# Locatie bepalen

Wanneer we de locatie gaan bepalen krijgen we een set van symbolen binnen. Elk van deze symbolen heeft een kleur en vorm, en de x- en y-coördinaat binnen de afbeelding. Eerst gaat het algoritme bepalen welk van de gegeven symbolen het meest centraal ligt ten opzichte van de andere symbolen. Dit symbool wordt het middelpunt.

Vervolgens gaan we driehoeken zoeken: twee punten rond het middelpunt, die vlak naast elkaar liggen en die samen met het middelpunt een gelijkzijdige driehoek binnen het raster vormen. Dit is omdat gegeven is dat elke set van 3 symbolen uniek is binnen het speelveld. Om dit te bereiken kijken we welke punt er het dichtst bij het middelpunt ligt, en filteren we de andere punten zodat we enkel punten overhouden waarvoor de afstand tot het middelpunt binnen een bepaalde marge (1,2) van de kortste afstand ligt. Deze punten worden gesorteerd op poolcoördinaten. Vervolgens proberen we te matchen voor de driehoek gevormd door dit middelpunt en twee punten die naast elkaar staan in de gesorteerde lijst. Deze twee punten mogen wel maximum 1,2x de kortste afstand uit elkaar liggen (voor het geval er symbolen ontbreken) en het tweede punt moet in wijzerzin van het eerste punt liggen.

In: symbols; out: x, y, alpha

Symbol center = calculateCenter(symbols)

Symbol closestToMid = closestSymbol(symbols,center) %get symbol closest to center symbol

closestToMidDist = dist(center,closestToMid)

List<Symbol> neighbours = filter(symbols,1.2\*closestToMidDist)

neighbours = sortPolar(neighbours,center) %sort around center

for all(Symbol s1 in neighbours)

s2 = next(s1) %symbol following s1

if(dist(s1,s2) <= 1.2\*closestToMidDist && clockwise(s1,s2))

possibleLocation = match(center,s1,s2)

Dan kan het echte vergelijken met het raster gebeuren: alle punten van het veld worden afgegaan. Indien het middelpunt overeen komt met een punt uit het raster, worden alle buren van dit punt opgevraagd. Indien de andere twee punten overeen komen met twee punten van deze buren, die ook naast elkaar liggen en in dezelfde volgorde staan, hebben we een match en hebben we de locatie gevonden. De locatie die wordt gegeven is de locatie van het middelpunt.

Match: In: center, s1, s2; out: x, y, alpha

for all(Symbol s on map)

If(s matches center)

for all(Symbol sm1 in map.neighbours(center) ) %sorted neighbours

sm2 = next(sm1)

if(s1 matches sm1 and s2 matches sm2)

location found

Vervolgens moet de hoek bepaald worden. Hiervoor gaan we kijken welke van de drie figuren horizontaal liggen op het raster. Zo zijn er telkens twee te vinden voor elke driehoek. Dan kijken we met welke twee punten van de gevonden punten deze twee overeenkomen. Nu kunnen we door gebruik te maken van het verschil in x- en y-coördinaat in de afbeelding en van de tangens, berekenen wat de hoek is ten opzichte van de werkelijke hoek die 0 zou moeten zijn.

Hiervoor wordt een onderscheid gemaakt op basis van de groottes van de coördinaten (x1 < x2?, y1 < y2?), om aan de hand daarvan een bepaalde transformatie te doen op de hoek gevonden van de tangens (omdat de positiecontroller de hoek in een bepaald formaat verwacht en om alle hoeken tussen –pi en pi te vinden).

# Positiecontrole

(al geschreven)

# Software

Image processing => locatie bepalen => positiecontrole