Camera calibratie voor de kanteltafel

Camera Calibratie voor de Kanteltafel

Jan Peter Meeuwse

WFW-rapportnr 94.042

Stageverslag van : Jan Peter Meeuwse
Stageinstelling : Technische Universiteit Eindhoven
Faculteit Werktuigbouwkunde
Vakgroep Fundamentele Werktuigbouwkunde
Sectie Regeltechniek
Periode : December 1993 t/m Maart 1994
Begeleiders : Ing. Niels Olthuis
Ir. Jos Banens

Eindhoven, Maart 1994

Samenvatting

Als resultaat van een stageopdracht, uitgevoerd bij de sectie Regeltechniek van de vakgroep Fundamentele Werktuigbouwkunde, ligt hier een stageverslag. De opdracht van deze stage is het ontwerpen van een calibratieprocedure voor het camerameetsysteem van de kanteltafel. Na een literatuuronderzoek over de calibratie van camerameetsystemen is een begin gemaakt met de praktische uitvoer hiervan in MATLAB en C++.
In deze stage is regelmatig een beroep gedaan op de kennis van de vakgroep Meten en Regelen van de faculteit Elektrotechniek. De ondersteuning bestond onder andere uit het beschikbaar stellen van bestaande calibratieprogrammatuur. De faculteit Elektrotechniek is reeds geruime tijd bezig met het calibreren van verwante camerameetsystemen.

De kanteltafel is een systeem dat bestaat uit een vierkante plaat die te roteren is om zijn twee middellijnen. Deze vrijheidsgraden kunnen worden gestuurd door twee elektromotoren. Er wordt naar gestreefd om een kogeltje op deze plaat, elke gewenst traject te kunnen laten volgen. Om deze kanteltafel succesvol te kunnen regelen is een nauwkeurig meetsysteem voor de bepaling van de positie van de kogel noodzakelijk. Er is gekozen om de positie te bepalen uit een combinatie van de balpositie in het camerabeeld en de stand van de plaat. De beelden worden geleverd door een eenvoudige CCD-bewakingscamera en de stand van de plaat is te bepalen uit de encoderstanden van beide elektromotoren.
Om nauwkeurige metingen te kunnen doen met een camera dient men een groot aantal parameters van de camera te kennen. Deze parameters zijn nodig om een 2D camerabeeld terug te kunnen rekenen naar de 3D werkelijkheid. Hiervoor wordt gebruik gemaakt van een theoretisch cameramodel. De parameters zijn te verdelen in twee groepen, de intrinsieke en extrinsieke parameters. De intrinsieke parameters zijn parameters die uniek en constant zijn voor een specifiek camerasysteem. Met een specifiek camerasysteem wordt bedoeld de combinatie van camera en lens. Een voorbeeld van een intrinsieke parameter is de brandpuntsafstand. Extrinsieke parameters zijn parameters die niet vast zijn voor een specifiek camerasysteem. De extrinsieke parameters worden beinvloed door de gebruiker en zijn omgeving en dienen dan ook vaak opnieuw bepaald te worden. Een voorbeeld van extrinsieke parameters zijn de parameters die nodig zijn voor de beschrijving van de positie en oriëntatie van de camera ten opzichte van een door de gebruiker gedefinïeerd coördinatenstelsel. Dit coördinatenstelsel ligt vast ten opzichte van de kanteltafel en wordt het wereldcoördinatenstelsel genoemd. In deze stage is hoofdzakelijk gekeken naar deze rotatie en translatie parameters. Voor de bepaling van de extrinsieke parameters dienen

de intrinsieke parameters bekend te zijn, en deze zullen in deze stage eveneens bepaald worden.

Voor de bepaling van de cameraparameters is gebruik gemaakt van een aantal vaste meetpunten waarvan de posities in het wereldcoördinatenstelsel nauwkeurig bekend zijn. De CCD-camera maakt een beeld met hierin de meetpunten, dit beeld wordt vervolgens in een framegrabber gedigitaliseerd. In het gedigitaliseerde beeld kunnen met behulp van de computer de posities van de meetpunten in het beeldvlak worden bepaald. Met deze twee gegevens, te weten de posities in wereldcoördinaten en de projecties op het beeldvlak, is het mogelijk de intrinsieke en extrinsieke cameraparameters nauwkeurig te berekenen.

Inhoudsopgave

Inleiding

De belangstelling voor camerameetsystemen is de afgelopen jaren sterk toegenomen. In de zeventiger jaren werden verschillende onderzoeken op het gebied van beeldanalyse voor industriële toepassing gestart. De snel dalende kosten van computerapparatuur hebben deze ontwikkeling zeker gestimuleerd. Pas begin tachtiger jaren werd het economisch haalbaar om camerasystemen op industriële schaal toe te passen. Op tal van plaatsen wordt de beeldanalyse ingezet voor het automatiseren van processen. Zo zijn er toepassingen op het gebied van lasrobots voor het lassen van grote werkstukken. De camera wordt gebruikt als meetsysteem aan de hand waarvan bepaalde processen geregeld kunnen worden.

Binnen de sectie Regeltechniek van de faculteit Werktuigbouwkunde wil men gebruik gaan maken van een camerameetsysteem voor de kanteltafel. De kanteltafel zal in de nabije toekomst het werkterrein worden voor stagiaires, afstudeerders en promovendi. Zij zullen regelconcepten beproeven op de kanteltafel. Met name het gebied van fuzzy control zal de nodige aandacht krijgen.

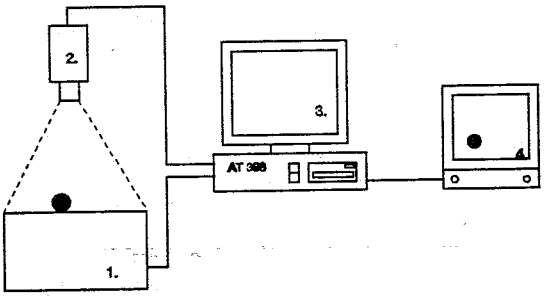
In deze stageopdracht is gewerkt aan een calibratieprocedure van het camerameetsysteem van de kanteltafel. Er is in de literatuur gezocht naar een eenvoudige en betrouwbare calibratieprocedure. Deze is vervolgens geprogrammeerd in MATLAB en C++.
De opbouw van dit verslag is als volgt. Na deze korte inleiding volgt een hoofdstuk waarin de opstelling van de kanteltafel behandeld wordt. Hoofdstuk twee bespreekt een theoretisch cameramodel. Dit theoretisch cameramodel dient als basis voor de calibratieprocedure, die uitvoerig beschreven wordt in hoofdstuk drie. Hoofdstuk vier bevat de resultaten die met de calibratieprocedure in deze stage bereikt zijn. In het laatste hoofdstuk worden enige conclusies en aanbevelingen gedaan met betrekking tot het camerameetsysteem van de kanteltafel. Aan het eind van dit verslag bevindt zich een literatuurverwijzing en de appendix met de voor de calibratie ontwikkelde computerprogramma's. Tevens bevindt zich hierbij een korte softwarebeschrijving.

Hoofdstuk 1

De opstelling

In dit hoofdstuk wordt de kanteltafel met het camerameetsysteem nader bekeken. In eerste instantie wordt een overzicht gegeven van de totale opstelling, vervolgens wordt de beeldverwerking met de CCD-camera en framegrabber toegelicht. Tenslotte wordt de werking van het totale meetsysteem beschreven.

1.1 De kanteltafel en het meetsysteem



Figuur 1.1: De opstelling van de kanteltafel

De opstelling zoals deze in het mechanicalaboratorium van de vakgroep Fundamentele Werktuigbouwkunde staat, bestaat uit de volgende deelsystemen:

1. De kanteltafel met de kogel.
2. De Philips CCD-bewakingscamera.
3. De computer met daarin de framegrabber. De framegrabber is een insteekkaart in de PC, waarmee CCD-camerabeelden gedigitaliseerd worden.

4. De monitor voor de uitvoer van de framegrabber. Deze is in principe niet noodzakelijk maar biedt de gebruiker de mogelijkheid de instellingen van de camera te controleren. Met behulp van de monitor kunnen de lichtcondities voor de camera beoordeeld worden en is te zien of de gehele kanteltafel in beeld is.

In figuur (1.1) zijn twee verbindingen te zien tussen de PC en de opstelling. Het transport van de camerabeelden verloopt via de verbinding tussen de CCD-camera en de PC. De verbinding tussen de kanteltafel en de PC transporteert de gegevens over de stand van de tafel in de vorm van de encoderstanden van de elektromotoren.

1.2 De CCD-camera en framegrabber

Omdat in het meetsysteem de beeldverwerking van groot belang is volgt hier een toelichting op de werking van de CCD-camera met de framegrabber.
De hoofdonderdelen in een CCD-camera zijn de lens en de lichtgevoelige sensor. Zoals dat bij elke camera het geval is, worden beelden geprojecteerd op een beeldvlak achter de lens. Beelden die worden opgenomen met een CCD-camera worden geprojecteerd op de lichtgevoelige sensor die geplaatst is in het beeldvlak. De CCD-camera bevat een zogenaamde Solid-State-Sensor. Deze sensoren kenmerken zich door kleine afmetingen, eenvoudige toepassing en een goedkoop produktieproces. De beeldkwaliteit is voldoende voor amateurcamera's en voor de meeste camera's met industriële toepassing.
De werking van de CCD-sensor berust op het principe van ladingsdragergeneratie in een halfgeleider materiaal onder invloed van licht. Anders gezegd, licht is in staat om elektronen vrij te maken in halfgeleidermateriaal waardoor het ladingspakket in de halfgeleider toeneemt. Het aantal gegenereerde elektronen in een bepaald tijdbestek is evenredig met de hoeveelheid binnentredend licht. Zodoende is de grootte van het ladingspakket een maat voor de belichting. De CCD-sensor bestaat in principe uit een groot aantal van deze lichtgevoelige cellen die gerangschikt zijn in een matrix. Iedere cel vertegenwoordigt een pixel in het totale beeld.
De sensorafmetingen van de in deze stage gebruikte CCD-camera bedragen 12.7x9.5 [mm], dit komt overeen met 782x582 cellen (pixels). De uitlezing van het ladingsbeeld van de CCD-sensor vindt seriëel en rij voor rij plaats. Er wordt een analoog signaal verkregen waarin alle pixelinformatie in de tijd achter elkaar geplaatst is. Dit analoge signaal gaat naar een framegrabber waarin analoog-digitaal omzetting plaatsvindt. De framegrabber is een insteekkaart in de PC met behulp waarvan de CCD beelden gedigitaliseerd worden. In de framegrabber wordt het analoge signaal per beeldlijn bemonsterd met 512 samples. De framegrabber bemonstert alleen de eerste 512 lijnen van het CCD-beeld zodat er uiteindelijk een gedigitaliseerd beeld van 512x512 pixels ontstaat met in elk punt een grijswaarde tussen de 0 en 255. Dit beeld is in de zogenaamde framegrabber beschikbaar voor uitlezing (en eventuele bewerking) door de processor in de PC.
Een beeld (frame) wordt opgebouwd uit twee rasters (fields) die elk maar de helft van het aantal beeldlijnen bevatten. De lijnen van het ene raster vallen tussen de lijnen van het andere raster. Dit wordt interliniëring genoemd. Het ene raster bevat nu de lijnen met

een even nummer en het andere raster bevat nu de lijnen met een oneven nummer. Het beeld wordt afwisselend ververst met oneven en even lijnen. Terwijl het ene beeld wordt ververst, is het andere beeld stabiel en hierin kunnen dan bewerkingen uitgevoerd worden. De beeldfrequentie is 25 Hz maar door de interliniëring komt nieuwe informatie met een frequentie van 50 Hz beschikbaar. De opbouw en timing van de camera voldoet aan de CCIR-B norm voor videosignalen.

1.3 De werking van het meetsysteem

Na deze uitleg over de beeldverwerking volgt nu een beschrijving van de wijze waarop de positie van de kogel op het tafelblad wordt bepaald.
Een even of oneven raster wordt van de camera naar de framegrabber gestuurd. Terwijl het ene raster in de framegrabber gedigitaliseerd wordt, is het andere raster stabiel. In het stabiele raster kunnen bewerkingen worden uitgevoerd. Een computerprogramma evalueert de grijswaarde van elk pixel in het stabiele raster en bepaalt door middel van een zwaartepuntsmeting de positie van de kogel in het beeldvlak. Tegelijkertijd worden de encoderstanden van de elektromotoren uitgelezen. Door de encoderstanden zijn de rotatiehoeken van de tafel bekend. De positie van de kogel op de tafel wordt nu berekend uit de combinatie van de positie van de kogel in het beeld en de rotatiehoek van de tafel.
De berekening van de positie van de kogel in het beeld is het onderwerp van deze stage en staat uitvoerig beschreven in de rest van dit verslag.

Hoofdstuk 2

Het cameramodel

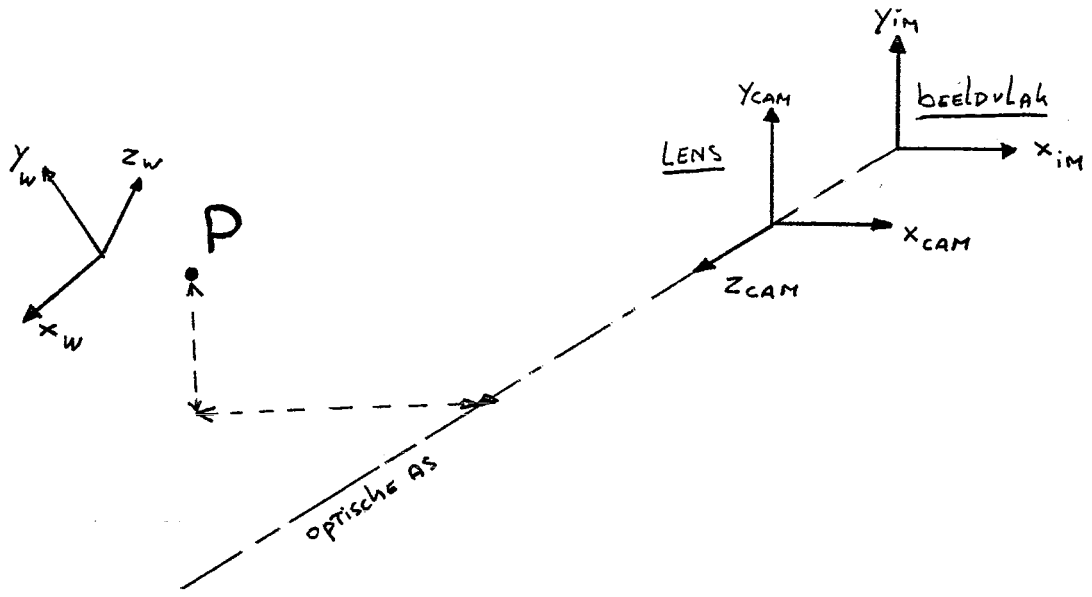
In dit hoofdstuk wordt een mathematisch cameramodel besproken waarmee de CCD-camera voldoende nauwkeurig beschreven kan worden. De parameters in een cameramodel worden onderscheiden in twee groepen, hierover gaat de eerste paragraaf. Daarna wordt een verstoringsvrij cameramodel voorgesteld. Vervolgens komt een uitgebreider model aan de orde waarin rekening wordt gehouden met een drietal lensverstoringen. Het uitgebreide model dient als basis voor de calibratie van het camerameetsysteem van de kanteltafel. Dit hoofdstuk is voornamlijk gebaseerd op [Weng, Cohen, Herniou].

2.1 Intrinsieke en extrinsieke parameters

Een mathematisch cameramodel geeft de relatie tussen de wereldcoördinaten en de beeldcoordinaten van een punt in de ruimte. In een cameramodel zitten een groot aantal parameters. Deze parameters worden onderverdeeld in twee groepen, namelijk de intrinsieke en extrinsieke parameters. De intrinsieke parameters zijn parameters die constant zijn voor een specifiek camerasysteem. In dit verslag zijn dit de lensverstoringsparameters G_1 tot en met G_5 en de brandpuntsafstand f. Deze staan in dit verslag in de kolom $d = (f\ G_1\ G_2\ G_3\ G_4\ G_5)^T$.
De extrinsieke parameters zijn de parameters die niet vast liggen voor een specifiek camerasysteem. Dit zijn de positie en oriëntatie van de camera ten opzicht van een door de gebruiker gedefiniëerd coördinatensysteem. Deze liggen vast met drie rotatiehoeken ϕ_1 tot en met ϕ_3 en drie translatieparameters t_x tot en met t_z. In dit verslag worden staan ze in een kolom $m = (\phi_1\ \phi_2\ \phi_3\ t_x\ t_y\ t_z)^T$.

2.2 Een verstoringsvrij cameramodel

Een punt P in de ruimte, kan beschreven worden ten opzichte van een door de gebruiker gedefiniëerd coördinatenstelsel met de zogenaamde wereldcoördinaten $\underline{x}_w = (x_w, y_w, z_w)^T$, maar ook ten opzichte van een cameracoördinatensysteem met $\underline{x}_{cam} = (x_{cam}, y_{cam}, z_{cam})^T$,



Figuur 2.1: Wereldcoördinaten en cameracoördinaten

zie Figuur (2.1). Het wereldcoördinatensysteem is een door de gebruiker vrij te kiezen, vast coördinatensysteem. Een cameracoördinatensysteem is gedefinïeerd met de oorsprong in het hart van de lens en de z-as parallel aan de optische as van de camera. Het beeldvlak is het vlak achter de lens, loodrecht op de optische as. Op het beeldvlak vindt de projectie van een punt in de ruimte plaats. In het beeldvlak wordt gebruik gemaakt van een 2D beeldcoördinatenstelsel $\underline{x}_{im} = (x_{im}, y_{im})^T$.
De positie van een punt P in wereldcoördinaten dient omgerekend te worden naar het cameracoördinatenstelsel. Dit vindt plaats door het wereldcoördinatenstelsel te roteren en transleren naar het cameracoördinatenstelsel. Voor deze transformatie wordt gebruik gemaakt van een (3x3) rotatiematrix R en een (3x1) translatievector T, zodat:

$$\underline{x}_{cam} = R\underline{x}_w + T \tag{2.1}$$

Hierin is:

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad \text{en} \quad T = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \tag{2.2}$$

De rotatiematrix R wordt bepaald door drie Eulerhoeken (ϕ_1, ϕ_2, ϕ_3). De rotatiematrix R kan met behulp van de afzonderlijke bijdrage van elke Eulerhoek als volgt berekend worden:

$$R = \begin{pmatrix} \cos\phi_1 & 0 & \sin\phi_1 \\ 0 & 1 & 0 \\ -\sin\phi_1 & 0 & \cos\phi_1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_2 & -\sin\phi_2 \\ 0 & \sin\phi_2 & \cos\phi_2 \end{pmatrix} \begin{pmatrix} \cos\phi_3 & \sin\phi_3 & 0 \\ -\sin\phi_3 & \cos\phi_3 & 0 \\ 0 & 0 & 1 \end{pmatrix} =$$

$$\begin{pmatrix} \cos\phi_3\cos\phi_2 + \sin\phi_3\sin\phi_2\sin\phi_1 & \sin\phi_3\cos\phi_1 & \cos\phi_3\sin\phi_2 + \sin\phi_3\sin\phi_2\cos\phi_1 \\ -\sin\phi_3\cos\phi_2 + \cos\phi_3\sin\phi_2\sin\phi_1 & \cos\phi_3\cos\phi_2 & -\sin\phi_3\sin\phi_2 - \cos\phi_3\sin\phi_1\cos\phi_2 \\ -\cos\phi_1\sin\phi_2 & \sin\phi_1 & \sin\phi_1\sin\phi_2 \end{pmatrix}$$

Voor de eenvoud zullen de afzonderlijke elementen van de rotatiematrix R verder in dit verslag met r_{ij} aangeduid worden. Nu de omrekening van wereldcoördinaten, naar cameracoördinaten heeft plaatsgevonden kan de projectie op het beeldvlak bepaald worden. De berekening van beeldcoördinaten (x_{im}, y_{im}) vindt plaats met gebruikmaking van de brandpuntsafstand f:

$$x_{im} = f\frac{x_{cam}}{z_{cam}} \tag{2.3}$$

$$y_{im} = f\frac{y_{cam}}{z_{cam}} \tag{2.4}$$

Na subsititutie van (2.2) in (2.3) en (2.4) onstaan de volgende vergelijkingen waarin de wereldcoördinaten worden omgerekend naar beeldcoördinaten:

$$x_{im} = f\frac{r_{11}x_w + r_{12}y_w + r_{13}z_w + t_x}{r_{31}x_w + r_{32}y_w + r_{33}z_w + t_z} \tag{2.5}$$

$$y_{im} = f\frac{r_{21}x_w + r_{22}y_w + r_{23}z_w + t_y}{r_{31}x_w + r_{32}y_w + r_{33}z_w + t_z} \tag{2.6}$$

De beeldcoördinaten $\underline{x}_{im}$ staan weergegeven in metrische eenheden, in de framegrabber wordt de positie weergegeven in pixels. Voor de omrekening van metrische eenheden naar pixels wordt gebruik gemaakt van een horizontale en verticale schaalfactor S_x en S_y. De schaalfactor is een maat voor het aantal metrische eenheden per pixel, bijvoorbeeld [mm/pix]. De berekening van de schaalfactoren op dezelfde wijze uitgevoerd als in [Tsai & Lenz]. In het algemeen wordt in het midden van het framegrabberbeeld de oorsprong gedefinïeerd. Dit wordt gedaan omdat aangenomen wordt dat dit ongeveer het snijpunt is van de optische as met het beeldvlak. In dit geval zijn de afmetingen van het framegrabberbeeld 512x512 pixels, en dus wordt de oorsprong gesitueerd op de 256^e rij en 256^e kolom. De verschuiving van de oorsprong wordt tot uitdrukking gebracht in de verschuivingsfactoren C_x en C_y, in pixels. De transformatie van metrische beeldcoördinaten $\underline{x}_{im}$ naar framegrabbbercoördinaten $\underline{x}_f$ wordt nu:

$$x_f = \frac{x_{im}}{S_x} + C_x \tag{2.7}$$

$$y_f = \frac{y_{im}}{S_y} + C_y \tag{2.8}$$

Omgekeerd wordt de transformatie van framegrabbercoördinaten naar metrische beeldcoordinaten:

$$x_{im} = S_x(x_f - C_x) \tag{2.9}$$

$$y_{im} = S_y(y_f - C_y) \tag{2.10}$$

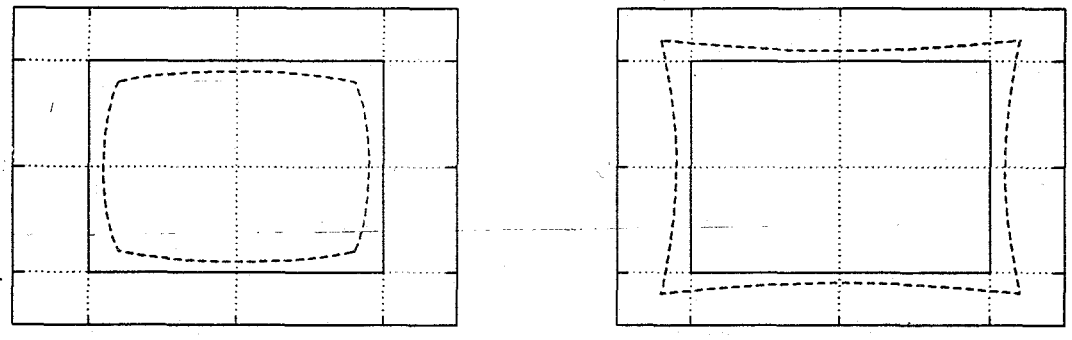
2.3 Het cameramodel met lensverstoring

Het cameramodel zal nu uitgebreid worden met lensverstoringen. Hier zullen drie soorten verstoringen worden bekeken: de radiale verstoring, decentrering verstoring en prisma verstoring.

Radiale verstoring: De radiale verstoring veroorzaakt een verschuiving van de beeldpunten, naar binnen of buiten ten opzichte van de optische as. Deze lensfout wordt veroorzaakt door imperfecties in de lens. De radiale lensfout heeft twee verschijningsvormen namelijk de ton-vervorming en de kussen-vervorming, zie Figuur (2.3). Een benadering van de radiale lensfout δx_r en δy_r is afhankelijk van de lensvervormingsconstante κ_1 en wordt gegeven door:

$$\delta x_r = \kappa_1 x_{im}(x_{im}^2 + y_{im}^2) + O[(x_{im}, y_{im})^5] \tag{2.11}$$

$$\delta y_r = \kappa_1 y_{im}(x_{im}^2 + y_{im}^2) + O[(x_{im}, y_{im})^5] \tag{2.12}$$



Figuur 2.2: De lensvervormingen ten gevolge van radiale verstoring

Decentrering verstoring: De decentrering verstoring treedt op wanneer meerdere lenzen achter elkaar geplaatst worden en de optische assen niet exact samenvallen. Deze verstoring heeft zowel een radiale als een tangentiële verplaatsing van de beeldpunten tot gevolg. Figuur(2.3) geeft een indicatie over de vorm van de verstoring. Een benadering voor de decentrering lensfout δx_d en δy_d wordt gegeven door:

$$\delta x_d = p_1(3x_{im}^2 + y_{im}^2) + 2p_2 x_{im} y_{im} + O[(x_{im}, y_{im})^4] \tag{2.13}$$

$$\delta y_d = 2p_1 x_{im} y_{im} + p_2(x_{im}^2 + 3y_{im}^2) + O[(x_{im}, y_{im})^4] \tag{2.14}$$

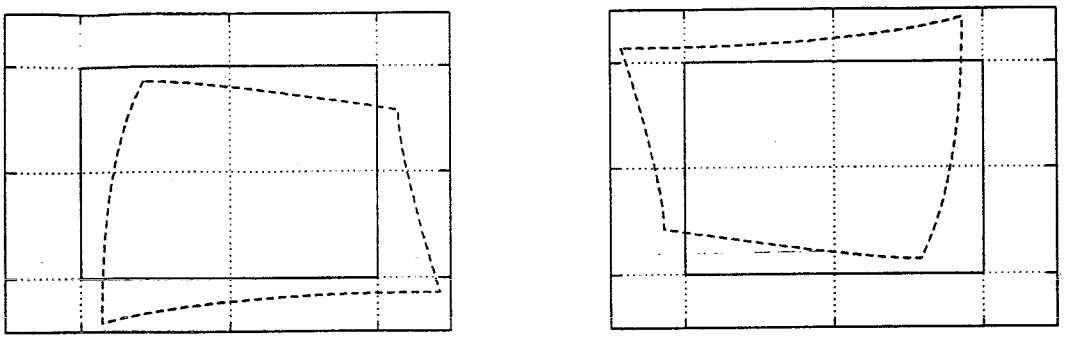
met hierin:

$$p1 = -j_1 \sin\theta_0$$

$$p2 = j_1 \cos\theta_0$$

Hierin zijn θ_0 is de hoek tussen de positieve x-as en de as met de grootste tangentïele verplaatsing en j_1 is de bijbehorende schaalfactor.

Prisma verstoring: De prismaverstoring wordt veroorzaakt door imperfecties in de lens en de onnauwkeurige montage van de lens in de camera. Deze verstoring heeft eveneens radiale en tangentiële verplaatsing van de beeldpunten tot gevolg. Figuur (2.3)



Figuur 2.3: De lensvervorming ten gevolge van decentrering verstoring

geeft een indicatie over de vorm van de verstoring. Een benadering voor de prisma verstoring δx_p en δy_p wordt gegeven door:

$$\delta x_p = s_1(x_{im}^2 + y_{im}^2) + O[(x_{im}, y_{im})^4] \tag{2.15}$$

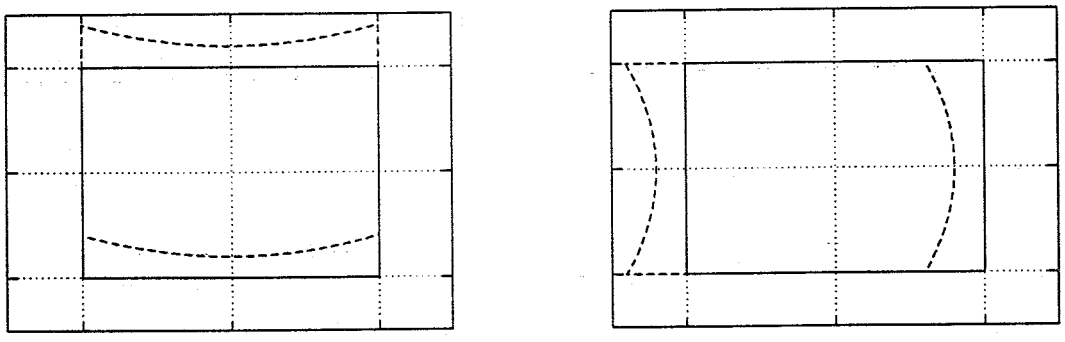
$$\delta y_p = s_2(x_{im}^2 + y_{im}^2) + O[(x_{im}, y_{im})^4] \tag{2.16}$$

met hierin:

$$s1 = -i_1 \sin\theta_1$$

$$s2 = i_1 \cos\theta_1$$

Hierin zijn θ_1 is de hoek tussen de positieve x-as en de as met de grootste tangentiële verplaatsing en i_1 is de bijbehorende schaalfactor.



Figuur 2.4: De lensvervorming ten gevolge van prisma verstoring

In deze stage wordt gebruik gemaakt van een lensfoutmodel waarin de drie bovengenoemde verstoringen worden verdisconteerd. De afzonderlijke lensfoutvergelijkingen worden hiertoe samengevoegd en zo wordt een niet-lineair model verkregen met vijf parameters. Deze parameters worden gecalibreerd met de in het volgende hoofdstuk beschreven methode. Het totale niet-lineaire lensfoutmodel met vijf parameters is nu als volgt op te schrijven:

$$\delta x_d = (G_1 + G_3)x_{im}^2 + G_4 x_{im} y_{im} + G_1 y_{im}^2 + G_5 x_{im}(x_{im}^2 + y_{im}^2) \tag{2.17}$$

$$\delta y_d = G_2 x_{im}^2 + G_3 x_{im} y_{im} + (G_2 + G_4)y_{im}^2 + G_5 y_{im}(x_{im}^2 + y_{im}^2) \tag{2.18}$$

Met hierin de volgende 5 parameters:

$$\begin{aligned} G_1 &= s_1 + p_1 \\ G_2 &= s_2 + p_2 \\ G_3 &= 2p_1 \\ G_4 &= 2p_2 \\ G_5 &= \kappa_1 \end{aligned}$$

In de literatuur worden lensfoutmodellen beschreven met slechts één of twee parameters, die eveneens tot goede resultaten leiden. In deze toepassing is het echter raadzaam het niet-lineaire model te gebruiken omdat een eenvoudige 'off-the-shelf' CCD-camera wordt gebruikt. De hierin gebruikte lenzen zijn in het algemeen onnauwkeurig en bevatten een behoorlijke verstoring. Tijdens het fabricageproces worden aan de plaatsing van de lens eveneens minder hoge eisen gesteld zodat ook hierdoor een zekere vertekening van het beeld kan ontstaan.
In de volgende hoofstukken wordt het cameramodel genoteerd als $\underline{\hat{x}}_{im} = f(m, d)$. Hierin zijn $m = (\phi_1\ \phi_2\ \phi_3\ t_x\ t_y\ t_z)^T$ en $d = (f\ G_1\ G_2\ G_3\ G_4\ G_5)^T$ dit zijn respectievelijk kolommen met extrinsieke en intrinsieke parameters. $\underline{\hat{x}}_{im} = f(m, d)$ is de kolom met $\hat{x}_{im}$ en $\hat{y}_{im}$:

$$\underline{\hat{x}}_{im} = f(m, d) = \begin{pmatrix} \hat{x}_{im} \\ \hat{y}_{im} \end{pmatrix} \tag{2.19}$$

Samenvattend wordt het totale niet-lineaire cameramodel als volgt:

$$\hat{x}_{im} = f\frac{r_{11}x_w + r_{12}y_w + r_{13}z_w + t_x}{r_{31}x_w + r_{32}y_w + r_{33}z_w + t_z} + (G_1 + G_3)x_{im}^2 + G_4 x_{im}y_{im} + G_1 y_{im}^2 + G_1 x_{im}(x_{im}^2 + y_{im}^2)$$

$$\hat{y}_{im} = f\frac{r_{21}x_w + r_{22}y_w + r_{23}z_w + t_y}{r_{31}x_w + r_{32}y_w + r_{33}z_w + t_z} + G_2 x_{im}^2 + G_3 x_{im}y_{im} + (G_2 + G_4)y_{im}^2 + G_1 y_{im}(x_{im}^2 + y_{im}^2)$$

Hoofdstuk 3

Calibratieprocedure

Dit hoofdstuk geeft een beschrijving van de cameracalibratieprocedure die geïmplementeerd is in MATLAB. De programmatuur staat vermeld in de appendix waar tevens een korte beschrijving van de verschillende modules te vinden is. De calibratie van de cameraparameters vindt plaats aan de hand van het niet-lineaire cameramodel met zes intrinsieke en zes extrinsieke parameters dat gepresenteerd is in het vorige hoofdstuk.

3.1 De globale aanpak

Voor de berekening van de parameters staan twee gegevens ter beschikking, namelijk de posities van de meetpunten in wereldcoördinaten en de projecties op de CCD-sensor. De parameters worden bepaald door de *berekende* projecties van de meetpunten zo goed mogelijk te fitten op de *gemeten* projecties. De som van de residuen dient hiertoe geminimaliseerd te worden. De globale aanpak van dit niet-lineaire probleem is als volgt:

1. Stel de lensfoutparameters op nul en geef een schatting voor de brandpuntsafstand $d = (f\ 0\ 0\ 0\ 0\ 0)^T$ en bepaal een hiermee een beginschatting van de extrinsieke parameters m.

2. Bereken de intrinsieke parameters d met de extrinsieke parameters m vast.

3. Bereken de extrinsieke parameters m met de intrinsieke parameters d vast.

4. Begin weer bij punt 2 totdat na een aantal iteraties een voldoende nauwkeurige oplossing voor m en d is gevonden.

In elke iteratie worden de parameters bijgesteld tot een optimale oplossing bereikt is. De procedures voor de bepaling van de intrinsieke en extrinsieke parameters zijn in deze methode van elkaar losgekoppeld, dit is gedaan om numerieke instabiliteit tegen te gaan. De robuustheid van de methode is hierdoor toegenomen. Bepaling van de intrinsieke en

extrinsieke parameters in één procdure is vrijwel onmogelijk doordat er een zekere afhankelijkheid tussen alle parameters bestaat, waardoor het probleem numeriek zeer instabiel is. In literatuur staan verschillende methodes beschreven waarmee deze problemen omzeild kunnen worden. De hier gehanteerde methode is het meest robuust.

3.2 Lineaire oplossing van de extrinsieke parameters

Een beginschatting van de extrinsieke parameters wordt berekend met de intrinsieke parameters gelijk aan nul. Hier wordt dus uitgegaan van het onverstoorde cameramodel. De extrinsieke parameters worden verkregen door het oplossen van een stelsel lineaire vergelijkingen. Hiertoe worden vergelijking (2.3) en (2.4) omgeschreven tot:

$$x_{im}(r_{31}x_w + r_{32}y_w + r_{33}z_w) - f(r_{11}x_w + r_{12}y_w + r_{13}z_w) = 0 \tag{3.1}$$

$$y_{im}(r_{31}x_w + r_{32}y_w + r_{33}z_w) - f(r_{21}x_w + r_{22}y_w + r_{23}z_w) = 0 \tag{3.2}$$

Daarna kunnen (3.1) en (3.2) gedeeld worden door f:

$$u(r_{31}x_w + r_{32}y_w + r_{33}z_w) - (r_{11}x_w + r_{12}y_w + r_{13}z_w) = 0 \tag{3.3}$$

$$v(r_{31}x_w + r_{32}y_w + r_{33}z_w) - (r_{21}x_w + r_{22}y_w + r_{23}z_w) = 0 \tag{3.4}$$

Met hierin $u = \frac{x_{im}}{f}$ en $v = \frac{y_{im}}{f}$. Wanneer nu van n posities zowel de wereldcoördinaten $\underline{x}_w$ als de bijbehorende geschaalde beeldcoördinaten u_i en v_i kennen, dan kunnen relaties (3.3) en (3.4) als volgt in matrixvorm genoteerd worden:

$$\begin{pmatrix} x_{w,1} & y_{w,1} & 1 & 0 & 0 & 0 & -x_{w,1}u_1 & -y_{w,1}u_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{w,n} & y_{w,n} & 1 & 0 & 0 & 0 & -x_{w,n}u_n & -y_{w,n}u_n \\ 0 & 0 & 0 & x_{w,1} & y_{w,1} & 1 & -x_{w,1}v_1 & -y_{w,1}v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & x_{w,n} & y_{w,n} & 1 & -x_{w,n}v_n & -y_{w,n}v_n \end{pmatrix} \begin{pmatrix} r_{11} \\ r_{12} \\ t_x \\ r_{21} \\ r_{22} \\ t_y \\ r_{31} \\ r_{32} \end{pmatrix} \frac{1}{t_z} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \\ v_1 \\ \vdots \\ v_n \end{pmatrix} \tag{3.5}$$

Van de rotatiematrix worden slechts zes van de negen parameters bepaald. Dit zijn de parameters van de eerste en tweede kolom, de derde kolom kan bepaald worden uit de eerste twee doordat een rotatiematrix orthogonaal is. In dit stelsel zijn acht onbekenden en er zijn dus minimaal acht vergelijkingen nodig. Elk meetpunt heeft twee vergelijkingen namelijk één in x- en in één y-richting. Er zijn dus minimaal vier meetpunten nodig voor een oplossing. In deze stage is gebruik gemaakt van negen meetpunten zodat er een kleinste kwadraten oplossing bepaald wordt.
De rotatiematrix zoals deze nu bepaald is, is een beginschatting is dus *niet* bepaald met behulp van de drie Eulerhoeken maar met zes rotatieparameters die onderling afhankelijk zijn! Deze beginschatting wordt echter vanaf nu wel steeds *updated* met de drie onafhankelijke rotatieparameters ϕ_1 tot en met $\phi 3$. (Zie in de appendix **updat_r.m**).

3.3 Niet-lineaire oplossing van de intrinsieke parameters

De bepaling van de intrinsieke parameters vindt plaats met behulp van een gelineariseerd verband tussen $\underline{\hat{x}}_{im}$ naar d waarbij de extrinsieke parameters vast blijven. Deze lineairisatie is als volgt:

$$\underline{\hat{x}}_{im} = f(m,d) = f(m,\tilde{d}) + \frac{\partial f(m,\tilde{d})}{\partial d}(d-\tilde{d}) + ...\text{hoger orde termen} \tag{3.6}$$

Indien de hogere orde termen worden weggelaten kan dit geschreven worden als:

$$\underline{\tilde{x}}_{im} = \underline{\hat{x}}_{im} - \underline{x}_{im} = \tilde{U} + J_d(d-\tilde{d}) \tag{3.7}$$

Hierin is de J_d Jacobiaan matrix met de linearisatie van $\underline{\hat{x}}_{im}$ naar d. De Jacobiaan is in algemene termen:

$$J_d = \begin{pmatrix} \frac{\partial x_{im,1}}{\partial f} & \frac{\partial x_{im,1}}{\partial G_1} & \frac{\partial x_{im,1}}{\partial G_2} & \frac{\partial x_{im,1}}{\partial G_3} & \frac{\partial x_{im,1}}{\partial G_4} & \frac{\partial x_{im,1}}{\partial \kappa_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial x_{im,n}}{\partial f} & \frac{\partial x_{im,n}}{\partial G_1} & \frac{\partial x_{im,n}}{\partial G_2} & \frac{\partial x_{im,n}}{\partial G_3} & \frac{\partial x_{im,n}}{\partial G_4} & \frac{\partial x_{im,n}}{\partial \kappa_1} \\ \frac{\partial y_{im,1}}{\partial f} & \frac{\partial y_{im,1}}{\partial G_1} & \frac{\partial y_{im,1}}{\partial G_2} & \frac{\partial y_{im,1}}{\partial G_3} & \frac{\partial y_{im,1}}{\partial G_4} & \frac{\partial y_{im,1}}{\partial \kappa_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial y_{im,n}}{\partial f} & \frac{\partial y_{im,n}}{\partial G_1} & \frac{\partial y_{im,n}}{\partial G_2} & \frac{\partial y_{im,n}}{\partial G_3} & \frac{\partial y_{im,n}}{\partial G_4} & \frac{\partial y_{im,n}}{\partial \kappa_1} \end{pmatrix} \tag{3.8}$$

De berekening van de intrinsieke parameters vindt plaats met behulp van het verschil tussen de schatting en de gemeten posities in het beeldvlak:

$$\underline{\tilde{x}}_{im} = \underline{\hat{x}}_{im} - \underline{x}_{im} = J_d(d-\tilde{d}) \tag{3.9}$$

Hierin is $d = (f\ G_1\ G_2\ G_3\ G_4\ \kappa_1)^T$ de kolom met intrinsieke parameters. De berekening van de parameters is een iteratief proces. De procedure dient enkele malen achter elkaar doorlopen te worden tot een optimale oplossing is gevonden.

3.4 Niet-lineaire oplossing van de extrinsieke parameters

De procedure voor de bepaling van de extrinsieke parameters is vrijwel gelijk aan de methode voor de bepaling van de intrinsieke parameters. Nu blijven echter de intrinsieke parameters d vast en wordt gewerkt met een linearisatie van $\underline{\hat{x}}_{im}$ naar m. De linearisatie is als volgt:

$$\underline{\hat{x}}_{im} = f(m,d) = f(\tilde{m},d) + \frac{\partial f(\tilde{m},d)}{\partial m}(m-\tilde{m}) + ...\text{hoger orde termen}$$

De hogere orde termen worden weggelaten en zo wordt verkregen:

$$\underline{\hat{x}}_{im} = f(m,d) = \underline{x}_{im} + J_m(m - \tilde{m}) \tag{3.10}$$

De Jacobiaan matrix J_m is de linearisatie van $\underline{x}_{im}$ naar m. In algemene termen is deze:

$$J_m = \begin{pmatrix} \frac{\partial x_{im,1}}{\partial t_x} & \frac{\partial x_{im,1}}{\partial t_y} & \frac{\partial x_{im,1}}{\partial t_z} & \frac{\partial x_{im,1}}{\partial \phi_1} & \frac{\partial x_{im,1}}{\partial \phi_2} & \frac{\partial x_{im,1}}{\partial \phi_3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial x_{im,n}}{\partial t_x} & \frac{\partial x_{im,n}}{\partial t_y} & \frac{\partial x_{im,n}}{\partial t_z} & \frac{\partial x_{im,n}}{\partial \phi_1} & \frac{\partial x_{im,n}}{\partial \phi_2} & \frac{\partial x_{im,n}}{\partial \phi_3} \\ \frac{\partial y_{im,1}}{\partial t_x} & \frac{\partial y_{im,1}}{\partial t_y} & \frac{\partial y_{im,1}}{\partial t_z} & \frac{\partial y_{im,1}}{\partial \phi_1} & \frac{\partial y_{im,1}}{\partial \phi_2} & \frac{\partial y_{im,1}}{\partial \phi_3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial y_{im,n}}{\partial t_x} & \frac{\partial y_{im,n}}{\partial t_y} & \frac{\partial y_{im,n}}{\partial t_z} & \frac{\partial y_{im,n}}{\partial \phi_1} & \frac{\partial y_{im,n}}{\partial \phi_2} & \frac{\partial y_{im,n}}{\partial \phi_3} \end{pmatrix} \tag{3.11}$$

De oplossing wordt iteratief berekend met:

$$\underline{\tilde{x}}_{im} = \underline{\hat{x}}_{im} - \underline{x}_{im} = J_m(m - \tilde{m}) \tag{3.12}$$

Hierin is de vector met de extrinieke parameters: $m = (\phi_1 \ \phi_2 \ \phi_3 \ t_x \ t_y \ t_z)^T$. De procedure dient enkele malen achter elkaar doorlopen te worden tot een optimale oplossing is gevonden.

Opmerkingen

De hierboven beschreven procedures zijn geïmplementeerd in MATLAB in de volgende modules:

- **iniest.m** (=initial estimate) voor de bepaling van de beginschatting van de extrinsieke parameters.
- **op_in.m** voor de bepaling van de intrinsieke parameters met behulp van de Jacobiaan J_m.
- **op_ex.m** voor de bepaling van de extrinsieke parameters met behulp van de Jacobiaan J_d.

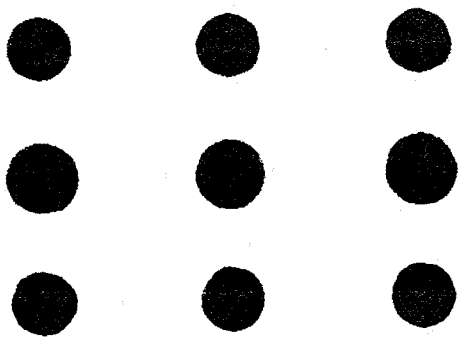
De methode is uitgebreid getest en is steeds zeer stabiel gebleken. De convergentie is in vrijwel alle gevallen zeer goed. Een voldoende nauwkeurig resultaat wordt in het algemeen bereikt na maximaal 5 keer de routines te hebben doorlopen. De resultaten van de calibratie komen aan de orde in het volgende hoofstuk.

Hoofdstuk 4

Resultaten

Dit hoofstuk geeft een overzicht van de resultaten die bereikt worden met de in deze stage geschreven calibratiemethode. Vergeleken worden de resultaten die bereikt worden met een camermodel zonder lensverstoringen en cameramodel met *lensverstoring. De beide cameramodellen worden beoordeeld op de verschillen tussen de gemeten en berekende posities.*

De calibratie is uitgevoerd met een patroon van negen stippen op een A4-vel. Zie figuur(4.1). De grafische resultaten staan weergegeven in figuur (4.2) en figuur(4.3). In deze figuren



Figuur 4.1: Calibratiepatroon

zijn de '+' tekens de *gemeten* projecties op het beeldvlak en de '*' tekens de *berekende* projecties op het beeldvlak uitgaande van de meetpunten in wereldcoördinaten. Het doel van de calibratie is natuurlijk de gemeten en berekende beeldpunten zo goed mogelijk met elkaar overeen te laten komen. Het verschil tussen beide cameramodellen is in eerste instantie in de figuur niet duidelijk zichtbaar, de bereikte resultaten staan dan ook vermeld

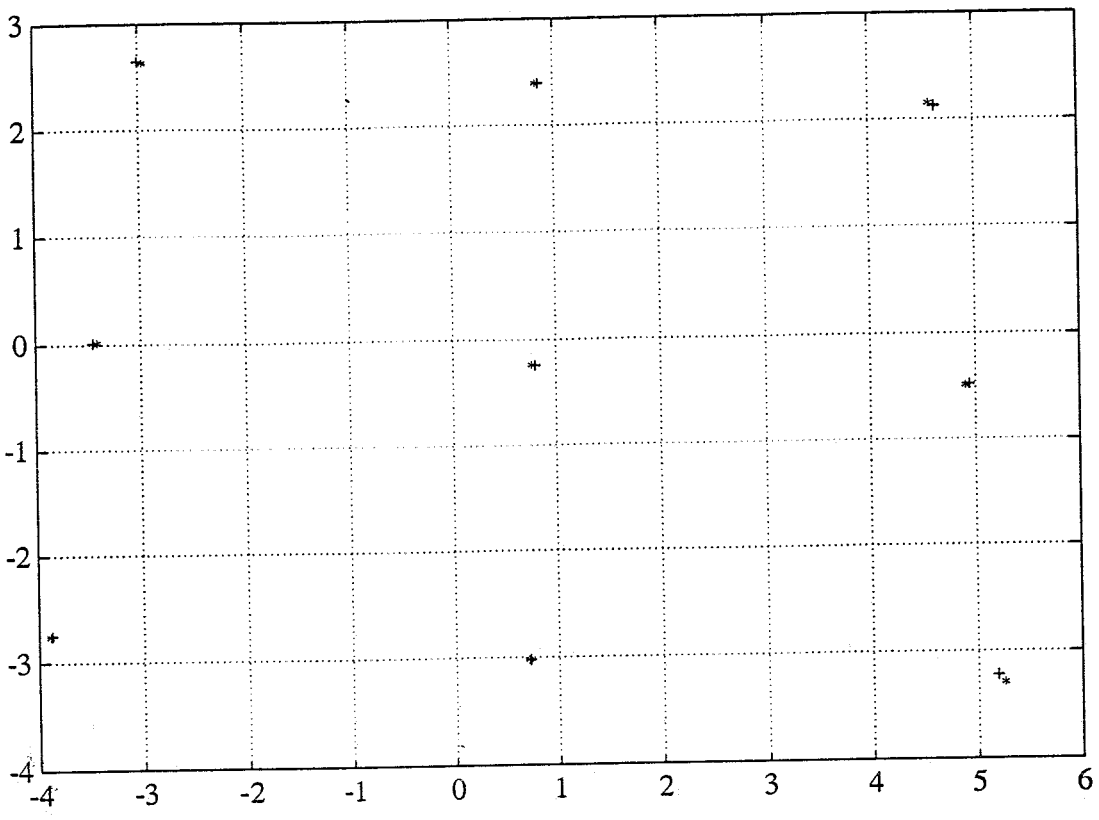
in tabel(4.1). Er is gekeken naar het gemiddelde van het absolute verschil tussen de gemeten en berekende posities in x- en y-richting. Daarnaast is de maximale afwijking bekeken en een schatting gemaakt voor de standaarddeviatie. Deze grootheden staan weergeven in milimeters in het beeldvlak. Om een uitspraak te kunnen doen over de nauwkeurigheid van het meetsysteem dienen deze waarden omgerekend te worden naar milimeters op het tafelvlak. Een benaderingsformule hiervoor is:

$$a_{tafel} \approx \frac{a_{im} t_z}{f} \tag{4.1}$$

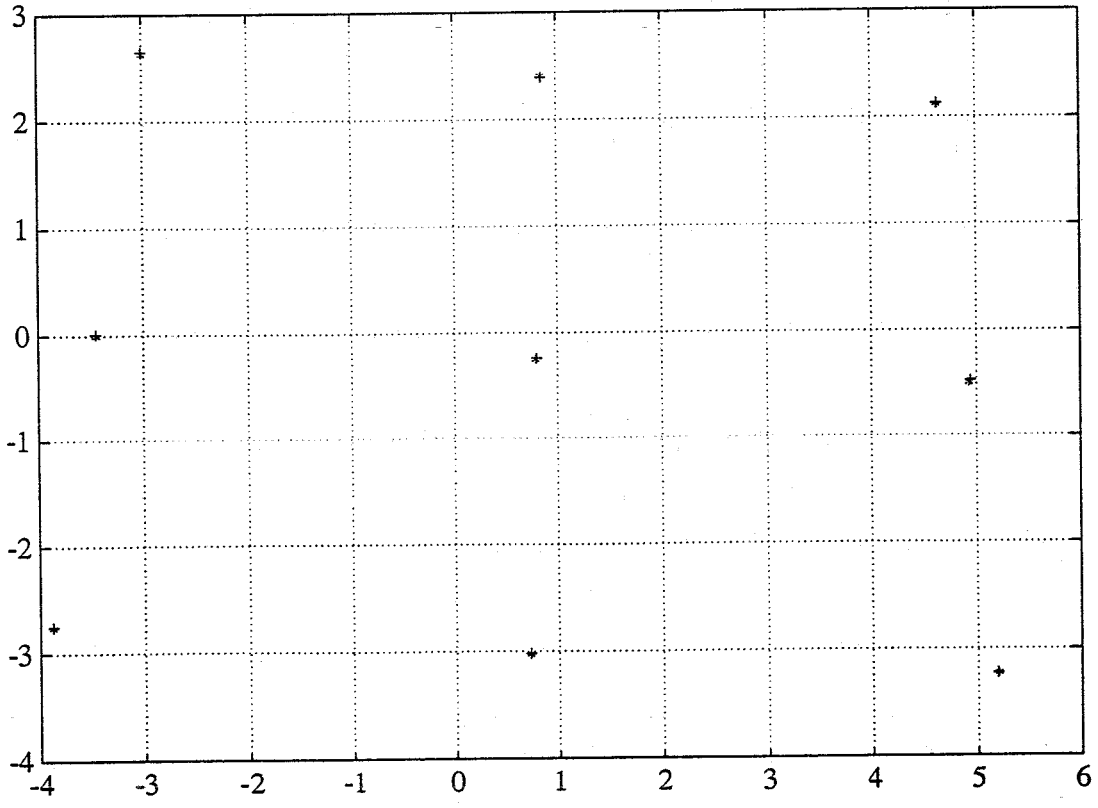
Hierin is a_{im} de maximale afwijking in het beeldvlak en a_{tafel} de afwijking op het tafelvlak dat zich op een afstand t_z van de camera bevindt. De te bereiken nauwkeurigheid met het cameramodel zonder lensverstoring is ongeveer $a_{tafel} = \frac{0.0876*257}{17} = 1.34$ [mm] en met het camermodel met lensverstoring $a_{tafel} = \frac{0.0207*251}{17} = 0.31$ [mm]. Het meenemen van de lensverstoring levert dus een aanzienlijke verbetering op. In deze stage zijn vele tests uitgevoerd met verschillende oriëntaties en posities van de camera ten opzichte van het tafelvlak. De bereikte resultaten zijn representatief voor de nauwkeurigheid van het meetsysteem.

Tabel 4.1: Resultaten camera calibratie

Resultaten Calibratie			
alle grootheden in [mm]			
cameramodel		ideaal	verstoord
maximum residu	$\max \lvert\tilde{x}_{im}\rvert$	0.0352	0.0045
	$\max \lvert\tilde{y}_{im}\rvert$	0.0152	0.0075
maximale afwijking	$\sqrt{\tilde{x}_{im}^2 + \tilde{y}_{im}^2}$	0.0876	0.0207
standaard dev.	$\sigma_{\tilde{x}}$	0.0388	0.0063
	$\sigma_{\tilde{y}}$	0.0234	0.0096
translaties	t_x	11.6298	11.7291
	t_y	-3.5478	-3.7050
	t_z	256.8762	251.1575
brandpuntsafstand	f	17.0000	16.9989
lensfoutparameters	G_1	0.0000	0.0002
	G_2	0.0000	-0.0003
	G_3	0.0000	-0.0008
	G_4	0.0000	0.0045
	G_5	0.0000	-0.0007



Figuur 4.2: Resultaten: cammeramodel zonder lensverstoring



Figuur 4.3: Resultaten: cameramodel met lensverstoring

Hoofdstuk 5

Conclusies en aanbevelingen

5.1 Conlusies

Aan de hand van deze stage kunnen de volgende conclusies worden getrokken:

- Het gebruik van een cameramodel met lensverstoring verdient de voorkeur boven dat zonder lensverstoring. Dit levert betere resultaten zonder veel extra inspanningen.
- De ontkoppeling van de bepaling van intrinsieke en extrinsieke parameters geeft een robuuste cameracalibratie.

5.2 Aanbevelingen

Naar aanleiding van deze stage kunnen de volgende aanbevelingen gedaan ten aanzien van de kanteltafel:

- De extra monitor die het beeld van de framegrabber direct weergeeft kan belangrijke informatie verschaffen over de lichtcondities voor de camera. En zou, indien de camera vast is opsteld toch voor dit doel behouden moeten blijven.
- De bepaling van de intrinsieke parameters kan betrouwbaarder gemaakt worden door een groter aantal meetpunten gebruiken. Hiernaar is in deze stage geen verder onderzoek verricht.
- Verder onderzoek naar het effect van calibratie van de verschuivingsfactoren C_x en C_y. De bevestiging van de lens bevat veel speling. Daardoor zal het snijpunt van de optische as en het beeldvlak nooit exact in het centrum van het framegrabberbeeld liggen. Indien nu ook de verschuivingsfactoren C_x en C_y gecalibreerd zouden worden zou de nauwkeurigheid van metingen kunnen toenemen. Dit kan echter ook weer numerieke instabiliteit in de calibratieprocedure tot gevolg hebben.

Bibliografie

[Tsai] Roger Y. Tsai
A Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses
IEEE journal of robotics and automation
August 1987, volume RA-3, nummer 4

[Lenz] Reimar K. Lenz
Lens Distortion Corrected CCD-Camera Calibration with Co-Planar Calibration Points for Real-Time 3D Measurements
Proceedings of the ISPRS 1987

[Tsai & Lenz] Reimar K. Lenz & Roger Y. Tsai
Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3D Machine Vision Metrology
IEEE transactions on pattern analysis and machine intelligence
September 1988, volume 10, nummer 5

[Weng, Cohen, Herniou] Juyang Weng, Paul Cohen, Marc Herniou
Calibration of Stereo Cameras using a Non-Lineair Distortion Model
IEEE int. conference on pattern recognition
Juni 1990, Atlantic City, New Jersey, USA

[Otto] Pim Otto
Camera Calibration, A Method for Calibrating the Intrinsic end Extrinsic Camera Parameters
Stageverslag Technische Universiteit Eindhoven, Faculteit Elektrotechniek
Eindhoven, Juli 1993

[Hanajik] Milan Hanajík
Softwaremodule in C voor calibratie intrinsieke en extrinieke parameters van een CCD-camera: CALIB.C
TU Eindhoven, Faculteit Elektrotechniek, Vakgroep Meten en Regelen
versie Augustus 1993.

[Stap] J.W. Stap
Automatic 3D Measurement of Workpieces on a Ship Repair Yard
Afstudeerverslag Technische Universiteit Eindhoven, Faculteit Elektrotechniek
Eindhoven, Juni 1992

[v. Vliet] R.G. van Vliet
Dictaat: Beeldverwerking
TU Eindhoven, Faculteit Elektrotechniek, Vakgroep Meten en Regelen
uitgave 1990, dictaat nr. 5734

[DT] Data Translation
Handleiding: DT 2851 framegrabber
Data Translation, 1992

[Borland] Borland
Turbo C++, library reference
Borland International, 1990

[Philips] Philips
Handleiding: CCD Observation Camera LDH 0703/3x
Eindhoven, Philips Communication & Security Systems

[CWI] Centrum voor Wiskunde en Informatica
Publiceren met LaTeX
Stichting Mathematisch Centrum, Amsterdam 1988

Bijlage A

Softwarebeschrijving

In deze stage is gewerkt aan software voor de calibratie van het meetsysteem van de kanteltafel. De geschreven en bewerkte programma's zijn in deze appendix afgedrukt en worden kort toegelicht.

A.1 Positiebepaling van meetpunten

De bepaling van de positie van de meetpunten vindt plaats met de module **camcal.c** die geschreven is in C++. Dit programma maakt deel uit van een hoofdprogramma dat geschreven is door Ing. Niels Olthuis.

De werking van de module is als volgt: Het framegrabberbeeld wordt verdeeld in negen windows van gelijke afmetingen. De gebruiker dient te zorgen dat elk meetpunt zich in het beeld in een van de windows bevind. De grijswaarde van elk pixel in het window wordt geëvalueerd en nu wordt via een zwaartepuntsberekening het middelpunt van een meetpunt bepaald. Dit vindt, per window afwisselend in het even en oneven raster, plaats. Deze bewerking wordt tien maal achter elkaar uitgevoerd zodat een gemiddelde positie van de meetpunten verkregen wordt. Hiermee worden eventuele verstoringen uitgemiddeld. De verkregen posities in het beeldvlak van de meetpunten worden naar een MATLAB-file geschreven. De posities zijn weergegeven in framegrabber coordinaten (pixels).

Opmerkingen

De module **camcal.c** kan eenvoudig aangepast worden aan een ander aantal meetpunten. De lichtcondities voor de camera zijn van groot belang voor een succesvolle uitvoer van **camcal.c**. De verdere bewerkingen voor de bepaling van intrinsieke en extrinsieke parameters worden uitgevoerd met MATLAB.

camcal.c

```
//      camcal.c
//      ========
#include "cam.h"
#include <tce.h>
#include <conio.h>
#include <dos.h>
#include <stdio.h>

void    CAM::cam_calib(void)    {

        interval=2*int(NX);

        float x_ca[9]={0,0,0,0,0,0,0,0,0};
        float y_ca[9]={0,0,0,0,0,0,0,0,0};
        double x_sum_ca[9]={0,0,0,0,0,0,0,0,0};
        double y_sum_ca[9]={0,0,0,0,0,0,0,0,0};
        double x_av_ca[9]={0,0,0,0,0,0,0,0,0};
        double y_av_ca[9]={0,0,0,0,0,0,0,0,0};
        double tij[9];

        int p;
        int xw1[9]={0  ,171,341,  0,171,341,  0,171,341};
        int xw2[9]={170,340,512,170,340,512,170,340,512};
        int yw1[9]={  0,  0,  0,171,171,171,341,341,341};
        int yw2[9]={170,170,170,340,340,340,512,512,512};

        clrscr();
        printf("\n x_cal_array    y_cal_array   time  \n");

for (k=1;k<11;k++){
        outport(INCSR1,0x0099);
        while((inport(INCSR1))&0x0080);
        for (i=0;i<9;i++){
                x1=xw1[i];          /* definition of the window  */
                x2=xw2[i];
                y1=yw1[i];
                y2=yw2[i];

                     if (k&1){
                             v=v_init+y1*NX;            /* start adress      */
                             tim_t0();
                             get_pos_calc();            /* get pos even field */
                             tij[i]=tim_t1();
```

```
            }


            else{                                   /* even field busy     */

                    y1=y1+1;                        /* odd line     */
                    y2=y2+1;                        /* odd line     */
                    v=v_init+y1*NX;                 /* start adress */
                    tim_t0();
                    get_pos_calc();                 /* get pos odd field   */
            }

        tij[i]=tim_t1();

        x_ca[i]=(float)x_pos/16.0;
        y_ca[i]=(float)y_pos/16.0;

        x_sum_ca[i]=x_sum_ca[i]+x_ca[i];
        y_sum_ca[i]=y_sum_ca[i]+y_ca[i];


}

      printf(" \n ");

    for (p=0;p<4;p++) {
    printf("  %7.2lf    %7.2lf      %7.4lf \n ",x_ca[p],y_ca[p],tij[p]);
    }

}
   for (p=0;p<9;p++){
   x_av_ca[p]=x_sum_ca[p]/10.0;
   y_av_ca[p]=y_sum_ca[p]/10.0;
   }

   printf("\n average x_calibration:\n");
   printf(" %7.2lf %7.2lf %7.2lf %7.2lf \n ",x_av_ca[0],....
                     .....x_av_ca[1],x_av_ca[2],x_av_ca[3]);
   printf("\n average y_calibration: \n");
   printf(" %7.2lf %7.2lf %7.2lf %7.2lf \n ",y_av_ca[0],....
                     .....y_av_ca[1],y_av_ca[2],y_av_ca[3]);

    ml_open("calidat",1);
```

```
            ml_put_vec(x_av_ca,9,"xc");
            ml_put_vec(y_av_ca,9,"yc");
            ml_close();
}

//      end of camcal.h
```

A.2 Cameracalibratie

Als basis voor de calibratieprocedure is gebruik gemaakt van een programma ontwikkeld bij de Faculteit Elektrotechniek, zie [Hanajik]. Dit programma is geschreven door Ing. Milan Hanajík. Bij de vakgroep Meten en Regelen is Milan Hanajík bezig met de calibratie van camerameetsystemen die gebruikt worden voor de automatisering van lasrobots. In deze stage is gewerkt aan een calibratieprogramma dat geprogrammeerd wordt in MATLAB. Er is gekozen voor MATLAB omdat bij de calibratie, tijdsduur geen rol speelt (MATLAB is veel langzamer dan C++), daarnaast is een MATLAB programma toegankelijker voor gebruikers met weinig programmeerervaring.

De calibratieprocedure in MATLAB is als volgt opgebouwd: Het programma **main.m** is het hoofdprogramma waarin de gebruiker de gewenste calibratie kan kiezen. De gebruiker kan kiezen uit de calibratie van alleen de extrinsieke parameters maar ook calibratie van extrinsieke en intrinsieke parameters. Dit laatste zal echter minder vaak nodig zijn. De procedures dragen de namen **cal_ex** en **cal_all**. Deze maken weer gebruik van elf subprogramma's. Hier volgt een overzicht van de subprogramma's

1. **op_ex.m:** optimaliseert extrinsieke parameters zoals beschreven in hoofdstuk 3.
2. **op_in.m:** optimaliseert intrinsieke parameters zoals beschreven in hoofdstuk 3.
3. **mm_pix.m:** rekent metrische beeldcoördinaten om naar framegrabber coördinaten.
4. **pix_mm.m:** rekent framegrabbercoördinaten om naar metrische beeldcoördinaten.
5. **iniest.m:** berekent een beginschatting van de extrinsieke parameters (rotatiematrix R en translatievector T) zoals beschreven in hoofdstuk 3.
6. **dis_bas.m:** berekent de basis van de lensverstoring (wordt o.a. toegepast voor de routines **dis_val.m** en **op_all**).
7. **dis_val.m:** berekent de verstoring in x- en y-richting ten gevolge van de lensfout (door gebruik te maken van **dis_bas.m**
8. **update_r.m:** berekent de aanpassing van de van de rotatiematrix, die met de gegevens uit de routine **op_ex.m** kan worden uitgevoerd.
9. **comp_res.m:** berekent de residuen tussen de gemeten beeldpunten en de met het model berekende meetpunten.
10. **ev_res.m:** bepaalt de maximale afwijking, het gemiddelde en de standaardeviaties van de residuen, in x- en y-richting.
11. **plot_res.m:** geeft grafisch de resultaten weer tussen de gemeten en berekende meetpunten.

Het principe van de cameracalibratie staat beschreven in hoofdstuk 3 en zal hier dan ook niet verder uitgelegd worden.

op_ex.m

```
% Subroutine to optimize extrinsic camera parameters. Called with:
%    [T]=op_ex(M,W,T,Ci0,Dpar)

[n,m]=size(M);                  % size of matrix M

Dbas=dis_bas(M);
Dist=dis_val(M,Dpar,Dbas);

%------wr=rotation only, wrt=rotation and translation-------------------
wr=W*T(:,1:3)';                 % rotation only
wrt=wr+ones(W)*diag(T(:,4));    % rotation and translation=camera coordinates
wsq=wrt(:,3).*wrt(:,3);         % square of z-cam

%------compute residuals-----------------------------------------------
R=comp_res(M,Ci0,wrt,Dist,n);

%------compute jacobian------------------------------------------------
J(1:n,1)      =Ci0(5)./wrt(:,3);                                    % drx/dtx
J((n+1):2*n,1)=zeros(n,1);                                          % dry/dtx
J(1:n,2)      =zeros(n,1);                                          % drx/dty
J((n+1):2*n,2)=Ci0(5)./wrt(:,3);                                    % dry/dty
J(1:n,3)      =-Ci0(5).*wrt(:,1)./wsq;                              % drx/dtz
J((n+1):2*n,3)=-Ci0(5).*wrt(:,2)./wsq;                              % dry/dfz
J(1:n,4)      =Ci0(5).*(-wrt(:,3).*wr(:,3)-wrt(:,1).*wr(:,1))./wsq; % drx/df1
J((n+1):2*n,4)=Ci0(5).*(-wrt(:,2).*wr(:,1))./wsq;                   % dry/df1
J(1:n,5)      =Ci0(5).*(-wr(:,2).*wrt(:,3))./wsq;                   % drx/df2
J((n+1):2*n,5)=Ci0(5).*(wr(:,1).*wrt(:,3))./wsq;                    % dry/df2
J(1:n,6)      =Ci0(5).*(-wrt(:,1).*wr(:,2))./wsq;                   % drx/df3
J((n+1):2*n,6)=Ci0(5).*(-wrt(:,3).*wr(:,3)-wrt(:,2).*wr(:,2))./wsq; % dry/df3

Sol=J\R;

%------update transformationmatrix-------------------------------------
T(:,4)=T(:,4)+Sol(1:3);
T(:,1:3)=updat_R(T(:,1:3),Sol(4:6));

%------transform worldcoordinates into cameracoordinates---------------
%------with updated T--------------------------------------------------
wr=W*T(:,1:3)';                 % rotation
wrt=wr+ones(W)*diag(T(:,4));    % rotation+translation=camera coordinates
```

op_in.m

```
% Subroutine to optimize intinsic and extrinsic camera parameters.

Dbas=dis_bas(M);
Dist=dis_val(M,Dpar,Dbas);

%------compute residuals-------------------------------------------------
R=comp_res(M,Ci0,wrt,Dist,n);

%------compute jacobian--------------------------------------------------
J(1:n,1)       =wrt(:,1)./wrt(:,3); % drx/df
J((n+1):2*n,1) =wrt(:,2)./wrt(:,3); % dry/df
J(1:n,2)       =Dbas(:,1) ;         % drx/dG1
J((n+1):2*n,2) =Dbas(:,2) ;         % dry/dG1
J(1:n,3)       =Dbas(:,3) ;         % drx/dG2
J((n+1):2*n,3) =Dbas(:,4) ;         % dry/dG2
J(1:n,4)       =Dbas(:,5) ;         % drx/dG3
J((n+1):2*n,4) =Dbas(:,6) ;         % dry/dG3
J(1:n,5)      =Dbas(:,7) ;          % drx/dG4
J((n+1):2*n,5)=Dbas(:,8) ;          % dry/dG4
J(1:n,6)      =Dbas(:,9) ;          % drx/dkappa1
J((n+1):2*n,6)=Dbas(:,10);          % dry/dkappa1

Sol=J\R;

%------COMPUTE UPDATES---------------------------------------------------
Ci0(5)=Ci0(5)+Sol(6);
Dpar=Dpar+Sol(2:6)';
```

mm_pix.m

```
% Subroutine to transform metric image coordinates into pixelcoordinates.
% Called with:
%
%   [P]=mm_pix(M,Ci0);
%
% P    : pixelcoordinates
%
% M    : metric imagecoordinates
% Ci0  : CCD camera parameters Sx Sy Cx Cy and f

function [P]=mm_pix(M,Ci0);
```

```
P(:,1)=M(:,1)/Ci0(1)+Ci0(3)
P(:,2)=M(:,2)/Ci0(2)+Ci0(4)
```

pix_mm.m

```
% Subroutine to transform pixelcoordinates into metric image coordinates.
% Called with:
%
%    [M]=pix_mm(P,Ci0)
%
% M    : metric image coordinates
%
% P    : pixelcoordinates
% Ci0  : CCD camera parameters Sx Sy Cx Cy and f

function [M]=pix_mm(P,Ci0);

M(:,1)=Ci0(1)*(P(:,1)-Ci0(3)*ones(P(:,1)));
M(:,2)=Ci0(2)*(P(:,2)-Ci0(4)*ones(P(:,2)));
```

dis_bas.m

```
% Subroutine to compute lensfault distortion basis used...
% ...in Jacobian matrix. Called with:
%
%   [Dbas]=dis_basl(M)
%
% Dbas  : Distortion basis
%
% M     : Metric image coordinates

function [Dbas]=dis_val(M)

[n,m]=size(M);
Dbas=zeros(n,10);

% lensfault basis
Dbas(:,1)=M(:,1).*M(:,1)+M(:,2).*M(:,2); % xx+yy
Dbas(:,2)=zeros(n,1);                    % zeros
Dbas(:,3)=zeros(n,1);                    % zeros
Dbas(:,4)=Dbas(:,1);                     % xx+yy
Dbas(:,5)=M(:,1).*M(:,1);                % xx
```

```
Dbas(:,6)=M(:,1).*M(:,2);                % xy
Dbas(:,7)=Dbas(:,6);                     % xy
Dbas(:,8)=M(:,2).*M(:,2);                % yy
Dbas(:,9)=M(:,1).*Dbas(:,1);             % x(xx+yy)
Dbas(:,10)=M(:,2).*Dbas(:,1);            % y(xx+yy)
```

dis_val.m

```
% Subroutine to compute lensfault distortion values using
% distortion basis computed in 'dis_bas.m'. Called with:
%
%   [Dist]=dis_val(M,Dpar,Dbas)
%
% Dist  : Distortion values, first column delta(x), second column delta(y)
%
% M : Metric image coordinates
% Dpar  : Vector with five lensfault parameters
% Dbas  : Distortionbasis

function [Dist]=dis_val(M,Dpar,Dbas)


% lensfault values
Dist(:,1)=Dpar(1)*Dbas(:,1)+Dpar(2)*Dbas(:,5)+Dpar(4)*Dbas(:,7)+Dpar(5)*Dbas(:,9);
Dist(:,2)=Dpar(2)*Dbas(:,4)+Dpar(3)*Dbas(:,6)+Dpar(4)*Dbas(:,8)+Dpar(5)*Dbas(:,10)
```

iniest.m

```
% Subroutine to compute initial estimate for rotation and translation
% matrix.  Called with:
%
% T=iniest(M,W,I,Ci0)
%
% T    : initial estimate of transformationmatrix
%
% M    : metric imagecoordinates computed from pixelcoordinates
% W    : world coordinates known from data-file
% I    : index of calbration point
% Ci0  : CCD camera fixed parameters Sx Sy Cx Cy and f

function [T]=iniest(M,W,I,Ci0);

O=ones(I)';
```

```
X=W(:,1);Y=W(:,2);Z=W(:,3);

U=M(:,1)/Ci0(5);
V=M(:,2)/Ci0(5);

A=[X Y 0 Z Z Z -X.*U -Y.*U
   Z Z Z X Y 0 -X.*V -Y.*V ];
B=[-U; -V];
p=-A\B;

T=[p(1) p(2) 0 p(3);p(4) p(5) 0 p(6);p(7) p(8) 0 1];
c=sqrt(norm(T(:,1))*norm(T(:,2)));
T=T/c;
T(:,1)=T(:,1)-T(:,2)*(T(:,1)'*T(:,2))/2;
T(:,2)=T(:,2)-T(:,1)*(T(:,2)'*T(:,1));
T(:,1)=T(:,1)/norm(T(:,1));
T(:,2)=T(:,2)/norm(T(:,2));

T(1,3)=det(T([2,3],[1,2]));
T(2,3)=det(T([3,1],[1,2]));
T(3,3)=det(T([1,2],[1,2]));
```

comp_res.m

```
% Subroutine to compute residuals. Called with:
%
%  [R]=comp_res(M,Ci0,wrt,Dist)
%
%   R    : Residuals
%
%   M    : Metric image coordinates
%   Ci0  : CCD camera parameters
%   Dist : vector distortion of image points
%   n    : number of calibration points

function    [R]=comp_res(M,Ci0,wrt,Dist,n)

R(1:n,1)      =M(:,1)-Ci0(5)*wrt(:,1)./wrt(:,3)-Dist(:,1);
R((n+1):2*n,1)=M(:,2)-Ci0(5)*wrt(:,2)./wrt(:,3)-Dist(:,2);
```

ev_res.m

```
% Subroutine to evaluate the residuals after the computation of the new
```

```
% extrinsic parameters. Called with:
%
%   E=ev_res(R,n)
%
%  E   : vector [max_x  max_y  average_x  average_y  sd_x  sd_y]
%
%  R   : residuals in x and y-direction
%  n   : number of calibration points

function  [E]=ev_res(R,n)

sumxx=0;
sumyy=0;

for k=1:n,
  sumxx=sumxx+R(k)^2;
  sumyy=sumyy+R(k+n)^2;
end

E(1)=max(abs(R(1:n)));
E(2)=max(abs(R((n+1):2*n)));
E(3)=mean(R(1:n));
E(4)=mean(R((n+1):2*n));
E(5)=sqrt(sumxx/n);
E(6)=sqrt(sumyy/n);
```

updat_r.m

```
% Subroutine to update rotation matrix R, which is a submatrix of T. This
% will take place in three steps, rotation with every angle separately.
% Called with:
%
%   [Tsub]=updat_R(Tsub,ang)
%
% 'input'  T   : transformationmatrix to be updated
%
% 'output' T   : transformationmatrix with updated rotation submatrix
%          ang : angles for updating the rotationmatrix



function  [Tsub]=updat_R(Tsub,ang)
```

```
T3=[        1            0            0
            0    cos(ang(3)) -sin(ang(3))
            0    sin(ang(3))  cos(ang(3))];

T2=[cos(ang(2)) -sin(ang(2))          0
    sin(ang(2))  cos(ang(2))          0
            0            0            1  ];

T1=[cos(ang(1))          0    -sin(ang(1))
            0            1             0
    sin(ang(1))          0     cos(ang(1))];

Tsub=T1*T2*T3*Tsub;
```

plotres.m

```
% Subroutine to plot results

x_im=Ci0(5)*wrt(:,1)./wrt(:,3);
y_im=Ci0(5)*wrt(:,2)./wrt(:,3);
x_im_d=x_im+Dist(:,1);
y_im_d=y_im+Dist(:,2);
plot(M(:,1),M(:,2),'+',x_im_d,y_im_d,'*');grid;
title('+ gemeten ,* berekend')
```