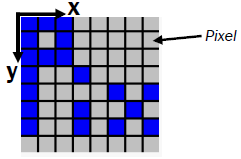
## Kamera til CrustCrawler

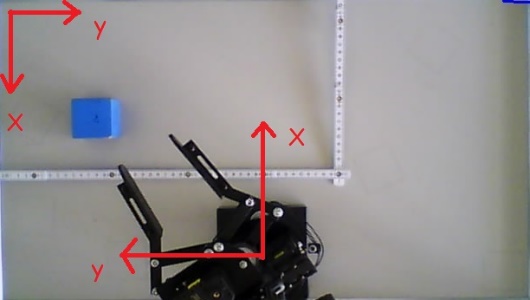
Ved hjælp af det påmonterede IP kamera og den udviklede vision algoritme, er det muligt at detektere hvilke objekter der er synlige på bordpladen. Når vision algoritmen har detekteret en klods, samt dens farve, returneres center punktet for objektet i form af pixelkoordinater. Da billedets koordinatsystem hverken i enhed eller orientering, stemmer overens med robottens koordinatsystem, er det nødvendigt at lave en transformation mellem de to ”frames”.

Da objektets position leveres i pixelkoordinater, omregnes pixel til centimeter. Ved hjælp af formlen

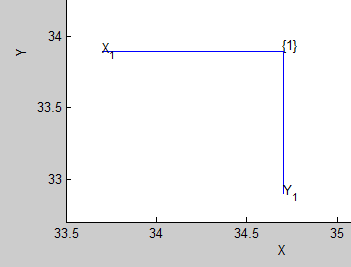
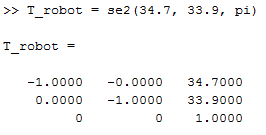
   
Formlen giver en god tilnærmelse for objektets resolution i cm. Ved at måle bordets fysiske længde og højde inden for aluminiumskanten, samt de tilsvarende længder i pixel på billedet, gav det følgende resultater:

Længde opløsning: 

Højde opløsning: 

For kun at detektere klodser på bordpladen, beskæres billedet til kun at dække bordpladen. Dette har også den fordel at øverste venstre hjørne på billedet svarer til øverste venstre hjørne på bordpladen set fra kameraets position. Da kameraets ”frame” er defineret som øverste venstre hjørne er det muligt at måle robottens fysiske position ud fra hjørnet. Robottens position måles til samt en rotation på 180o i forhold til kameraets ”frame”.

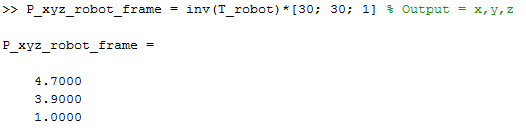
Ved hjælp af Matlab funktionen ”se2” i Robotic toolbox udviklet af Peter Corke, er det muligt at få returneret den homogene transformation som repræsentere translation (34.7, 33,9) samt rotationen på 180o.



Som det ses på plottet stemmer transformationen overens med de to frames position på billedet.

For at finde en position opgivet ud fra kameraets frame, er det nødvendigt at tage den inverse af ovenstående transformation for at finde klodsens position ud fra robotarmen. Det vil sige at punktet set fra robottens frame er:



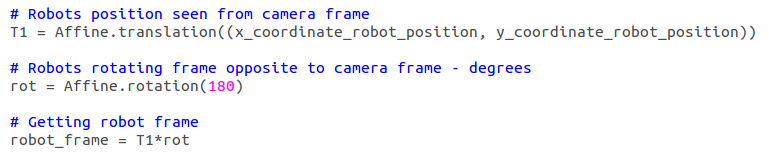


Da robottens position er ses det tydeligt at det beregnede punktet er korrekt.

# Matlab til python:

For at kunne bruge udregningerne lavet i Matlab i det samlede projekt skulle vi have konverteret disse rotations transformationer til python kode. Da det ikke var muligt at mappe direkte fra matlab til python pga. den brugte ”se2” funktion fra robotics toolbox, fandt vi et 2D geometri bibliotek til python ved navn ”Planar”. Vha. dette bibliotek, kunne vi lave vores lineær transformation i 2D og derefter rotere vores transformerede frame så det passer til QRBotten. Python koden herunder viser hvordan vi benytter Planar bibliotekets translation funktion til at flytte koordinat systemet så det passer til robottens placering, og rotation funktionen som roterer dette koordinat system så vender rigtigt ift. kamera framet. Variablerne x\_coordinate\_robot\_position og y\_coordinate\_robot\_position er robottens verdens koordinater set fra kamera framet.





Måden hvorpå vi så udregner et objekts position ud fra robotten frame, er ved først at finde de aktuelle distancer set fra kamera framet. Dette gøres ved at multiplicere længde og højde opløsningen med de x og y koordinater, angivet i pixels, som modtages fra vision delen. Den aktuelle højde og længde udgør derfor et objekts position set fra kamera framet. Der gøres her brug af en funktion fra Planar biblioteket ”Vec2” laver vores aktuelle distancer om til en vector. For så at finde et objekts position set fra robotten frame er ved at tage den inverse robot frame og multiplicerer med vores vector som derved retuneres.

// Insert picture