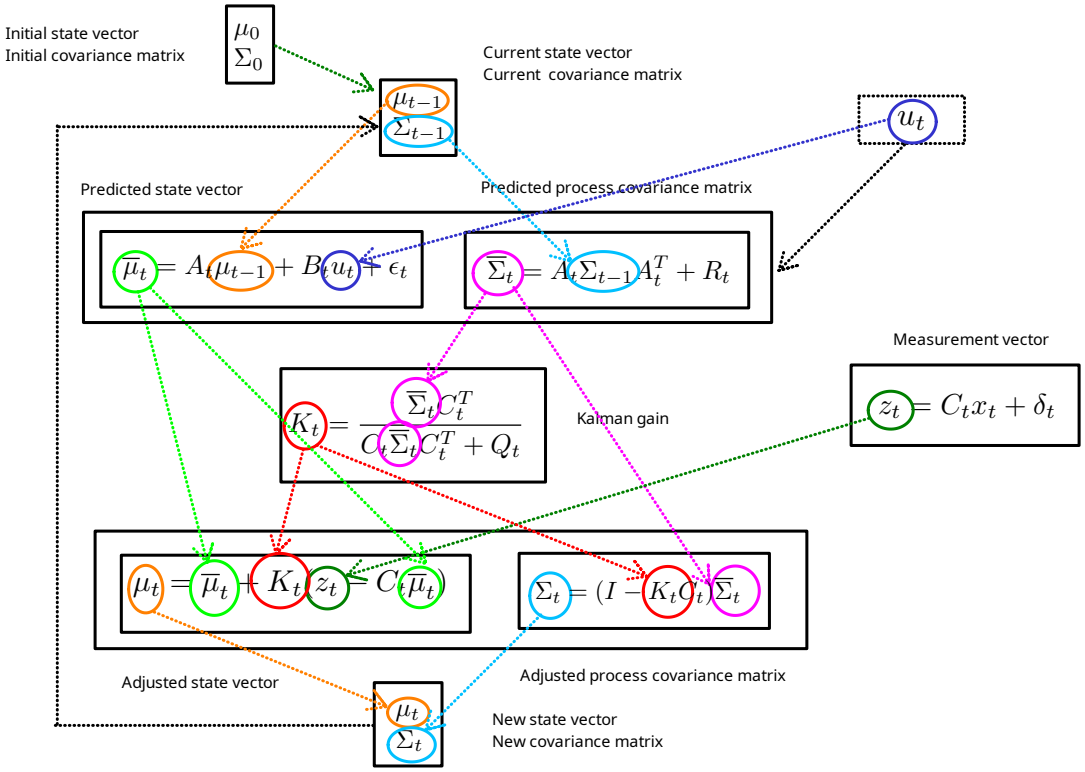
10.3. HET CONTROL UPDATE-MODEL

Is er niet iets meer te zeggen over het control update model en met name de matrices A en B? Jazeker.

Zoals al eerder genoemd bestaat de control update van een proces uit 3 elementen:

- de natuurlijke ontwikkeling van de state,
- de ontwikkeling van de state door de update,
- de noise in het proces.

Laten we dit eens bekijken voor een veel voorkomend Kalman-filter probleem: het inschatten van de plaats en snelheid van bewegend lichaam. En om het initieel niet te complex te maken kijken we in eerste instantie naar een ééndimensionaal probleem van een voertuig dat zich in de x-richting verplaatst en eenparig versneld met een versnelling a.



Figuur 10.4.: Flow van de variabelen in een Kalman-filter

Het eerste uitgangspunt hierbij is de natuurkundige formule die vrijwel iedereen die natuurkunde gedaan heeft op de middelbare school wel eens gezien heeft[5]:

$x_t = x_0 + v_0 t + \frac{1}{2}at^2$

De formule zegt dat de plaats van een lichaam op tijdstip t (x_t), de som is van de initiële plaats x_0, de verplaatsing als gevolg van de initiële snelheid v_0 en tijd t, en tot slot de verplaatsing als gevolg van de versnelling a en de tijd in het kwadraat, t^2.

De tweede formule, die toen wellicht ook voorbij is gekomen, is de formule om de snelheid te berekenen:

$v_t = v_0 + at$

Deze formule zegt dat de snelheid bij een versnelling op tijdstip t (v_t), de som is van de initiële snelheid en de toename van de snelheid als gevolg van de versnelling gedurende de tijd t.

Het bepalen van de diverse matrices is nu een kwestie van op listige wijze de matrices zo te kiezen dat de de twee formules tot stand komen[6]. De matrices voor μ_t en u_t liggen voor de hand:

$\mu_t = \begin{bmatrix} x \\ v \end{bmatrix}$ waarbij x de positie is en v de snelheid.

$u_t = \begin{bmatrix} a \end{bmatrix}$ waarbij a de versnelling is.

Dan de matrices A en B. Vaak zien deze matrices eruit als een indentiteitsmatrix met op enkele plaatsen in plaats van een 0 een formule waarbij tijd een rol speelt. Dat is op zich logisch: bij een Kalman-filter komen de metingen immers met een bepaalde frequentie binnen. Dat leidt er toe dat (het belief in) de state telkens met een Δt wordt berekend.

In dit geval ziet de matrix A er als volgt uit: $A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$

Wat hieruit af te leiden is, is dat de natuurlijke ontwikkeling van de x-positie lineair afhankelijk is van de tijd en dat de snelheid niet wijzigt.

Als we de vermenigvuldiging van A en μ uitschrijven leidt dit tot het volgende:

5 Soms wordt ook de notatie $x_t = x_0 + \dot{x}t + \frac{1}{2}\ddot{x}t^2$ gebruikt. Hierbij is $\dot{x}$ natuurlijk v_0 en $\ddot{x}$ is dan a

6 In de diverse matrices laat ik in dit deel de tijd-suffixen weg.

$A\mu = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} x + v\Delta t \\ 0 + v \end{bmatrix}$

Wat hier staat is dat de nieuwe x-positie de som is van de oude positie en de afgelegde weg in Δt met snelheid v en dat de nieuwe snelheid gelijk is aan de oude snelheid. Het eerste is niet echt verrassend maar het tweede wellicht wel: we versnellen immers dus de snelheid zou toch moeten wijzigen? We zullen zien dat dit in de update wordt geregeld.

In dit geval ziet de matrix B er als volgt uit: $B = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix}$

Wat hieruit af te leiden is, is dat de ontwikkeling van de x-positie door de update kwadratisch afhankelijk is van de tijd en dat die van de snelheid lineair afhankelijk.

Als we de vermenigvuldiging van BA en μ uitschrijven leidt dit tot het volgende:

$Bu = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} \begin{bmatrix} a \end{bmatrix} = \begin{bmatrix} \frac{1}{2}a\Delta t^2 \\ a\Delta t \end{bmatrix}$

Wat hier staat is dat de bijdrage aan de nieuwe x-positie als gevolg van de update (versnelling) berekend kan worden door $\frac{1}{2}a\Delta t^2$. De snelheid wijzigt als gevolg van de versnellen met $a\Delta t$.

Als we $\mu = A\mu + Bu$ volledig uitschrijven (we laten ϵ even weg) dan zien we het volgende:

$\mu_t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} \begin{bmatrix} a \end{bmatrix} = \begin{bmatrix} x + v\Delta t \\ 0 + v \end{bmatrix} + \begin{bmatrix} \frac{1}{2}a\Delta t^2 \\ a\Delta t \end{bmatrix} = \begin{bmatrix} x + v\Delta t + \frac{1}{2}a\Delta t^2 \\ v + a\Delta t \end{bmatrix}$

Wat hier staat is dat de nieuwe x-positie uitgerekend wordt met behulp van $x + v\Delta t + \frac{1}{2}a\Delta t^2$ en dat de nieuwe snelheid uitgerekend wordt met behulp van $v + a\Delta t$. En deze formules komen overeen met de middelbare school formules zoals eerder genoemd.

10.4. HET CONTROL UPDATE-MODEL, VERVOLG

10.4.1. 2-DIMENSIONAAL

Het uitbreiden van het voorgaande naar 2 dimensies is niet echt ingewikkeld. Feitelijk worden alleen de matrices groter. De matrices voor μ en u worden in dit geval:

$\mu = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}$ waarbij x en y de posities zijn en v_x en vv_y de snelheid in de respectievelijkr richtingen.

$u = \begin{bmatrix} a_x \\ a_y \end{bmatrix}$ waarbij a_x en a_y de versnellingen zijn in de respectievelijke richtingen.

De matrix A ziet er als volgt uit: $A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

De vermenigvuldiging van A en μ wordt dan:

$A\mu = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} = \begin{bmatrix} x + 0 + v_x\Delta t + 0 \\ 0 + y + 0 + v_y\Delta t \\ 0 + 0 + v_x + 0 \\ 0 + 0 + 0 + v_y \end{bmatrix} = \begin{bmatrix} x + v_x\Delta t \\ y + v_y\Delta t \\ v_x \\ v_y \end{bmatrix}$

De matrix B ziet er als volgt uit: $B = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}$

De vermenigvuldiging van B en u wordt dan:

$Bu = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} = \begin{bmatrix} \frac{1}{2}a_x\Delta t^2 + 0 \\ \frac{1}{2}a_y\Delta t^2 + 0 \\ 0 + a_x\Delta t \\ 0 + a_y\Delta t \end{bmatrix} = \begin{bmatrix} \frac{1}{2}a_x\Delta t^2 \\ \frac{1}{2}a_y\Delta t^2 \\ a_x\Delta t \\ a_y\Delta t \end{bmatrix}$

De volledige berekening wordt dan:

$$\mu = A\mu + Bu = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix}$$

$$= \begin{bmatrix} x + v_x\Delta t \\ y + v_y\Delta t \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} \frac{1}{2}a_x\Delta t^2 \\ \frac{1}{2}a_y\Delta t^2 \\ a_x\Delta t \\ a_y\Delta t \end{bmatrix} = \begin{bmatrix} x + v_x\Delta t + \frac{1}{2}a_x\Delta t^2 \\ y + v_y\Delta t + \frac{1}{2}a_y\Delta t^2 \\ v_x + a_x\Delta t \\ v_y + a_y\Delta t \end{bmatrix}$$

10.4.2. 3-DIMENSIONAAL

Dan zonder veel omhaal voor 3 dimensies.

$$\mu = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$$

$$= \begin{bmatrix} x + v_x\Delta t \\ y + v_y\Delta t \\ z + v_z\Delta t \\ v_x \\ v_y \\ v_z \end{bmatrix} + \begin{bmatrix} a_x\frac{1}{2}\Delta t^2 \\ a_y\frac{1}{2}\Delta t^2 \\ a_z\frac{1}{2}\Delta t^2 \\ a_x\Delta t \\ a_y\Delta t \\ z_y\Delta t \end{bmatrix} = \begin{bmatrix} x + v_x\Delta t + \frac{1}{2}a_x\Delta t^2 \\ y + v_y\Delta t + \frac{1}{2}a_y\Delta t^2 \\ z + v_z\Delta t + \frac{1}{2}a_z\Delta t^2 \\ v_x + a_x\Delta t \\ v_y + a_y\Delta t \\ v_z + a_z\Delta t \end{bmatrix}$$

10.4.3. 2-DIMENSIONAAL, ALTERNATIEVE STATE MATRIX

In het voorgaande hebben we telkens een μ gebruikt met de positie aan de bovenkant van de vector en de versnelling aan de onderkant. Dit is een veel voorkomende indeling van μ. Maar een andere indeling die je tegen kan komen is een groepering naar dimensie. Maakt dat wat uit? Nee, dan blijkt dat we de voorgaande vermenigvuldigingen eigenlijk nog steeds kunnen doen maar dat we dan alleen de matrices iets anders vorm moeten geven. De matrices voor μ en u worden in dit geval:

$$\mu = \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix}$$

$$u = \begin{bmatrix} a_x \\ a_y \end{bmatrix}$$

De matrix A ziet er als volgt uit: $A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$

De vermenigvuldiging van A en μ wordt dan:

$$A\mu = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix} = \begin{bmatrix} x + v_x\Delta t + 0 + 0 \\ 0 + v_x + 0 + 0 \\ 0 + 0 + y + v_y\Delta t \\ 0 + 0 + 0 + v_y \end{bmatrix} = \begin{bmatrix} x + v_x\Delta t \\ v_x \\ y + v_y\Delta t \\ v_y \end{bmatrix}$$

De matrix B ziet er als volgt uit: $B = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{bmatrix}$

De vermenigvuldiging van B en u wordt dan:

$$Bu = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} = \begin{bmatrix} \frac{1}{2}a_x\Delta t^2 + 0 \\ a_x\Delta t + 0 \\ 0 + \frac{1}{2}a_y\Delta t^2 \\ 0 + a_y\Delta t \end{bmatrix} = \begin{bmatrix} \frac{1}{2}a_x\Delta t^2 \\ a_x\Delta t \\ \frac{1}{2}a_y\Delta t^2 \\ a_y\Delta t \end{bmatrix}$$

De volledige berekening wordt dan:

$$\mu = A\mu + Bu = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix}$$

$$= \begin{bmatrix} x + v_x\Delta t \\ v_x \\ y + v_y\Delta t \\ v_y \end{bmatrix} + \begin{bmatrix} \frac{1}{2}a_x\Delta t^2 \\ a_x\Delta t \\ \frac{1}{2}a_y\Delta t^2 \\ a_y\Delta t \end{bmatrix} = \begin{bmatrix} x + v_x\Delta t + \frac{1}{2}a_x\Delta t^2 \\ v_x + a_x\Delta t \\ y + v_y\Delta t + \frac{1}{2}a_y\Delta t^2 \\ v_y + a_y\Delta t \end{bmatrix}$$

10.5. COVARANTIE

De covariantie tussen twee variabelen geeft aan in hoeverre de twee variabelen (lineair) met elkaar samenhangen.

10.5.1. HERINNERT U ZICH DEZE NOG?

Een korte herhaling van wat er IoT al voorbij gekomen met betrekking statistiek.

Een individuele meting: x_i

Het gemiddelde van een reeks metingen: $\overline{x} = \frac{\sum_{i=1}^{n} x_i}{N}$

Afwijking van het gemiddelde: $x_i - \overline{x}$

Het kwadraat van de afwijkingen $(x_i - \overline{x})^2$

Variantie van de gehele populatie: $\sigma_x^2 = \frac{\sum_{i=1}^{N} (x_i - \overline{x})^2}{N}$

Variantie van een steekproef: $\sigma_x^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}$

Standaarddeviatie van de gehele populatie: $\sigma_x = \sqrt{\sigma_x^2} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \overline{x})^2}{N}}$

Standaarddeviatie van een steekproef: $\sigma_x = \sqrt{\sigma_x^2} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}}$

10.5.1.1. VOORBEELD

Een klein voorbeeld van het berekenen van de variantie en standaardeviatie voor een gehele populatie.

Stel in een klas zijn de volgende cijfers gehaald: $8, 4, 6, 2, 7, 9, 8, 5, 4, 7$.

Dan is het gemiddelde cijfer: $\frac{\sum_{i=1}^{n} x_i}{N} = \frac{8+4+6+2+7+9+8+5+4+7}{10} = 6$

Dan zijn de afwijkingen van het gemiddelde:

$8 - 6 = 2$
$4 - 6 = -2$
$6 - 6 = 0$
$2 - 6 = -4$
$7 - 6 = 1$
$9 - 6 = 3$
$8 - 6 = 2$
$5 - 6 = -1$
$4 - 6 = -2$
$7 - 6 = 1$

Dan is de variantie:

$$\sigma_x^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{N} = \frac{(2)^2+(-2)^2+(0)^2+(-4)^2+(1)^2+(3)^2+(2)^2+(-1)^2+(-2)^2+(-1)^2}{10} = \frac{4+4+0+16+1+9+4+1+4+1}{10} = 4.4$$

tot slot is dan de standaarddeviatie:
$\sigma_x = \sqrt{\sigma_x^2} = \sqrt{4.4} = 2.1$

Als het bovenstaande een steekproef betrof dan moet de $N = 10$ vervangen worden door $n = 10 - 1$.

10.5.2. COVARIANTIE

Zoals gezegd, de covariantie tussen twee variabelen geeft aan in hoeverre de twee variabelen (lineair) met elkaar samenhangen. Als de covariantie groter dan 0 is dan geldt dat als de ene variable groter wordt dan wordt de andere variabele ook groter. Hetzelfde geldt voor kleiner worden: als de ene variable kleiner wordt dan wordt de andere variabele ook kleiner. Als de covariantie 0 is dan hebben de beide variabelen geen verband. Het veranderen van de grootte van de ene variabele leidt niet tot het veranderen van de ander variabele. Als de covariantie kleiner is dan 0 dan geldt dat als de ene variabele groter wordt de andere variabele kleiner wordt en omgekeerd.

Covariantie is daarmee te vergelijken met de correlatiecoëfficiënt[7]. Maar daar waar de correlatiecoëfficiënt gestandaardiseerd is tussen -1 en 1 is de covariantie niet gestandaardiseerd en kan dus beginsel iedere waarde hebben.

Net als bij de variantie en standaard deviatiatie wordt een onderscheid gemaakt tussen die van de totale populatie en die van een steekproef.

Covariantie tussen twee variabelen x en y in de totale populatie wordt als volgt berekend:

$$\sigma_{xy} = \sigma_x\sigma_y = \frac{\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})}{N}$$

Covariantie tussen twee variabelen x en y in een steekproef wordt als volgt berekend:

$$\sigma_{xy} = \sigma_x\sigma_y = \frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{n-1}$$

De covariantie tussen variabele en zichzelf is dus $\sigma_x\sigma_x = \sigma_x^2$ en daarmee de variantie van die variabele.

10.5.3. COVARANTIEMATRIX

Een covariantiematrix is aan vierkante matrix met op de diagonaal van linksboven naar rechtonder de variantie van de variabele en in de overige cellen de covariantie van de twee variabelen. Een covariantiematrix is symmetrisch in de diagonaal.

Drie covariantiematrices van met respektievelijk 1,2 en 3 variabelen:

1. $$\left[\sigma_x^2\right] = \left[\frac{\sum_{i=1}^{n}(x_i-\overline{x})^2}{N}\right]$$

2. $$\begin{bmatrix}\sigma_x^2 & \sigma_x\sigma_y\\ \sigma_y\sigma_x & \sigma_y^2\end{bmatrix} = \begin{bmatrix}\frac{\sum_{i=1}^{n}(x_i-\overline{x})^2}{N} & \frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{N}\\ \frac{\sum_{i=1}^{n}(y_i-\overline{y})(x_i-\overline{x})}{N} & \frac{\sum_{i=1}^{n}(y_i-\overline{y})^2}{N}\end{bmatrix}$$

3. $$\begin{bmatrix}\sigma_x^2 & \sigma_x\sigma_y & \sigma_x\sigma_z\\ \sigma_y\sigma_x & \sigma_y^2 & \sigma_y\sigma_z\\ \sigma_z\sigma_x & \sigma_z\sigma_y & \sigma_z^2\end{bmatrix} = \begin{bmatrix}\frac{\sum_{i=1}^{n}(x_i-\overline{x})^2}{N} & \frac{\sum_{i=1}^{n}(x_i-\overline{x})(y_i-\overline{y})}{N} & \frac{\sum_{i=1}^{n}(x_i-\overline{x})(z_i-\overline{z})}{N}\\ \frac{\sum_{i=1}^{n}(y_i-\overline{y})(x_i-\overline{x})}{N} & \frac{\sum_{i=1}^{n}(y_i-\overline{y})^2}{N} & \frac{\sum_{i=1}^{n}(y_i-\overline{y})(z_i-\overline{z})}{N}\\ \frac{\sum_{i=1}^{n}(z_i-\overline{z})(z_i-\overline{x})}{N} & \frac{\sum_{i=1}^{n}(z_i-\overline{z})(y_i-\overline{y})}{N} & \frac{\sum_{i=1}^{n}(z_i-\overline{z})^2}{N}\end{bmatrix}$$

7 Sterker nog: correlatie is de covariantie tussen de twee variabelen gedeeld door het product van de standaarddeviaties van beide variabelen.

10.5.3.1. VOORBEELD

Hier een voorbeeld voor het berekenen van een 3×3 covariantiematrix. Bij het berekenen ervan doen we het op een iets andere manier dan je wellicht zou verwachten.

Gegeven zijn de (wederom) 5 cijfers van een aantal studenten maar nu voor 3 verschillende vakken, wiskunde, natuurkunde, engels: $\begin{bmatrix} 90 & 80 & 40 \\ 90 & 60 & 80 \\ 30 & 50 & 70 \\ 30 & 40 & 70 \\ 30 & 30 & 90 \end{bmatrix}$

Het is mogelijk om met behulp van wat simpele matrix-berekeningen een covariantiematrix uit te rekenen. Dat gaat als volgt. Eerst berekenen we een deviatie-matrix: $a = A - [1] \cdot A \cdot \frac{1}{n}$ waarbij n het aantal rijen in de matrix is en $[1]$ een $n \times n$ matrix gevuld met 1-tjes. Vervolgens is de covariantiematrix $\frac{1}{n-1} a^T a$.

Eerst de deviatie-matrix:

$$a = \begin{bmatrix} 90 & 80 & 40 \\ 90 & 60 & 80 \\ 30 & 50 & 70 \\ 30 & 40 & 70 \\ 30 & 30 & 90 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 90 & 80 & 40 \\ 90 & 60 & 80 \\ 30 & 50 & 70 \\ 30 & 40 & 70 \\ 30 & 30 & 90 \end{bmatrix} \frac{1}{5} = \begin{bmatrix} 90 & 80 & 40 \\ 90 & 60 & 80 \\ 30 & 50 & 70 \\ 30 & 40 & 70 \\ 30 & 30 & 90 \end{bmatrix} - \begin{bmatrix} 270 & 260 & 350 \\ 270 & 260 & 350 \\ 270 & 260 & 350 \\ 270 & 260 & 350 \\ 270 & 260 & 350 \end{bmatrix} \frac{1}{5} =$$

$$\begin{bmatrix} 90 & 80 & 40 \\ 90 & 60 & 80 \\ 30 & 50 & 70 \\ 30 & 40 & 70 \\ 30 & 30 & 90 \end{bmatrix} - \begin{bmatrix} 54 & 52 & 70 \\ 54 & 52 & 70 \\ 54 & 52 & 70 \\ 54 & 52 & 70 \\ 54 & 52 & 70 \end{bmatrix} = \begin{bmatrix} 36 & 28 & -30 \\ 36 & 8 & 10 \\ -24 & -2 & 0 \\ -24 & -12 & 0 \\ -24 & -22 & 20 \end{bmatrix}$$

Vervolgens de covariantiematrix:

$$\frac{1}{n-1} a^T a = \frac{1}{5-1} \begin{bmatrix} 36 & 36 & -24 & -24- & -24 \\ 28 & 8 & -2 & -12 & -22 \\ -30 & 10 & 0 & 0 & 20 \end{bmatrix} \begin{bmatrix} 36 & 28 & -30 \\ 36 & 8 & 10 \\ -24 & -2 & 0 \\ -24 & -12 & 0 \\ -24 & -22 & 20 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 4320 & 2160 & -1200 \\ 2160 & 1480 & -1200 \\ -1200 & -1200 & 1400 \end{bmatrix} =$$

$$\begin{bmatrix} 1080 & 540 & -300 \\ 540 & 370 & -300 \\ -300 & -300 & 350 \end{bmatrix}$$

En hierbij zijn de verbanden dus als volgt (w = wiskunde, n = natuurkunde, e = engels):

$$\begin{bmatrix} ww & wn & we \\ nw & nn & ne \\ ew & en & ee \end{bmatrix}$$

En zoals je kan zien is het goed zijn in exacte vakken een reden om niet zo goed in de talen te zijn...

11. PARTICLE-FILTER

Zie ook Wikipedia:

- Particle-filter: Particle filter
- Monte Carlo localization: Monte Carlo localization

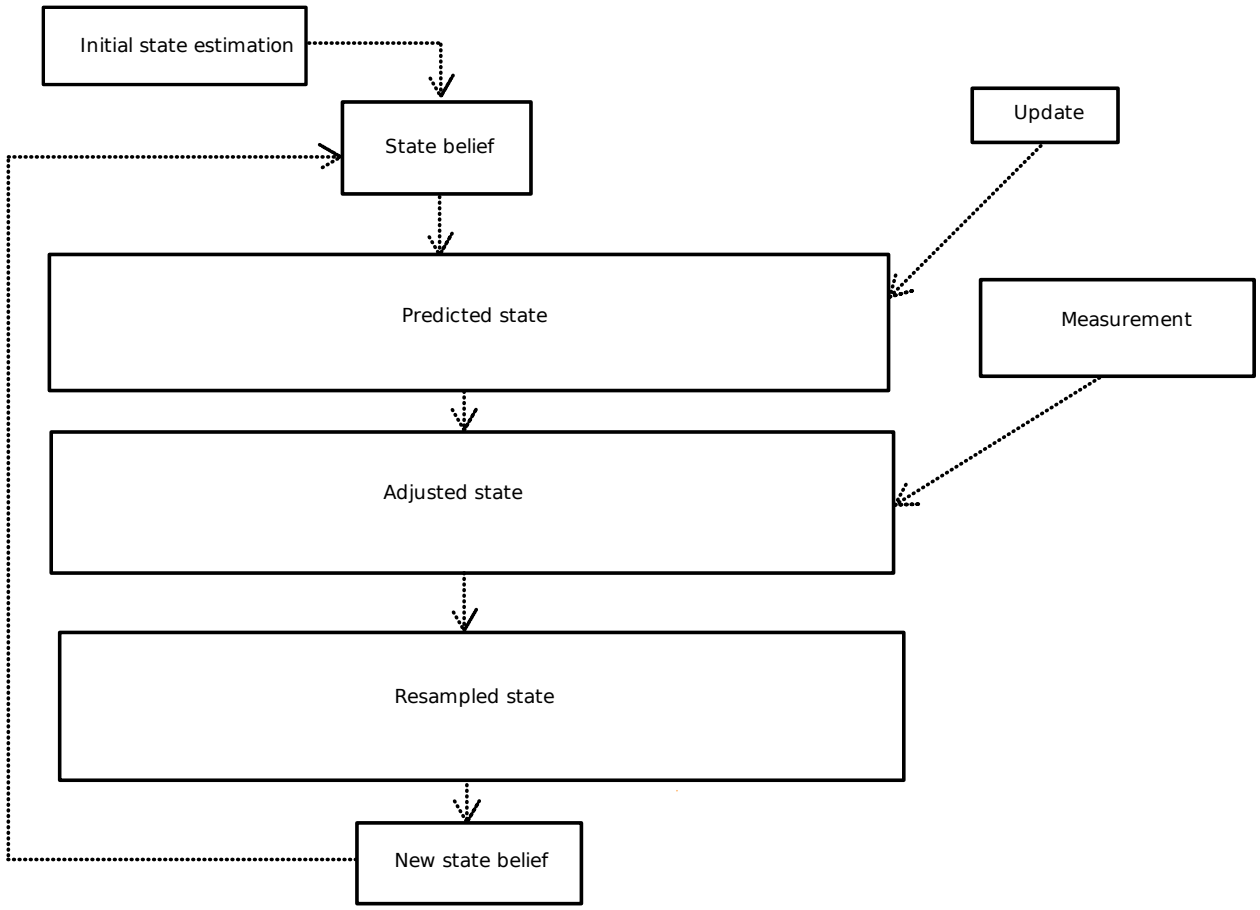
11.1. INLEIDING

Een particle-filter is een filter dat met behulp van particles (samples of steekproeven) een willekeurige kansverdeling probeert te benaderen en op basis daarvan de state probeert te vinden. In tegenstelling tot het Kalman-filter is de kansverdeling van particle-filter een multimodale kansverdeling: meerdere states zijn met een gelijke kans mogelijk. In figuur 11.1 staat de werking van het particle-filter schematisch weergegeven.

Zoals een vergelijking van figuur 11.1 en figuur 9.1 laat zien gaat het particle-filter van dezelfde twee fundamentele stappen uit als het Bayes-filter maar voegt een extra stap, het resamplen, toe:

1. de *control update*: voorspel (het geloof in) de nieuwe toestand ($\overline{\chi}_t$) op basis van de vorige toestand (χ_{t-1}) en de update u_t,
2. de *measurement update*: berekenen een update van (het geloof in) de nieuwe toestand ($\overline{\chi}_t$), gegeven de waarneming (z_t),
3. het *resamplen*: berekenen (het geloof in) de nieuwe toestand (χ_t) op basis van kansverdeling van (het geloof in) de nieuwe toestand ($\overline{\chi}_t$).

In de robotica wordt het particle-filter vaak gebruikt voor het bepalen van de plaats en orientatie van een robot gegeven een kaart van de omgeving. In dat geval wordt het ook wel *Monte Carlo localization* genoemd. Het basisidee van het particle-filter in een Monte Carlo localization is het volgende. We gaan uit van een robot met een kaart van de omgeving en een lidar als sensor. Initieel worden er met een uniforme random verdeling een groot aantal punten of coördinaten (particles) gekozen op de kaart. Voor elk van die punten wordt berekent hoe de point cloud van de lidar eruit zou zien als de robot op die plaats zou staan. Die point clouds worden vervolgens vergeleken met de meting die de robot met zijn eigen lidar heeft gedaan. De vergelijkingen leveren voor iedere particle een mate van gelijkheid op met de meting. Het idee is vervolgens dat de robot daar is waar het particle het best lijkt op de gemeten point cloud. Omdat het zeer goed mogelijk is dat meerdere particles “even goed” lijken wordt dit na een (kleine) verplaatsing opnieuw gedaan. Hierbij wordt niet telkens een nieuwe random verzameling particles gebruikt maar worden vooral de meest relevante oude particles opnieuw gebruikt. Na een aantal iteraties zal er een cluster van dicht bij elkaar liggende particles overblijven. En daar is de robot dan.



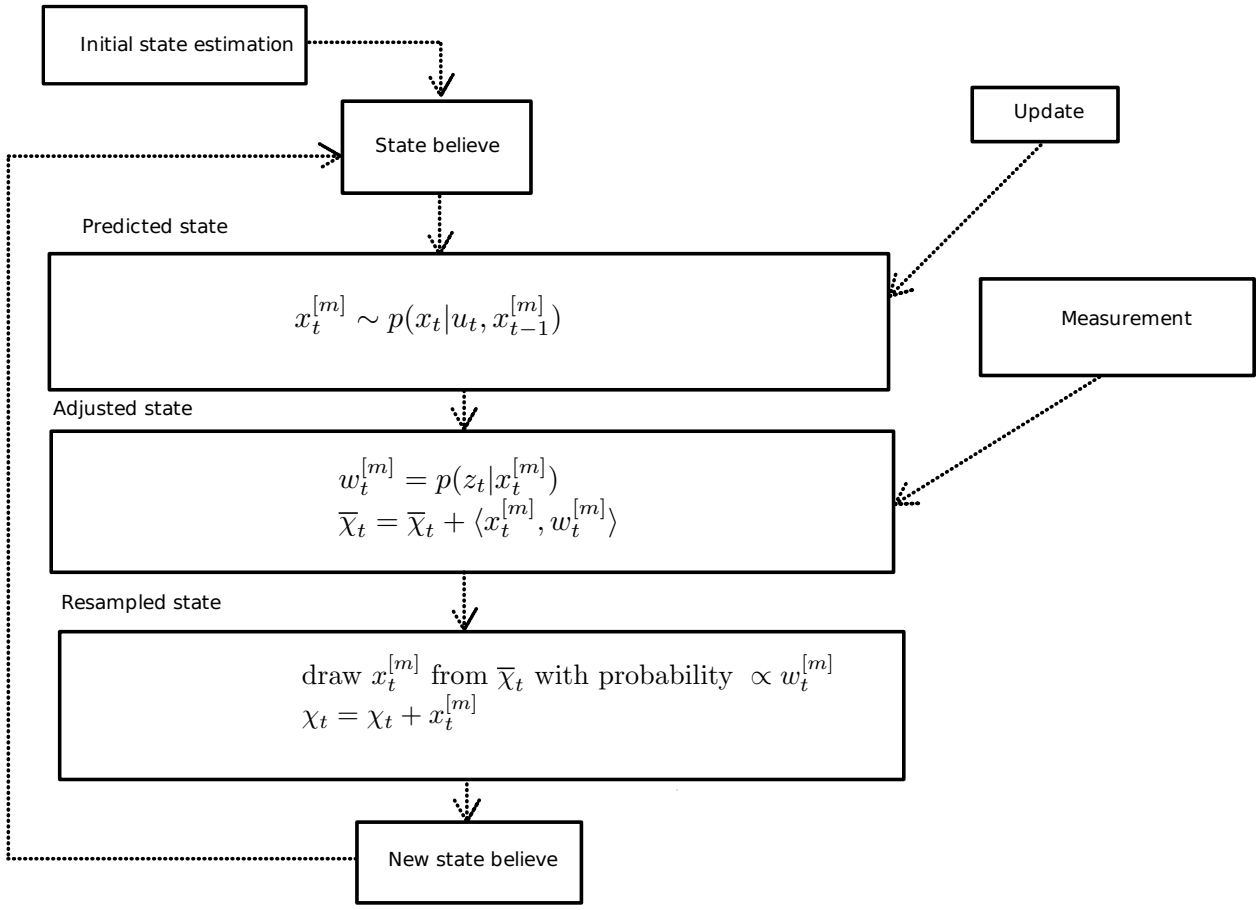
Figuur 11.1.: Overzicht particle-filter

11.2. HET PARTICLE-FILTER ALGORITME

Het algoritme van het particle-filter is als volgt:

input χ_{t-1}, u_t, z_t
$\overline{\chi}_t = \chi_t = 0$
for m = 1 to M do
    // Control update
    $x_t^{[m]} \sim p(x_t|u_t, x_{t-1}^{[m]})$
    // Measurement update van $\overline{\chi}_t$!!
    $w_t^{[m]} = p(z_t|x_t^{[m]})$
    $\overline{\chi}_t = \overline{\chi}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
endfor
// Resample the space
for m = 1 to M do
    draw $x_t^{[m]}$ from $\overline{\chi}_t$ with probability $\propto w_t^{[m]}$
    // Construeer χ_t
    $\chi_t = \chi_t + x_t^{[m]}$
endfor
return χ_t

In figuur 11.2 is de plaats van de diverse formules weergegeven in de flow van het algoritme. In het plaatje gaan de twee for-loops uit het algoritme helaas verloren.



Figuur 11.2.: Overzicht particle-filter met formules

11.2.1. INITIALISATIE VAN HET ALGORITME

Bij de initialisatie van het algoritme worden de particles met een uniforme random kansverdeling op de kaart geplaatst[1]. Voor het aantal particles dat gebruikt moet worden geldt: hoe meer, hoe beter, i.e. hoe nauwkeuriger de plaatsbepaling. Dit gaat helaas wel ten koste van veel rekencapaciteit. In de praktijk zal er op empirische gronden een afweging gemaakt moeten worden tussen de gewenste nauwkeurigheid en snelheid.

11.2.2. INITIALISATIE VAN DE LOOP

Bij de initialisatie van de loop worden zowel het update-belief als het uiteindelijke belief op "0" geinitialiseerd. Dat betekent dat met name de weight van de vorige iteratie niet meegenomen worden in deze iteratie. Alle particles uit de vorige iteratie hebben een gelijke kans om het particle met de grootste kans te worden.

11.2.3. CONTROL UPDATE

In de control update wordt de vorige state ge-update op basis van de control update.

$x_t^{[m]} \sim p(x_t|u_t, x_{t-1}^{[m]})$

Voor iedere particle in de vorige state wordt een nieuw particle getrokken uit een kansverdeling op basis van de update. Hoe deze kansverdeling eruit zie hangt van de concrete situatie af.

Als het particle-filter gebruikt wordt in een Monte Carlo localization dan wordt de update, i.e. de verplaatsing samen met een error (onzekerheid), op ieder particle in χ_{t-1} toegepast. In effect betekent dit dus dat het belief zich in de richting van de update verplaatst met een bepaalde error. De error wordt veelal bepaald op basis van de onnauwkeurigheid van de actuatoren die verantwoordelijk voor zijn voor de update. Veelal zal dit een trekking uit een normaalverdeling met een gemiddelde van 0 en een voor de actuator gegeven standaarddeviatie zijn.

11.2.4. MEASUREMENT UPDATE

Voor ieder particle wordt een *weight* of *importance* factor bepaald en wordt deze toegevoegd aan het belief. Hiemee wordt het update-belief dus "ge-update" op grond van de measurement.

$w_t^{[m]} = p(z_t|x_t^{[m]})$

De weight factor geeft aan wat de kans is dat de waarneming bij het gegeven particle hoort. Naarmate deze hoger is is de kans dat de robot op de plaats van dat particle is groter.

In een Monte Carlo localization met behulp van een lidar zal voor ieder particle een "virtuele lidar scan" gemaakt worden, e.g. met behulp van ray-tracing en de kaart, die vervolgens vergeleken wordt met de de meting. Naarmate de virtuele lidar scan meer overeenkomt met de meting is de weight factor groter.

Aannemende dat alle metingen van een lidar dezelfde orientatie hebben, i.e. meting 0 naar het noorden of iets dergelijks kunnen de volgende maten gebruikt worden:

- de som van de kwadraat van vectoreindverschillen,
- de cosine simularity van de som van de $[x, y]^T$-vectoren in combinatie met het verschil in de lengte van de eindvector,

$\overline{\chi}_t = \overline{\chi}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

Tenslotte wordt een tuple van het particle $x_t^{[m]}$ met zijn gewicht $w_t^{[m]}$ toegevoegd aan het belief $\overline{\chi}_t$[2].

11.2.5. RESAMPLING

Bij het resamplen wordt het uiteindelijke definitieve belief berekend.

draw $x_t^{[m]}$ from $\overline{\chi}_t$ with probability $\propto w_t^{[m]}$

1 Als er veel informatie bekent is over de initiële locatie van de robot dan kan ervoor gekozen worden om die te verwerken in de plaatsing van de particles.

2 Eigenlijk staat er natuurlijk dat het nieuwe belief het oude belief plus het nieuwe particle met zijn weight is...

$\chi_t = \chi_t + x_t^{[m]}$

Tijdens de reampling worden uit $\overline{\chi}_t$ met teruglegging m particles getrokken waarbij de kans op trekken van een particle gelijk aan de weigth van het particle. Dit betekent in effect dat de kansrijke particles de minder kansrijke particles in het uiteindelijk belief verdringen.

BIBLIOGRAFIE

[1] David Poole en Alan Mackworth. *Artificial Intelligence: Foundations of Computational Agents*. 2de ed. Cambridge, UK: Cambridge University Press, 2017. ISBN: 978-0-521-51900-7. URL: http://artint.info/2e/html/ArtInt2e.html.

[2] Sebastian Thrun, Wolfram Burgard en Dieter Fox. *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005. ISBN: 0262201623 9780262201629. URL: http://www.amazon.de/gp/product/0262201623/102-8479661-9831324?v=glance&n=283155&n=507846&s=books&v=glance.

A. OPGAVEN

A.1. INLEIDING

A.2. DIFFERENTIEREN

A.3. MATRICES

A.3.1. MATRIX VERMENIGVULDIGEN MET EEN MATRIX

A.3.2. MATRIX INVERTEREN

A.3.3.

A.4. LINEAIRE VERGELIJKINGEN

A.5. FORWARD KINEMATICS

A.6. INVERSE KINEMATICS

A.7. KANSREKENING

A.8. ROBOTMODEL

A.9. BAYESFILTER

A.9.1. DE BRANDBLUSROBOT

Naar aanleiding van de vele bosbranden in Europa heeft het park “de Hoge Veluwe” besloten een blusrobot aan te schaffen. De blusrobot is een autonoom rondrijdende robot die van boom tot boom gaat en kijkt of de boom in de brand staat. Als dat zo is dan blust de robot de boom, anders niet en gaat hij verder naar de volgende boom.

A.9.1.1. DE UPDATE- EN MEASUREMENTKANSEN

Zowel de actuatoren als sensoren van de robot zijn noisy, i.e. probabilistisch.

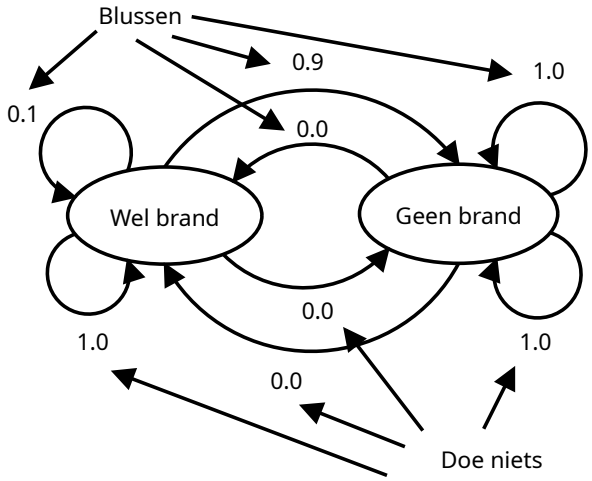
A.9.1.2. UPDATE-KANSEN

De robo kan twee dingen doen: de robot doet of niets, of gaat blussen.

Eerst blussen. Helaas voor de robot is het blussen niet geheel deterministisch:

$p(x_t = fire|u_t = extinguish\_fire, x_{t-1} = fire) = 0.1)$
$p(x_t = no\_fire|u_t = extinguish\_fire, x_{t-1} = fire) = 0.9)$
$p(x_t = fire|u_t = extinguish\_fire, x_{t-1} = no\_fire) = 0.0)$
$p(x_t = no\_fire|u_t = extinguish\_fire, x_{t-1} = no\_fire) = 1.0)$

Dan niets doen.



Figuur A.1.: De brand-blussende robot

$p(x_t = fire|u_t = do\_nothing, x_{t-1} = fire) = 1.0)$
$p(x_t = no\_fire|u_t = do\_nothing, x_{t-1} = fire) = 0.0)$
$p(x_t = fire|u_t = do\_nothing, x_{t-1} = no\_fire) = 0.0)$
$p(x_t = no\_fire|u_t = do\_nothing, x_{t-1} = no\_fire) = 1.0)$

Niets doen verandert uiteraard niet veel. Als de boom al in brand staat dan blijft die in brand staan. En als de boom niet in brand stond dan zal niets doen niet leiden tot een spontaan brandende boom.

A.9.1.3. MEASUREMENT-KANSEN

De robot kan met een warmtecamera zien of de boom in brand staat. Helaas is de camera noisy. Wat de camera meet is afhankelijk van de vraag of de boom daadwerkelijk in brand staat.

Staat de boom daadwerkelijk in brand dan zijn de kansen als volgt:

$p(z_t = see\_fire|x_t = fire) = 0.9)$
$p(z_t = see\_no\_fire|x_t = fire) = 0.1)$

Staat de boom evenwel niet in brand dan zijn de kansen als volgt:

$p(z_t = see\_fire|x_t = no\_fire) = 0.2)$
$p(z_t = see\_no\_fire|x_t = no\_fire) = 0.8)$

A.9.1.4. INITIAL STATE ESTIMATION

Als de robot opstart en bij een boom gaat staan heeft hij geen flauw idee of de boom in de hens staat of niet.

$bel(x_{t=0} = fire) = 0.5$
$bel(x_{t=0} = no\_fire) = 0.5$

A.9.1.5. VRAGEN

1. Bereken het voorspelde belief als de robot niets doet.
2. Bereken het uiteindelijke belief als de robot ziet dat de boom in brand staat.
3. Bereken het voorspelde belief als de robot besluit te gaan blussen.
4. Bereken het uiteindelijke belief als de robot ziet dat de boom niet meer in brand staat.

A.10. KALMANFILTER

A.11. PARTICLEFILTER