Cornelia Chrapal 160698
Lars Döhling 175094
- betriebspraktikanten -

1 Konventionen

1.1 Fehlermaß

Als Fehlermaß wird +/- 5% benutzt. Dies gilt sowohl für Winkel, als auch für Distanzen. Diese Wahl erscheint uns sinnvoll, da nur ein relatives Fehlermaß der Aufgabe angemessen ist und sich 5% als ausreichend herausgestellt hat. Der Wert lässt sich einfach in den entsprechenden Constraints anpassen.

1.2 Nullpunkte

Alle notwendigen Umrechnungsschritte zwischen verschiedenen Systemen sind jeweils direkt im Constraint definiert.

1.2.1 Spielfeld

Es wird das in der Aufgabenstellung beschrieben Koordinatensystem benutzt. Der Blickwinkel wird in Bogenmaß angegeben. Die positive x-Achse entspricht 0, die positive y-Achse $\pi/2$.

1.2.2 Bilder

Das Koordinatensystem ist analog zum Spielfeld, Bildmitte (0,0), oben rechts (+x,+y). Die Winkel sind ebenfalls im Bogenmaß. Pixel oberhalb der x-Achse haben einen positiven vertikalen Winkel, unterhalb einen negativen. Pixel rechts der y-Achse haben einen positiven horizontalen Winkel, links einen negativen.

1.2.3 Flaggenpositionen in den Bildern

Abweichend wird hier das in Grafikprogrammen üblich Koordinatensystem – Nullpunkt (0,0) in der linken oberen Ecke - verwendet.

2 Ideen

2.1 Kontinuierliche Domain versus Diskrete Domain

Da der Rechner nur sehr begrenzt in der Lage ist, kontinuierliche (unendliche) Domains zu speichern / zu bearbeiten, haben wir uns entschlossen, diese über diskrete Domains anzunähern. Dazu wird über die kontinuierliche Domain ein gleichförmiges Raster mit konfigurierbarem Abstand gelegt. Je kleiner der Rasterabstand ist, desto exakter kann die Originaldomain abgebildet werden.
Dadurch ergibt sich eine endliche Domain, welche gespeichert und bearbeitet werden kann. Unserer Meinung nach ergibt sich dadurch kein Nachteil in der Lösung der Aufgabe, da das verwendete Fehlermaß die Lücken in der Abtastung der Originaldomain kompensiert.

2.2 Bestimmung der Roboterposition

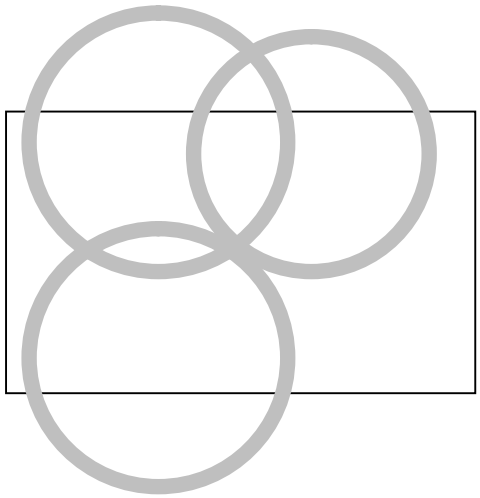
Aus den drei Bildern lässt sich die Entfernung des Roboters zu drei bekannten Flaggen abschätzen. Die dafür notwendigen Berechnungen wurden bereits in der letzten Serie skizziert und sollen hier nur noch einmal kurz angesprochen werden.

1. Aus der Breite des Bildes und dem Kameraöffnungswinkel lässt sich der virtuelle Bildabstand zur Bildebene berechnen
2. Damit lässt sich der Peilungswinkel zur Flagge im Bild bestimmen

3. Mit diesem Winkel wird wiederum der Abstand zur Flagge berechnet
4. Dieser Abstand wird über das Verhältnis FlaggeImBild/FlaggeInDerRealität in die reale Entfernung umgerechnet

Die genauen Formeln stehen in der `expression.config` und können dort nachgeschlagen werden.

Der Roboter muss sich auf den Kreisbahnen aller drei Flaggen befinden.



Die genauen Formeln stehen in der `constraints.config` und können dort nachgeschlagen werden.

Schlussendlich lässt sich der Zentroid der Schnittmenge bilden, welcher als Roboterposition aufgefasst werden kann.

Es besteht die Möglichkeit, sowohl das horizontale Ausmaß einer Flagge, als auch das Vertikale zur Berechnung des Abstands heranzuziehen. Wir benutzen beide und wichten das Ergebnis im Verhältnis Flaggenbreite/Flaggenhöhe.

2.3 Bestimmung des Roboterblickwinkels

Ausgehend von einer festen Roboterposition (Zentroid) lässt sich über die Peilungswinkel zu den Flaggen von dieser Position aus und den Peilungswinkel zu den Flaggen in den Bildern ein Bereich für den Blickwinkel des Roboters bestimmen. Im Prinzip handelt es sich dabei um vorzeichenkorrekte Additionen von Winkeln.

Hier lässt sich dann das arithmetische Mittel bestimmen, welches zusammen mit dem Zentroiden eine Pose bildet.

Im Prinzip wäre es auch machbar, für jede noch mögliche Positionen einen Blickwinkelbereich zu berechnen. Es stellt sich dann aber die Frage, wie man von diesem Punkt aus zu einer finalen Pose gelangt. Dazu müsste eine Position (x,y) als final deklariert werden und der Blickwinkelbereich für diese finale Position bestimmt werden. Nicht anderes machen wir ebenfalls.

Cornelia Chrapal 160698
Lars Döhling 175094
- betriebspraktikanten -

3 Constraint-Netz

3.1 Allgemein

C = { C_0, …, C_{40} } mit V = { V_0, …, V_{65} } mit

i	V_i	Dom(V_i)	Kommentar
0	`posX`	{ -3000, …, +3000 }	Position des Roboters
1	`posY`	{ -2000, …, +2000 }	
2	`theta`	{ 0, …, 2π }	Blickwinkel des Roboters
3	`kameraWinkelHorizontal`	{ 55/180*π }	Kameraöffnungswinkel
4	`kameraWinkelVertikal`	{ 44/180*π }	
5	`flaggeHoehe`	{ 400 }	Flaggenabmaße
6	`flaggeBreite`	{ 100 }	
7	`flaggeWichtungHorizontal`	{ 0, …, 1 }	Siehe 2.2
8	`flaggeWichtungVertikal`	{ 0, …, 1 }	
9	`flaggeNwX`	{ -1350 }	Koordinaten der Flaggen
10	`flaggeNwY`	{ 1800 + 150 }	
11	`flaggeNoX`	{ 1350 }	
12	`flaggeNoY`	{ 1800 + 150 }	
13	`flaggeSwX`	{ -1350 }	
14	`flaggeSwY`	{ -1800 – 150 }	
15	`bildLinksBreite`	{ 200, …, 210 }	Größe des linken Bildes
16	`bildLinksHoehe`	{ 150, … , 160 }	
17	`bildLinksFlaggeLinks`	{ 0, …, 209 }	Umriss der Flagge im linken Bild
18	`bildLinksFlaggeRechts`	{ 0, …, 209 }	
19	`bildLinksFlaggeOben`	{ 0, …, 159 }	
20	`bildLinksFlaggeUnten`	{ 0, …, 159 }	
21	`bildLinksFlaggeMitteHorizontal`	{ -104.5, …, 104.5 }	Flaggenmitte im linken Bild
22	`bildLinksFlaggeMitteVertikal`	{ -79.5, …, 79,5 }	
23	`bildLinksGroessenverhaeltnisHorizontal`	{ 0, …, 100 }	Größenverhältnisse im linken Bild (Siehe 2.2)
24	`bildLinksGroessenverhaeltnisVertikal`	{ 0, …, 100 }	
25	`bildLinksAbstandHorizontal`	{ 150, …, 250 }	Virtueller Abstand zur Bildebene im linken Bild
26	`bildLinksAbstandVertikal`	{ 150, …, 250 }	
27	`bildLinksFlaggeWinkelHorizontal`	{ -55/360*π, …, +55/360*π }	Winkel zur Flagge im linken Bild
28	`bildLinksFlaggeWinkelVertikal`	{ -44/360*π, …, +44/360*π }	
29	`bildLinksFlaggeAbstandHorizontal`	{ 150, …, 350 }	Virtueller Abstand zur Flagge im linken Bild
30	`bildLinksFlaggeAbstandVertikal`	{ 150, …, 350 }	
31	`abstandFlaggeSw`	{ 0, …, 8000 }	Realer Abstand zur SW Flagge

32	bildVornBreite	{ 200, …, 210 }	Größe des vorderen Bildes
33	bildVornHoehe	{ 150, … , 160 }	
34	bildVornFlaggeLinks	{ 0, …, 209 }	Umriss der Flagge im vorderen Bild
35	bildVornFlaggeRechts	{ 0, …, 209 }	
36	bildVornFlaggeOben	{ 0, …, 159 }	
37	bildVornFlaggeUnten	{ 0, …, 159 }	
38	bildVornFlaggeMitteHorizontal	{ -104.5, …, 104.5 }	Flaggenmitte im vorderen Bild
39	bildVornFlaggeMitteVertikal	{ -79.5, …, 79,5 }	
40	bildVornGroessenverhaeltnisHorizontal	{ 0, …, 100 }	Größenverhältnisse im vorderen Bild (Siehe 2.2)
41	bildVornGroessenverhaeltnisVertikal	{ 0, …, 100 }	
42	bildVornAbstandHorizontal	{ 150, …, 250 }	Virtueller Abstand zur Bildebene im vorderen Bild
43	bildVornAbstandVertikal	{ 150, …, 250 }	
44	bildVornFlaggeWinkelHorizontal	{ -55/360*π, …, +55/360*π }	Winkel zur Flagge im vorderen Bild
45	bildVornFlaggeWinkelVertikal	{ -44/360*π, …, +44/360*π }	
46	bildVornFlaggeAbstandHorizontal	{ 150, …, 350 }	Virtueller Abstand zur Flagge im vorderen Bild
47	bildVornFlaggeAbstandVertikal	{ 150, …, 350 }	
48	abstandFlaggeNw	{ 0, …, 8000 }	Realer Abstand zur NW Flagge
49	bildRechtsBreite	{ 200, …, 210 }	Größe des rechten Bildes
50	bildRechtsHoehe	{ 150, … , 160 }	
51	bildRechtsFlaggeLinks	{ 0, …, 209 }	Umriss der Flagge im rechten Bild
52	bildRechtsFlaggeRechts	{ 0, …, 209 }	
53	bildRechtsFlaggeOben	{ 0, …, 159 }	
54	bildRechtsFlaggeUnten	{ 0, …, 159 }	
55	bildRechtsFlaggeMitteHorizontal	{ -104.5, …, 104.5 }	Flaggenmitte im rechten Bild
56	bildRechtsFlaggeMitteVertikal	{ -79.5, …, 79,5 }	
57	bildRechtsGroessenverhaeltnisHorizontal	{ 0, …, 100 }	Größenverhältnisse im rechten Bild (Siehe 2.2)
58	bildRechtsGroessenverhaeltnisVertikal	{ 0, …, 100 }	
59	bildRechtsAbstandHorizontal	{ 150, …, 250 }	Virtueller Abstand zur Bildebene im rechten Bild
60	bildRechtsAbstandVertikal	{ 150, …, 250 }	
61	bildRechtsFlaggeWinkelHorizontal	{ -55/360*π, …, +55/360*π }	Winkel zur Flagge im rechten Bild
62	bildRechtsFlaggeWinkelVertikal	{ -44/360*π, …, +44/360*π }	

63	`bildRechtsFlaggeAbstandHorizontal`	{ 150, …, 350 }	Virtueller Abstand zur Flagge im rechten Bild
64	`bildRechtsFlaggeAbstandVertikal`	{ 150, …, 350 }	
65	`abstandFlaggeNo`	{ 0, …, 8000 }	Realer Abstand zur NO Flagge

und C_i = { $[v_0, \ldots, v_n] \in Dom(v_0) \times \ldots \times Dom(v_n)$: $expression(v_0,\ldots,v_n)$} mit

i	**C_i**
0	{ [flaggeWichtungHorizontal, flaggeBreite, flaggeHoehe] : flaggeWichtungHorizontal = flaggeBreite / (flaggeHoehe + flaggeBreite) }
1	{ [flaggeWichtungVertikal, flaggeBreite, flaggeHoehe] : flaggeWichtungVertikal = 1.0 – flaggeWichtungHorizontal }
2	{ [bildLinksFlaggeMitteHorizontal, bildLinksFlaggeLinks, bildLinksFlaggeRechts, bildLinksBreite] : bildLinksFlaggeMitteHorizontal = (bildLinksFlaggeLinks + bildLinksFlaggeRechts) / 2.0 - ((bildLinksBreite - 1) / 2.0) }
3	{ [bildLinksFlaggeMitteVertikal, bildLinksFlaggeOben, bildLinksFlaggeUnten, bildLinksHoehe] : bildLinksFlaggeMitteVertikal = ((bildLinksHoehe - 1) / 2.0) - ((bildLinksFlaggeOben + bildLinksFlaggeUnten) / 2.0) }
4	{ [bildLinksGroessenverhaeltnisHorizontal, flaggeBreite, bildLinksFlaggeRechts, bildLinksFlaggeLinks] : bildLinksGroessenverhaeltnisHorizontal = flaggeBreite / (bildLinksFlaggeRechts - bildLinksFlaggeLinks + 1) }
5	{ [bildLinksGroessenverhaeltnisVertikal, flaggeHoehe, bildLinksFlaggeUnten, bildLinksFlaggeOben] : bildLinksGroessenverhaeltnisVertikal = flaggeHoehe / (bildLinksFlaggeUnten - bildLinksFlaggeOben + 1) }
6	{ [bildLinksAbstandHorizontal, bildLinksBreite, kameraWinkelHorizontal] : bildLinksAbstandHorizontal = bildLinksBreite / 2.0 / TAN(kameraWinkelHorizontal/2.0) }
7	{ [bildLinksAbstandVertikal, bildLinksHoehe, kameraWinkelVertikal] : bildLinksAbstandVertikal = bildLinksHoehe / 2.0 / TAN(kameraWinkelVertikal/2.0) }
8	{ [bildLinksFlaggeWinkelHorizontal, bildLinksFlaggeMitteHorizontal, bildLinksAbstandHorizontal] : bildLinksFlaggeWinkelHorizontal = ARCTAN((bildLinksFlaggeMitteHorizontal) / bildLinksAbstandHorizontal) }
9	{ [bildLinksFlaggeWinkelVertikal, bildLinksFlaggeMitteVertikal, bildLinksAbstandVertikal] : bildLinksFlaggeWinkelVertikal = ARCTAN((bildLinksFlaggeMitteVertikal) / bildLinksAbstandVertikal) }
10	{ [bildLinksFlaggeAbstandHorizontal, bildLinksAbstandHorizontal, bildLinksFlaggeMitteHorizontal] : bildLinksFlaggeAbstandHorizontal = SQRT (POW(bildLinksAbstandHorizontal, 2) + POW(bildLinksFlaggeMitteHorizontal, 2)) }
11	{ [bildLinksFlaggeAbstandVertikal, bildLinksAbstandVertikal, bildLinksFlaggeMitteVertikal] : bildLinksFlaggeAbstandVertikal = SQRT (POW(bildLinksAbstandVertikal, 2) + POW(bildLinksFlaggeMitteVertikal, 2)) }
12	{ [abstandFlaggeSw, bildLinksFlaggeAbstandHorizontal, bildLinksGroessenverhaeltnisHorizontal, flaggeWichtungHorizontal, bildLinksFlaggeAbstandVertikal, bildLinksGroessenverhaeltnisVertikal, flaggeWichtungVertikal] : abstandFlaggeSw = (bildLinksFlaggeAbstandHorizontal * bildLinksGroessenverhaeltnisHorizontal * flaggeWichtungHorizontal) + (bildLinksFlaggeAbstandVertikal * bildLinksGroessenverhaeltnisVertikal * flaggeWichtungVertikal) }
13	{ [abstandFlaggeSw, posX, flaggeSwX, posY, flaggeSwY] : abstandFlaggeSw * 0.05 >= ABS(abstandFlaggeSw - SQRT(POW(posX-flaggeSwX, 2) + POW(posY-flaggeSwY, 2))) }

14	{ [bildVornFlaggeMitteHorizontal, bildVornFlaggeLinks, bildVornFlaggeRechts, bildVornBreite] : bildVornFlaggeMitteHorizontal = (bildVornFlaggeLinks + bildVornFlaggeRechts) / 2.0 - ((bildVornBreite - 1) / 2.0) }
15	{ [bildVornFlaggeMitteVertikal, bildVornFlaggeOben, bildVornFlaggeUnten, bildVornHoehe] : bildVornFlaggeMitteVertikal = ((bildVornHoehe - 1) / 2.0) - ((bildVornFlaggeOben + bildVornFlaggeUnten) / 2.0) }
16	{ [bildVornGroessenverhaeltnisHorizontal, flaggeBreite, bildVornFlaggeRechts, bildVornFlaggeLinks] : bildVornGroessenverhaeltnisHorizontal = flaggeBreite / (bildVornFlaggeRechts - bildVornFlaggeLinks + 1) }
17	{ [bildVornGroessenverhaeltnisVertikal, flaggeHoehe, bildVornFlaggeUnten, bildVornFlaggeOben] : bildVornGroessenverhaeltnisVertikal = flaggeHoehe / (bildVornFlaggeUnten - bildVornFlaggeOben + 1) }
18	{ [bildVornAbstandHorizontal, bildVornBreite, kameraWinkelHorizontal] : bildVornAbstandHorizontal = bildVornBreite / 2.0 / TAN(kameraWinkelHorizontal/2.0) }
19	{ [bildVornAbstandVertikal, bildVornHoehe, kameraWinkelVertikal] : bildVornAbstandVertikal = bildVornHoehe / 2.0 / TAN(kameraWinkelVertikal/2.0) }
20	{ [bildVornFlaggeWinkelHorizontal, bildVornFlaggeMitteHorizontal, bildVornAbstandHorizontal] : bildVornFlaggeWinkelHorizontal = ARCTAN((bildVornFlaggeMitteHorizontal) / bildVornAbstandHorizontal) }
21	{ [bildVornFlaggeWinkelVertikal, bildVornFlaggeMitteVertikal, bildVornAbstandVertikal] : bildVornFlaggeWinkelVertikal = ARCTAN((bildVornFlaggeMitteVertikal) / bildVornAbstandVertikal) }
22	{ [bildVornFlaggeAbstandHorizontal, bildVornAbstandHorizontal, bildVornFlaggeMitteHorizontal] : bildVornFlaggeAbstandHorizontal = SQRT (POW(bildVornAbstandHorizontal, 2) + POW(bildVornFlaggeMitteHorizontal, 2)) }
23	{ [bildVornFlaggeAbstandVertikal, bildVornAbstandVertikal, bildVornFlaggeMitteVertikal] : bildVornFlaggeAbstandVertikal = SQRT (POW(bildVornAbstandVertikal, 2) + POW(bildVornFlaggeMitteVertikal, 2)) }
24	{ [abstandFlaggeNw, bildVornFlaggeAbstandHorizontal, bildVornGroessenverhaeltnisHorizontal, flaggeWichtungHorizontal, bildVornFlaggeAbstandVertikal, bildVornGroessenverhaeltnisVertikal, flaggeWichtungVertikal] : abstandFlaggeNw = (bildVornFlaggeAbstandHorizontal * bildVornGroessenverhaeltnisHorizontal * flaggeWichtungHorizontal) + (bildVornFlaggeAbstandVertikal * bildVornGroessenverhaeltnisVertikal * flaggeWichtungVertikal) }
25	{ [abstandFlaggeNw, posX, flaggeNwX, posY, flaggeNwY] : abstandFlaggeNw * 0.05 >= ABS(abstandFlaggeNw - SQRT(POW(posX-flaggeNwX, 2) + POW(posY-flaggeNwY, 2))) }
26	{ [bildRechtsFlaggeMitteHorizontal, bildRechtsFlaggeLinks, bildRechtsFlaggeRechts, bildRechtsBreite] : bildRechtsFlaggeMitteHorizontal = (bildRechtsFlaggeLinks + bildRechtsFlaggeRechts) / 2.0 - ((bildRechtsBreite - 1) / 2.0) }
27	{ [bildRechtsFlaggeMitteVertikal, bildRechtsFlaggeOben, bildRechtsFlaggeUnten, bildRechtsHoehe] : bildRechtsFlaggeMitteVertikal = ((bildRechtsHoehe - 1) / 2.0) - ((bildRechtsFlaggeOben + bildRechtsFlaggeUnten) / 2.0) }
28	{ [bildRechtsGroessenverhaeltnisHorizontal, flaggeBreite, bildRechtsFlaggeRechts, bildRechtsFlaggeLinks] : bildRechtsGroessenverhaeltnisHorizontal = flaggeBreite / (bildRechtsFlaggeRechts - bildRechtsFlaggeLinks + 1) }
29	{ [bildRechtsGroessenverhaeltnisVertikal, flaggeHoehe, bildRechtsFlaggeUnten, bildRechtsFlaggeOben] : bildRechtsGroessenverhaeltnisVertikal = flaggeHoehe / (bildRechtsFlaggeUnten - bildRechtsFlaggeOben + 1) }
30	{ [bildRechtsAbstandHorizontal, bildRechtsBreite, kameraWinkelHorizontal] :

	bildRechtsAbstandHorizontal = bildRechtsBreite / 2.0 / TAN(kameraWinkelHorizontal/2.0) }
31	{ [bildRechtsAbstandVertikal, bildRechtsHoehe, kameraWinkelVertikal] : bildRechtsAbstandVertikal = bildRechtsHoehe / 2.0 / TAN(kameraWinkelVertikal/2.0) }
32	{ [bildRechtsFlaggeWinkelHorizontal, bildRechtsFlaggeMitteHorizontal, bildRechtsAbstandHorizontal] : bildRechtsFlaggeWinkelHorizontal = ARCTAN((bildRechtsFlaggeMitteHorizontal) / bildRechtsAbstandHorizontal) }
33	{ [bildRechtsFlaggeWinkelVertikal, bildRechtsFlaggeMitteVertikal, bildRechtsAbstandVertikal] : bildRechtsFlaggeWinkelVertikal = ARCTAN((bildRechtsFlaggeMitteVertikal) / bildRechtsAbstandVertikal) }
34	{ [bildRechtsFlaggeAbstandHorizontal, bildRechtsAbstandHorizontal, bildRechtsFlaggeMitteHorizontal] : bildRechtsFlaggeAbstandHorizontal = SQRT (POW(bildRechtsAbstandHorizontal, 2) + POW(bildRechtsFlaggeMitteHorizontal, 2)) }
35	{ [bildRechtsFlaggeAbstandVertikal, bildRechtsAbstandVertikal, bildRechtsFlaggeMitteVertikal] : bildRechtsFlaggeAbstandVertikal = SQRT (POW(bildRechtsAbstandVertikal, 2) + POW(bildRechtsFlaggeMitteVertikal, 2)) }
36	{ [abstandFlaggeNw, bildRechtsFlaggeAbstandHorizontal, bildRechtsGroessenverhaeltnisHorizontal, flaggeWichtungHorizontal, bildRechtsFlaggeAbstandVertikal, bildRechtsGroessenverhaeltnisVertikal, flaggeWichtungVertikal] : abstandFlaggeNw = (bildRechtsFlaggeAbstandHorizontal * bildRechtsGroessenverhaeltnisHorizontal * flaggeWichtungHorizontal) + (bildRechtsFlaggeAbstandVertikal * bildRechtsGroessenverhaeltnisVertikal * flaggeWichtungVertikal) }
37	{ [abstandFlaggeNo, posX, flaggeNoX, posY, flaggeNoY] : abstandFlaggeNo * 0.05 >= ABS(abstandFlaggeNo - SQRT(POW(posX-flaggeNoX, 2) + POW(posY-flaggeNoY, 2))) }
38	{ [theta, flaggeSwX, posX, flaggeSwY, posY, bildLinksFlaggeWinkelHorizontal] : theta * 0.05 >= ABS(theta - (3.0/2.0*PI - ARCTAN(ABS((flaggeSwX-posX) / (flaggeSwY-posY))) - ABS(bildLinksFlaggeWinkelHorizontal - PI/2.0))) }
39	{ [theta, flaggeNwX, posX, flaggeNwY, posY, bildVornFlaggeWinkelHorizontal] : theta * 0.05 >= ABS(theta - (ARCTAN(ABS((flaggeNwX-posX) / (flaggeNwY-posY))) + bildVornFlaggeWinkelHorizontal + PI/2.0)) }
40	{ [theta, flaggeNoX, posX, flaggeNoY, posY, bildRechtsFlaggeWinkelHorizontal] : theta * 0.05 >= ABS(theta - (ARCTAN(ABS((flaggeNoY-posY) / (flaggeNoX-posX))) + bildRechtsFlaggeWinkelHorizontal + PI/2.0)) }

3.2 Speziell

Folgende Werte konnten wir aus den Bildern unter `src/image` ablesen.

```
bildLinksBreite         = 200.0
bildLinksHoehe          = 155.0
bildLinksFlaggeLinks    = 81.0
bildLinksFlaggeRechts   = 88.0
bildLinksFlaggeOben     = 79.0
bildLinksFlaggeUnten    = 107.0

bildVornBreite          = 208.0
bildVornHoehe           = 160.0
bildVornFlaggeLinks     = 158.0
bildVornFlaggeRechts    = 168.0
bildVornFlaggeOben      = 3.0
bildVornFlaggeUnten     = 47.0
```

```
bildRechtsBreite        = 208.0
bildRechtsHoehe         = 160.0
bildRechtsFlaggeLinks   = 126.0
bildRechtsFlaggeRechts  = 134.0
bildRechtsFlaggeOben    = 74.0
bildRechtsFlaggeUnten   = 111.0
```

4 Umsetzung

Das Programm besteht aus zwei Komponenten

1. Konfigurationsdateien, welche die Definition der Variablen, Domains und Constraints enthalten,
2. Java-Klassen, welche diese Konfigurationendateien einlesen, auswerten und das Ergebnis (grafisch) darstellen („Darstellungs- und Interpretationskomponente“).

Damit kann die Definition des Constraint-Netzes weitgehend unabhängig vom Java-Teil erfolgen.

Zur Umsetzung der Interpretationskomponente wird ein Java-Interpreter eingesetzt, welcher die gegebenen Ausdrücke verarbeitet. Dies hat eine 40fach längere Ausführungsdauer zur Folge.

4.1 Systemvoraussetzungen

- Java 1.5 mit Java und Javac Executable im PATH
- Windows

Es sollte auch unter Linux mit X funktionieren.

4.2 Paketübersicht

- `.` Startscripte
- `bin` kompilierte Java-Klassen
- `doc` Dokumentation
- `doc/javadoc` Dokumentation der Java-Klassen
- `lib` benutze Bibliotheken
- `src/config` Konfigurationsdateien
- `src/image` Grafiken
- `src/java` Source der Java-Klassen

4.3 Dokumentation

Unter `doc/javadoc` befindet sich die Java-Dokumentation, unter `doc` dieses Dokument. Jede Konfigurationsdatei ist ebenfalls mit Kommentaren versehen.

4.4 Programmaufruf

`run.bat` starten.

4.5 Kompilieren

`compile.bat` starten.

4.6 Programmablauf

1. Ausdrücke + Constrains einlesen
2. Ausdrücke auswerten und für das spätere Benutzen bekannt machen
3. C0 bestimmen
4. Cn bestimmen --> (x,y) wird eingeschränkt, (theta)-Contraints werden zwischengespeichert
5. Zentroid bestimmen --> festes (x,y), um (theta) einzuschränken
6. Cn bestimmen(2) --> (theta) wird eingeschränkt (mit festem (x,y))
7. Den Durchschnitt aller (theta) bestimmen
8. Finale Pose ausgeben

4.7 Ausgabe

Pro Domaingeneration werden die noch möglichen Positionen (x,y) durch blaue Kreuze dargestellt, die noch möglichen Blickwinkel (theta) als rote Strahlen, welche mit Mittelpunkt ausgehen.
Falls es während der Abarbeitung keine global konsistente Belegung mehr gibt, so wird dies dem Benutzer mitgeteilt. Andernfalls wird eine finale Pose ausgegeben.

4.8 Konfiguration

Beide Konfigurationsdateien werden vom gleichen Interpreter verarbeitet. Der Unterschied ist, dass die Ausdrücke in `expressions.config` nur einmal während der Initialisierung abgearbeitet werden, während die in `constraints.config` für jede mögliche Position/Winkel überprüft werden.

Bei den Ausdrücken muss es sich um valide Java-Ausdrücke handeln. Zur Vereinfachung der Notation von mathematischen Formeln haben wir einen kleinen Ersetzungsmechanismus eingebaut.

Zulässige Abkürzung	**Ersetzung**
`ARCSIN`	`Math.asin`
`ARCCOS`	`Math.acos`
`ARCTAN`	`Math.atan`
`SIN`	`Math.sin`
`COS`	`Math.cos`
`TAN`	`Math.tan`
`ABS`	`Math.abs`
`SQRT`	`Math.sqrt`
`POW`	`Math.pow`
`EXP`	`Math.exp`
`PI`	`Math.PI`

4.8.1 expressions.config

Hier werden neben Parametern für die Darstellungs- und Interpretationskomponente auch alle Variablen mit einelementigen Domains definiert. Dazu werden diesen Variablen ihre berechneten oder abgelesenen Werte zugewiesen.

Beispiel: `abstandAB=25` definiert die Variable `abstandAB` mit Dom(`abstandAB`)={25}

Parameter für die Darstellungs- und Interpretationskomponente

Name	**Kommentar**	**Standardwert**
`debug`	Nun rate mal :o)	`false`
`strichLaenge`	Länge des Striches in Pixeln, welcher zur Darstellung	`300`

	eines Blickwinkel gezeichnet wird.	
`spielfeldRasterX`	Rasterbreite in mm, welches für die Erstellung der diskreten Domain für die x Koordinate des Roboters benutzt wird.	`10.0`
`spielfeldRasterY`	Rasterbreite in mm, welches für die Erstellung der diskreten Domain für die y Koordinate des Roboters benutzt wird.	`10.0`
`thetaRaster`	Rasterbreite in rad, welches für die Erstellung der diskreten Domain für Blickwinkel des Roboters benutzt wird.	`0.01`

4.8.2 constraints.config

Hier werden alle Constraints definiert, welche Variablen mit mehrelementigen Domains betreffen. Folgende Variablen sind implizit definiert

- `posX` X-Koordinate im Spielfeld
- `posY` Y-Koordinate im Spielfeld
- `theta` Richtungswinkel
- Alle Variablen, welche auf der linken Seite einer Zuweisung in der Ausdrücke-Config stehen.

Die Domain für `posX` und `posY` ist implizit durch die Spielfeldabmaße begrenzt und kann nur auf Sourcecode angepasst werden (Klasse Spielfeld).

Die Domain für `theta` ist implizit durch 0 und 2π begrenzt und kann nur im Sourcecode angepasst werden (Klasse Spielfeld).

Im Falle von Constraints, welche sich nur auf die Position beziehen, sind `posX` und `posY` abhängige Variablen. Bei allen Blickwinkel(theta)-Constraints sind `posX`, `posY` unabhängig und `theta` ist die abhängige Variable (siehe 2.3).