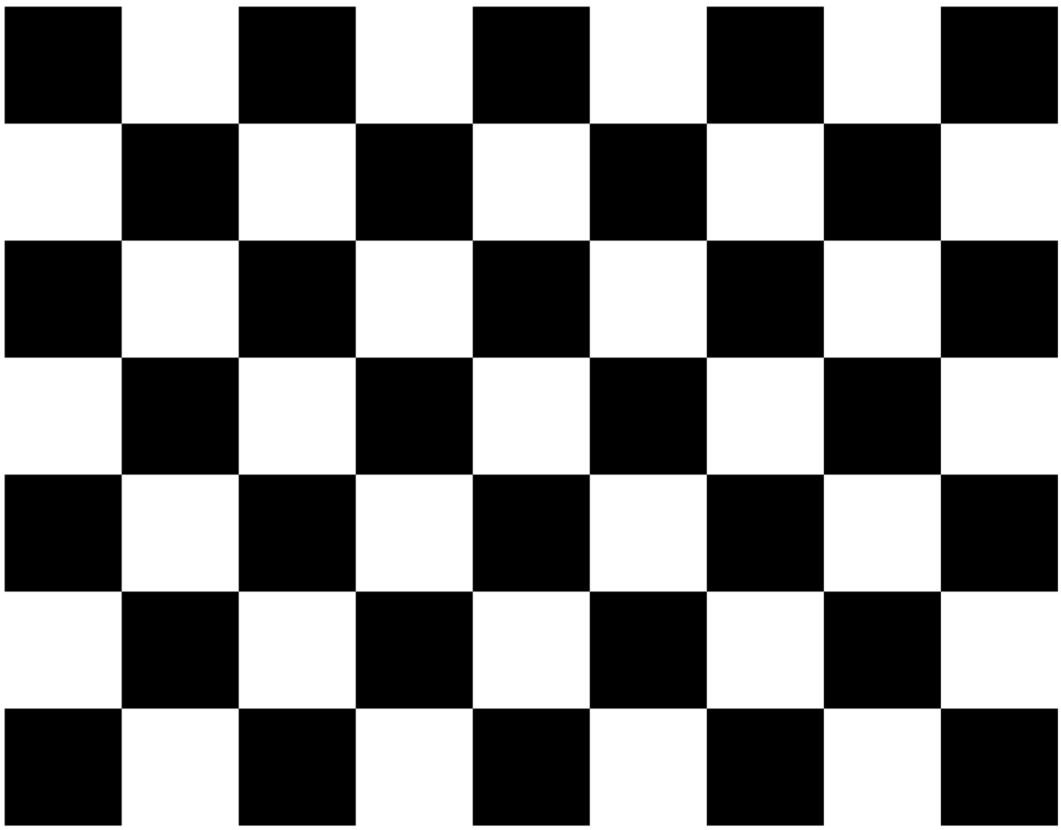
EWI/SAS 2013-021

# Kalibratie van de camera in een endoscoop en een 6DOF-EM-sensor

*Bacheloropdracht van T.F. Kerkhof*
*Bacheloropdrachtcommissie:*
*Voorzitter: C.H.Slump (SAS)*
*Mentor: F. van der Heijden (SAS)*
*Lid van andere werkeenheid: N. van der Stap (MISR)*



Plaatje op het voorblad is afkomstig van:
http://www.robot.sh.cn/attachments/camera_calibration(2f)Tutorials(2f)StereoCalibration/check-108.png

# Inhoudsopgave

# 1. Inleiding

De minister van Volksgezondheid, Welzijn en Sport, Edith Schippers, wilt een grootschalig bevolkingsonderzoek naar dikke darmkanker gaan invoeren, omdat dit op termijn 2.400 sterfgevallen per jaar kan voorkomen **[1]**. Hierdoor komen er op den duur per jaar 66.000 extra colonoscopiën bij, 28.000 extra in 2014 tot 74.000 in 2019 **[2]**. Het kost veel tijd en oefening om een endoscoop goed te leren besturen. Een gastro-enteroloog heeft 100 tot 500 colonoscopiën gedaan over een periode van 1 tot 3 jaar voor hij/zij de besturing en andere functies onder de knie heeft **[3-5]**. Om op termijn het grote aantal colonoscopiën uit te voeren, wordt er gekeken naar methodes om de besturing van de endoscoop te vereenvoudigen.

Op Universiteit Twente is een model gemaakt die de *focus of expansion* tussen twee beelden vindt **[6]**. Dit model kan gebruikt worden om tijdens een colonoscopie te meten waar de punt van de endoscoop naar toe beweegt. Zo kan de navigatie van de endoscoop makkelijker gemaakt worden.

Om het model van N. van der Stap et al. **[6]** te testen wordt gebruik gemaakt van het Aurora Electromagnetic Tracking System van Northern Digital Inc. (NDI). Een elektromagnetische (EM) sensor wordt op de punt van de endoscoop geplakt. Deze EM-sensor geeft zijn positie en oriëntatie op een bepaald moment in een magnetisch veld gemaakt door NDI-fieldgenerator. Door te meten waar de sensor naar toe beweegt moet te achterhalen zijn of de beweging in de werkelijkheid overeen komt met de beweging gevonden door het model.

Om de beweging die gevonden is door het model te kunnen vergelijken met de beweging gevonden door de EM-sensor, moeten de twee metingen in hetzelfde assenstelsel staan. De camera geeft data in twee dimensies: het beeld in pixels in twee richtingen of X en Y. De EM-sensor geeft data in zes dimensies: de plaats en de oriëntatie of X, Y en Z en Quaternions of Euler Angles.

Dit verslag gaat over de transformatie tussen de camera en de EM-sensor en de manier om deze vast te stellen.

## 2. Probleemstelling

De EM-sensor is op de punt van de endoscoop naast de camera bevestigd. De EM-sensor meet de plaats en oriëntatie van de sensor. De camera geeft beeld. Omdat de sensor op een vaste plaats naast de camera zit, kan een vaste transformatiematrix gemaakt worden, die de zesdimensionale (6D) data in het assenstelsel van de tweedimensionale (2D) data van de camera plaatst.

Deze transformatie is niet in één keer te vinden, omdat de afstanden op het beeld niet direct te correleren zijn met de afstanden in de ruimte. Daarom wordt er een extra assenstelsel toegevoegd die door beide meetmethoden te meten zijn. Door foto's te nemen van een schaakbord, kan de plaats en oriëntatie van een camera ten opzichte van dat schaakbord bepaald worden **[7]**. Met een EM-pointer-sensor kan de plaats van het schaakbord in het assenstelsel van de NDI-Fieldgenerator gemeten worden. De transformatie wordt daarom in vier stappen gevonden, zie *figuur 1*:

### Stap 1: EM-sensor naar NDI-fieldgenerator

De 6D EM-sensor geeft zijn plaats en oriëntatie in het assenstelsel van de NDI-fieldgenerator. Uit deze plaats en oriëntatie is de transformatiematrix te maken, die de transformatie van het assenstelsel van de EM-sensor naar het assenstelsel van de NDI-fieldgenerator beschrijft.

### Stap 2: NDI-fieldgenerator naar schaakbord

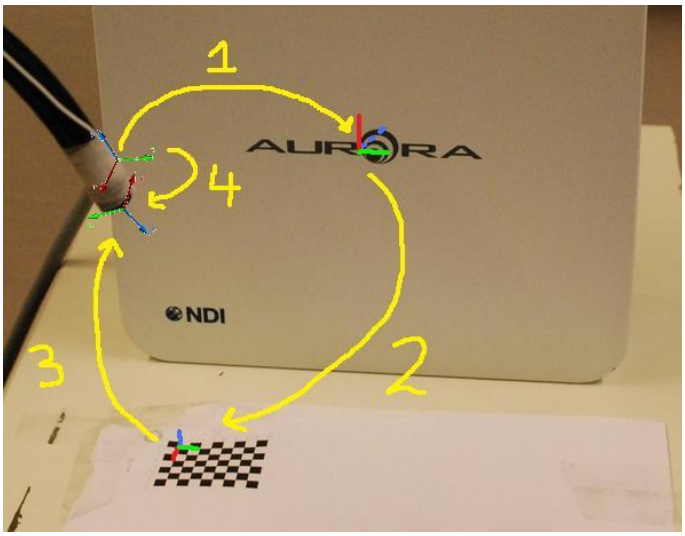
Het schaakbord wordt met een EM-pointer-sensor opgemeten en met het Kabsch-algoritme **[8-10]** wordt de transformatie van het assenstelsel van de NDI-fieldgenerator naar het assenstelsel van het schaakbord gevonden.

### Stap 3: schaakbord naar camera

Met de kalibratiemethode Calib van J.-Y. Bouguet **[7]** wordt de transformatie van het assenstelsel van het schaakbord naar het assenstelsel van de camera gevonden.

### Stap 4: EM-sensor naar camera

Stappen 1, 2 en 3 worden bij elkaar genomen en de transformatie van het assenstelsel van de EM-sensor naar het assenstelsel van de camera wordt berekend met het Kabsch-algoritme **[8-10]**.




**Figuur 1: De verschillende assenstelsels met stappenplan.**

# 3. Methode

## Voorbereiding

De data van de plaats van de EM-sensor worden in een genormaliseerd assenstelsel gezet, zodat er gebruik gemaakt kan worden van een transformatiematrix in plaats van een rotatiematrix en translatievector. Alle assenstelsels zijn in millimeters (mm) uitgedrukt, het normaliserende getal is daarom 1 (mm).

## Stap 1: EM-sensor naar NDI-fieldgenerator

De data van de EM-sensor worden in twee delen gegeven. Een plaats in het magnetisch veld uitgedrukt in een translatievector en een oriëntatie in het magnetisch veld uitgedrukt in de zxy Euler Angles, een rotatievector.

$$^{ndi}P_{EM} = \begin{bmatrix} x\ (mm) \\ y\ (mm) \\ z\ (mm) \\ 1 \end{bmatrix} \ en\ ^{ndi}EA_{EM} = \begin{bmatrix} rotZ\ (°) \\ rotX\ (°) \\ rotY\ (°) \end{bmatrix}$$

Van de rotatievector wordt een rotatiematrix gemaakt:

$$^{ndi}R_{EM} = \begin{bmatrix} \cos(rotY) & 0 & \sin(rotY) \\ 0 & 1 & 0 \\ -\sin(rotY) & 0 & \cos(rotY) \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(rotX) & -\sin(rotX) \\ 0 & \sin(rotX) & \cos(rotX) \end{bmatrix} * \begin{bmatrix} \cos(rotZ) & -\sin(rotZ) & 0 \\ \sin(rotZ) & \cos(rotZ) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

En van de rotatievector en de translatievector wordt een transformatiematrix gemaakt:

$$^{ndi}T_{EM} = \begin{bmatrix} & ^{ndi}R_{EM} & & ^{ndi}P_{EM} \\ 0 & 0 & 0 & \end{bmatrix}$$

Deze transformatiematrix beschrijft de transformatie van het assenstelsel van de EM-sensor op één moment naar het assenstelsel van de NDI-fieldgenerator.

## Stap 2: NDI-fieldgenerator naar schaakbord

Het schaakbord wordt naast de NDI-fieldgenerator geplaatst, zodat alle punten op het schaakbord meetbaar zijn. **Tijdens de kalibratie wordt het schaakbord niet meer verplaatst, zodat de vaste transformatiematrix vast blijft!** De plaatsen van de kruisingen op het schaakbord in het assenstelsel van de NDI-fieldgenerator worden gemeten met een pointer waar een EM-sensor in zit. De afmetingen van het schaakbord worden ook opgemeten. Met behulp van het Kabsch-algoritme **[8-10]** wordt de rotatiematrix en translatievector met de *least root mean square* (LRMS) van de fout gevonden.

$$^{sch}Tc_{ndi} = \begin{bmatrix} tx\ (mm) \\ ty\ (mm) \\ tz\ (mm) \end{bmatrix} \ en\ ^{sch}Rc_{ndi} = \begin{bmatrix} X(x) & X(y) & X(z) \\ Y(x) & Y(y) & Y(z) \\ Z(x) & Z(y) & Z(z) \end{bmatrix}$$

Het Kabsch-algoritme **[8-10]** maakt gebruik van puntenparen. Het is bekend welke punten uit de twee datasets bij elkaar horen. De twee punten in een puntenpaar beschrijven hetzelfde, maar in een ander assenstelsel. De transformatie tussen de twee assenstelsels wordt berekend door het Kabsch algoritme **[8-10]**. Deze rotatiematrix en translatievector kunnen in een transformatiematrix gezet worden.

$$^{sch}T_{ndi} = \begin{bmatrix} & ^{sch}Rc_{ndi} & & ^{sch}Tc_{ndi} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Deze transformatiematrix beschrijft de transformatie van het assenstelsel van de NDI-fieldgenerator naar het assenstelsel van het schaakbord.

### Stap 3: schaakbord naar camera

De camerakalibratie van Bouguet **[7]** gebruikt meerdere beelden van het schaakbord om de intrinsieke en extrinsieke waarden van de camera te bepalen. De intrinsieke waarden beschrijven het effect van de lens van de camera op de beelden. De extrinsieke waarden beschrijven de plaats en oriëntatie van de omgeving ten opzichte van de camera. De extrinsieke waarden worden uitgedrukt in translatievectoren en rotatiematrices.

$$^{cam}Tc_{sch} = \begin{bmatrix} tx\ (mm) \\ ty\ (mm) \\ tz\ (mm) \end{bmatrix} en\ ^{cam}Rc_{sch} = \begin{bmatrix} X(x) & X(y) & X(z) \\ Y(x) & Y(y) & Y(z) \\ Z(x) & Z(y) & Z(z) \end{bmatrix}$$

Van deze translatievector en rotatiematrix is een transformatiematrix gemaakt:

$$^{cam}T_{sch} = \begin{bmatrix} & ^{cam}Rc_{sch} & & ^{cam}Tc_{sch} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Deze transformatiematrix beschrijft de transformatie van het assenstelsel van het schaakbord naar het assenstelsel van de camera.

### Stap 4: EM-sensor naar camera

De drie transformatiematrices $^{ndi}T_{EM}$, $^{sch}T_{ndi}$ $en$ $^{cam}T_{sch}$ worden nu samengebracht tot één transformatiematrix:

$$^{cam}T_{EM} = {}^{cam}T_{sch} * {}^{sch}T_{ndi} * {}^{ndi}T_{EM}$$

Deze transformatiematrix beschrijft de transformatie van het assenstelsel van de EM-sensor naar het assenstelsel van de camera op een bepaald moment.

In een perfecte wereld is de transformatiematrix voor ieder meetpunt hetzelfde. Maar omdat er altijd meetfouten zijn, wordt het Kabsch-algoritme **[8-10]** gebruikt om de transformatiematrix te vinden met de LRMS van de metingen.

# 4. Uitvoering

Voor de uitvoering is een kalibratiehandleiding gemaakt. In deze handleiding staat de kalibratie van de EM-sensor met de camera stap voor stap beschreven. De Matlabscripts en -functies zijn bijgevoegd in de bijlagen.

## 4.1. Handleiding voor de kalibratie van de camera in de endoscoop met de EM-sensor aan de punt van de endoscoop.

### Benodigdheden

- Endoscoop
- 6DOF kathetrische elektromagnetische sensor (EM-sensor)
- 6DOF pointer EM-sensor
- NDI Aurora fieldgenerator met systeem
- Zwart/wit schaakbord
- Computer

### Voorbereiding

- Volg de tutorial van de camerakalibratie Matlab®-toolbox Calib voor je aan de kalibratie begint: http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html
- Instaleer de NDI-software op de computer.
- Bevestig het NDI-systeem aan de computer.
- Zet de NDI Aurora fieldgenerator neer op een plek zo ver mogelijk verwijderd van metalen voorwerpen.
- Meet de afmetingen van het schaakbord in mm.
- Leg/zet het schaakbord met het schaakbordvlak richting de rest van het EM-veld op een plek in het bereik van de NDI Aurora fieldgenerator, zoals in *figuur 2*. Contoleer dit met het programma 'NDI-track'. Als er dode plekken in het veld blijken te zijn, zet dan de NDI Aurora fieldgenerator op een andere plek.

Figuur 2: Het schaakbord bij de fieldgenerator.

- Bind de kathetrische EM-sensor op de endoscoop, zoals in *figuur 3*.
- Start Matlab® op en zet je *current directory* op de plek waar je de kalibratie wilt opslaan.
- Plaats het bestand **Kalibrati_script_blanco.m** toe aan deze map.
- Voeg in Matlab® de mappen: **EMcam_kalibratie, ndi_Matlab** en **toolbox_calib** toe aan *path:* druk met de rechter muisknop op de map en selecteer *add to path -> current folder.*

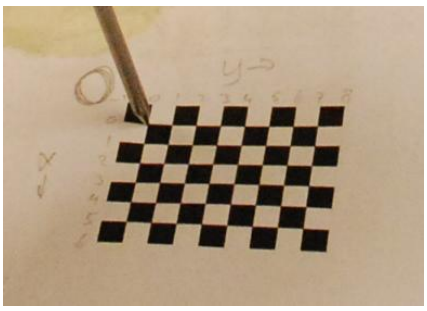

Figuur 3: De EM-sensor aan de punt van de endoscoop.

- Vul het eerste deel ***voorbereiding*** van het script **Kalibrati_script_blanco.m** in, sla het script voor eigen referentie op onder een naam gerelateerd aan de naam van je kalibratie door bijvoorbeeld 'blanco' te veranderen. Voer nu ***voorbereiding*** uit.

## Metingen

### *Het schaakbord in het EM veld*

Neem de 6DOF-pointer-EM-sensor en meet de plaats van elke hoek op het schaakbord zoals weergegeven in *figuur 4* met ***functie 1***. De volgorde wordt aangegeven door de functie in je *command window*. Het schaakbord wordt opgeslagen in *Pndi_sch.mat*. Als er een fout optreedt, omdat de meting een error geeft, wordt de meting automatisch gestopt. Met ***functie 1.1*** kun je de meting vervolgen vanaf het punt waar het fout is gegaan.



Figuur 4: 6DOF-pointer-EM-sensor op het schaakbord.

### *De camera ten opzichte van het schaakbord EN de EM-sensor in het EM veld*

Neem de endoscoop met EM-sensor. Laat de camera een opname maken met ***functie 2.1*** en laat op hetzelfde moment het NDI-systeem metingen uitvoeren van de plaats en oriëntatie van de kathetrische EM-sensor met de functie ***functie 2.2***. Start eerst de *image acquiring tool* op en maak hem klaar om op te nemen, start daarna de meting van de NDI. Deze geeft een pauze met de mededeling: '*Start de meting door op de spatiebalk te drukken*' wacht hiermee. Neem de computer in beeld zoals in *figuur 5* en start de video-opname, wacht een seconde en start de NDI-meting door in het commandwindow op de spatiebalk te drukken, terwijl de camera deze handeling opneemt. Maak een stuk of 20 tot 25 stationaire beelden van het **hele** schaakbord. Je mag meer stationaire beelden maken.

Figuur 5: Startframe bij opname

## Verwerken van data

### *Maak sch_T_ndi*

Maak met ***functie* 3** het assenstelsel van het schaakbord. Maak ${}^{sch}T_{ndi}$ met ***functie 4***. Deze functie maakt gebruik van het Kabsch-algoritme **[8-10].**

### *Kies de stationaire momenten uit*

Speel de opname van de kalibratie af met ***functie 5.1***. Loop de opnames frame voor frame door. Noteer het *begin* en het *eind* van de intervallen waarin het plaatje nagenoeg niet verandert, negeer de intervallen van minder dan 14 frames of waar het schaakbord toch niet helemaal op staat. Noteer het framenummer waarin de meting van de NDI wordt gestart (Het frame waar de spatiebalk wordt ingedrukt). Controleer ook het aantal frames per seconde (fps) door te kijken hoeveel frames tussen de eerste secondeverandering en de laatste secondeverandering zitten. Het aantal fps valt anders uit dan ingevuld is in de *image acquiring tool*!

Vul het begin en het eind van de intervallen in ***functie 5.2*** in en maak de losse plaatjes voor de camerakalibratie. Als er meer dan één opname is gemaakt om de beelden vast te leggen, volg dan de instructies in het script.

Vul het aantal opnames in bij *N*, de start van de NDI-meting voor iedere opname bij *start* en het aantal fps van iedere opname bij *fps* in ***functie 6*** in en speel de functie af om de datapunten en de bijbehorende oriëntaties te vinden die bij de frames horen.

### *Voer de camerakalibratie uit*

Voer de camerakalibratie van Bouguet **[7]** uit, zoals je in de tutorial geleerd hebt. Optimaliseer naar eigen inzicht. Als je besluit onbruikbare frames te negeren, worden deze ook genegeerd in latere scripts.

Nu hoef je enkel ***functie 7*** nog uit te voeren.

***Functie 7*** doet vijf dingen. Allereerst negeert hij de frames waar de trilling van de hand te groot was. Hij zet de rotatie- en translatiematrices die gevonden zijn in de camerakalibratie om naar transformatiematrices ${}^{cam}T_{sch}n$. Daarna maakt hij van de plaats en oriëntaties van de geselecteerde datapunten de transformatiematrices ${}^{ndi}T_{EM}n$.

Als vierde worden de transformatiematrices ${}^{cam}T_{sch}1{:}N$, ${}^{sch}T_{ndi}$ en ${}^{ndi}T_{EM}1{:}N$ gebruikt om het assenstelsel van de EM-sensor in het assenstelsel van de camera te plaatsen. Dit wordt gedaan voor ieder geselecteerd punt met een gesimuleerde eenheidsmatrix:

$$ {}^{cam}T_{sch}n * {}^{sch}T_{ndi} * {}^{ndi}T_{EM}n * \begin{bmatrix} 0 & 0.3 & 0 & 0 & -0.3 & 0 & 0 \\ 0 & 0 & 0.3 & 0 & 0 & -0.3 & 0 \\ 0 & 0 & 0 & 0.3 & 0 & 0 & -0.3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} $$

Doordat de balans van de gesimuleerde eenheidsmatrix op de oorsprong ligt, maakt het niet uit of de lengte van de eenheidsvectoren 0.3, 1 of 5 mm is. Er wordt altijd dezelfde transformatiematrix gevonden met het Kabsch-algoritme **[8-10]**. Voor een overzichtelijke weergave is er gekozen voor 0,3 mm. Daarna wordt met behulp van het Kabsch-algoritme **[8-10]** de transformatie met de LRMS van het assenstelsel van de EM-sensor naar het assenstelsel van de camera ${}^{cam}T_{EM}$ uitgerekend.

Tot slot worden de absolute en relatieve fouten van de alle metingen bepaald en opgeslagen.

### *Resultaten*

Aan het eind heb je, in de map die je aan het begin hebt aangemaakt, een aantal bestanden. Het bestand cam_T_EM.mat heeft je transformatiematrix ${}^{cam}T_{EM}$. Het bestand fouten.mat bezit de absolute en relatieve fouten van de kalibratie.

# 5. Resultaten

Na het uitvoeren van de kalibratie is de transformatiematrix ${}^{cam}T_{EM}$ gevonden. Met deze transformatiematrix is de plaats en de oriëntatie van de EM-sensor voor elk gemeten punt in het assenstelsel van de camera uit te drukken.

De kalibratie is twee keer uitgevoerd. Eén keer met een groot schaakbord van 184 mm bij 235 mm en één keer met een klein schaakbord van 28,2 mm bij 37 mm. Beide schaakborden waren 7 bij 9 vakjes groot.

De afzonderlijke transformatiematrices van ieder punt en de uiteindelijke transformatiematrix tussen de camera en de EM-sensor zijn enkel relevant voor de metingen na de kalibratie. Wat wel relevant is voor verder onderzoek, is de nauwkeurigheid van de transformatiematrices. Daarom zullen alle uitkomsten visueel weergegeven worden en de absolute fouten zoveel mogelijk uit een relatief perspectief bekeken worden.

### Stap 1: EM-sensor naar NDI-fieldgenerator

In stap 1 zijn de transformatiematrices ${}^{ndi}T_{EM}1{:}N$ gevonden. Deze transformatiematrices leggen de plek van ieder uitgekozen meetpunt vast. Deze meetpunten zijn onderdeel van een reeks meetpunten over een vastgestelde periode. Voor de meting met het grote schaakbord waren dat drie metingen van ieder 90 seconden en één meting van 35 seconden. Het bleek op het moment van de meting niet mogelijk om een video-opname van langer dan 35 seconden te maken, dus zijn van de drie metingen van 90 seconden enkel de eerste 35 seconden gebruikt. Voor de meting met het grote schaakbord zijn vier metingen van 35 seconden uitgevoerd.

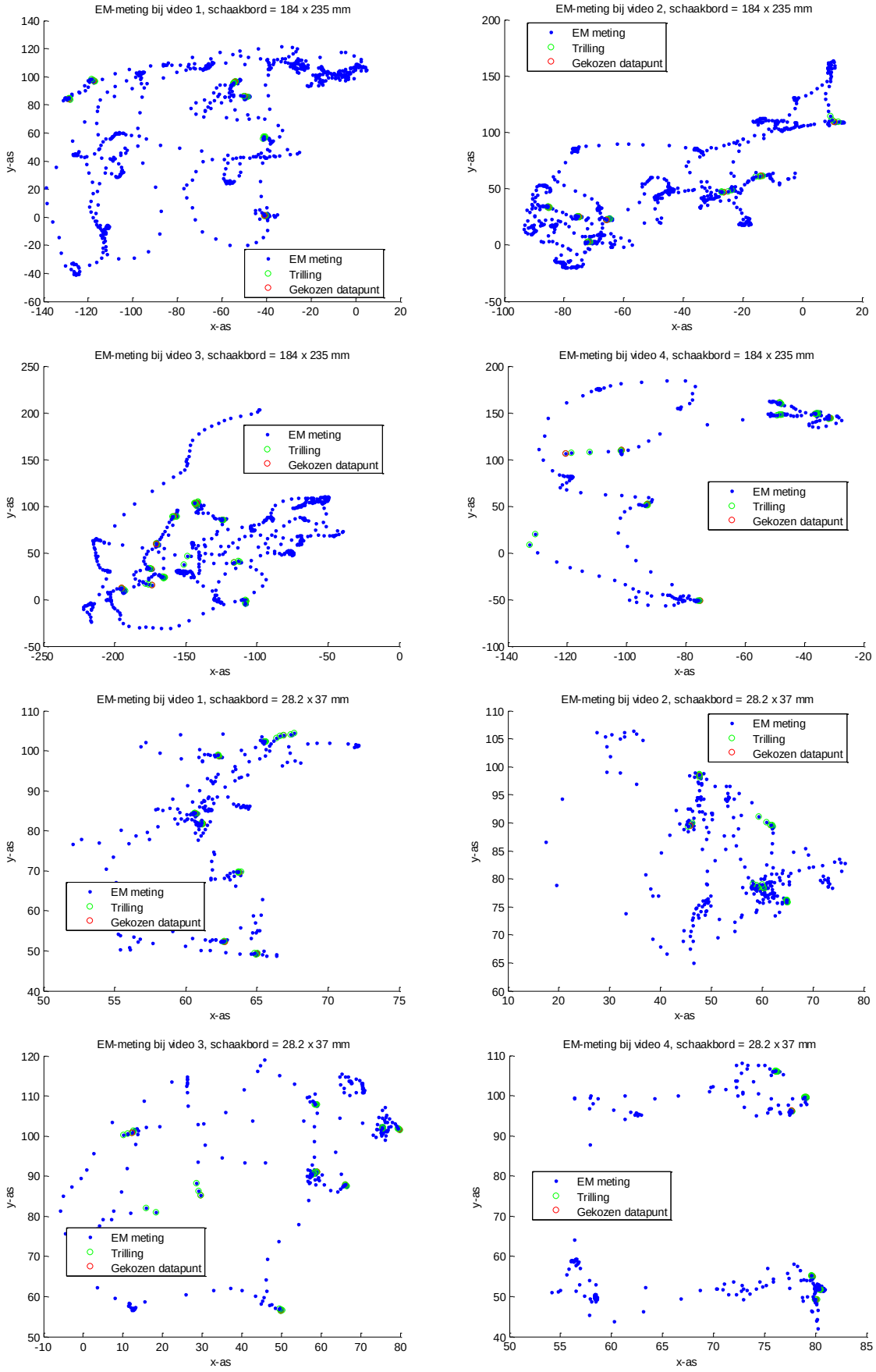
Helaas zijn de meetfouten, waarmee het NDI-systeem gemeten heeft, niet met de data meegekomen. Die kunnen daarom niet bestudeerd worden. Er kan wel gekeken worden naar de afwijking van het gekozen punt en de punten die direct voor en direct na dat punt gemeten zijn. De endoscoop is tijdens de meting met de hand vastgehouden. Omdat wij mensen door fysieke aspecten als bloedsomloop, spierspanningen en ademhaling altijd bewegen is de meting niet volledig stationair. Er is een vorm van trilling. Deze trilling is een vorm van verplaatsing. Deze verplaatsing is te berekenen met de plaats van de gemeten punten:

$$verplaatsing = \sqrt{\big(x(n) - x(n+1)\big)^2 + \big(y(n) - y(n+1)\big)^2 + \big(z(n) - z(n+1)\big)^2}$$

Voor de fout wordt de verplaatsing tussen de vier punten direct om de gekozen punten en de gekozen punten bekeken, dus vijf punten of vier verplaatsingen per gekozen punt.

In de programmering is ook een opdracht ingebouwd dat punten die een te grote trilling hebben, worden genegeerd, dus die zullen ook hier genegeerd worden. Voor deze metingen is een grens van 10 mm gebruikt.

In *figuur 6* zijn de metingen van de EM-sensor te zien.



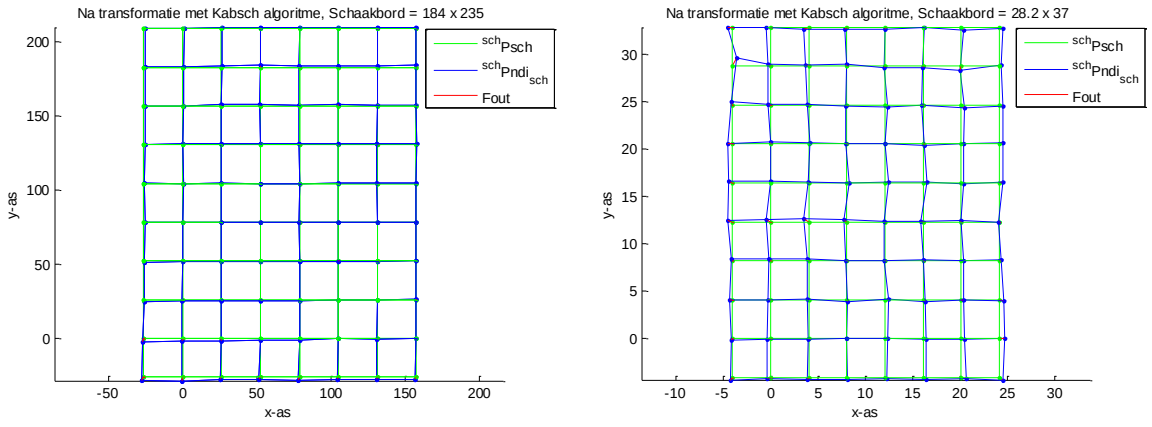

Figuur 6: EM-metingen

De RMS van de fout in de transformatiematrices ${}^{ndi}T_{EM}1{:}N$ door de trilling voor het grote schaakbord is 1,01 mm, en voor het kleine schaakbord 0,24 mm. Het verschil kan komen door het feit dat de endoscoop, om het hele schaakbord in beeld te krijgen, verder van het grote schaakbord gehouden moet worden dan van het kleine schaakbord. Er is daardoor minder ruimte om houvast aan de tafel te vinden tijdens het vasthouden. Als de absolute fouten gerelativeerd worden met de grootste afstand tussen de gekozen datapunten zijn de fouten $\frac{1{,}01}{263} * 100\% = 0{,}38\%$ voor het grote schaakbord en $\frac{0{,}24}{72.3} * 100\% = 0{,}34\%$ voor het kleine schaakbord.

In *figuur 6, video 4 van schaakbord 284 x 235 mm,* is te zien, dat er in sommige datasets datapunten gekozen zijn, die niet in de stationaire gedeeltes van de datasets liggen. (Dit was voor dat de grens van trilling aangegeven was.) Dit kan komen door het afwijkende aantal fps van de opname. Het afwijkende aantal fps geeft afwijkende tijdstippen voor ieder frame. En daardoor worden sommige frames niet aan het bijbehorende datapunt gekoppeld, maar aan een ander datapunt.

## Stap 2: NDI-fieldgenerator naar schaakbord

In stap twee is de transformatiematrix ${}^{sch}T_{ndi}$ gevonden. Dit is gevonden door alle kruispunten op het schaakbord in de ruimte van de NDI-fieldgenerator te meten. Het Kabsch-algoritme **[8-10]** heeft daarna de transformatie met de kleinste RMS uitgerekend. De RMS van de fout in de transformatiematrix van de metingen van het grote schaakbord is 1.13 mm, die van het kleine schaakbord is 0.40 mm. Als we dit relativeren naar de totale lengte van het schaakbord worden de fouten $\frac{1{,}13}{\sqrt{235^2+184^2}} * 100\% = 0.38\%$ voor het grote schaakbord en $\frac{0{,}40}{\sqrt{37^2+28{,}4^2}} * 100\% = 0{,}85\%$ voor het kleine schaakbord. Ook goed om te vermelden is dat de grootte van de fout van het NDI-systeem 0,48 mm is voor de RMS voor een 6D-EM-sensor **[11]**, dus de absolute fout van het kleine schaakbord valt binnen de technische specificaties van het meetsysteem. In *figuur 7* is het resultaat te zien van de transformatie van de meetpunten van het schaakbord in het assenstelsel van de NDI-fieldgenerator naar het assenstelsel van het schaakbord.



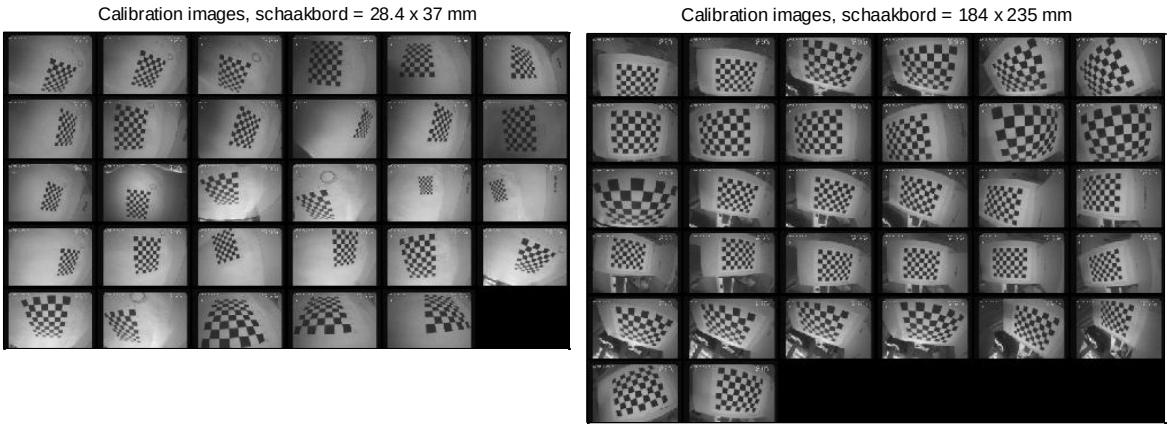

**Figuur 7: Het resultaat na de transformatie ${}^{sch}T_{ndi}$ gevonden met behulp van het Kabsch-algoritme [8-10]**

## Stap 3: schaakbord naar camera

In stap drie zijn de transformatiematrices ${}^{cam}T_{sch}1{:}N$ gemaakt. Dit is gedaan door de camerakalibratie van Bouguet **[7]**. De camerakalibratie geeft niet de fout van de rotatiematrix, maar wel de fout van de translatievector weer in de resultaten. Deze fout is opgesplitst in de fout in x-, y- en z-richting. Allereerst wordt deze opgesplitste fout samengevoegd met Pytagoras, waarna de RMS van alle

fouten genomen wordt. De RMS van de fout van de translatie in de transformatiematrices ${}^{cam}T_{sch}1{:}N$ na de camerakalibratie voor het grote schaakbord is 1.34 mm en die van het kleine schaakbord is 0.45 mm. Dit verschil kan komen door het feit dat de afstand tussen camera en schaakbord bij het grote schaakbord groter is dan bij het kleine schaakbord, om het hele schaakbord in beeld te krijgen.

Als er gerelativeerd wordt met de grootste afstand tussen de gekozen punten wordt het percentage van de fout voor het grote schaakbord $\frac{1{,}34}{263} = 0{,}51\ \%$ en voor het grote schaakbord $\frac{0{,}45}{72.3} * \ 100\% = \ 0{,}62\%$. Dit sluit aan op de uitleg van de absolute fout. Hoewel de afstand van de camera tot het grote schaakbord groter is dan tot het kleine schaakbord, is de relatieve fout nagenoeg gelijk. Het verschil in de relatieve fout kan komen doordat in de beelden van het grote schaakbord het schaakbord meer van de plaatjes beslaat dan in de beelden van het kleine schaakbord. Dit is in *figuur 8* gevisualiseerd.




**Figuur 8: De beelden die gebruikt zijn bij de camerakalibratie.**

Bij de relativering van de absolute fout is dezelfde grootste afstand gebruikt als bij de EM meting. Omdat de translatievectoren van de camerakalibratie steeds een x-, y- en z-richting in een ander assenstelsel beschrijven, het assenstelsel van de camera op dat moment, zijn ze niet bruikbaar om de afstand tussen de plaatsen te bepalen. Dus is hier aangenomen dat dezelfde grootste afstand te gebruiken is om de relatieve fout te bepalen.

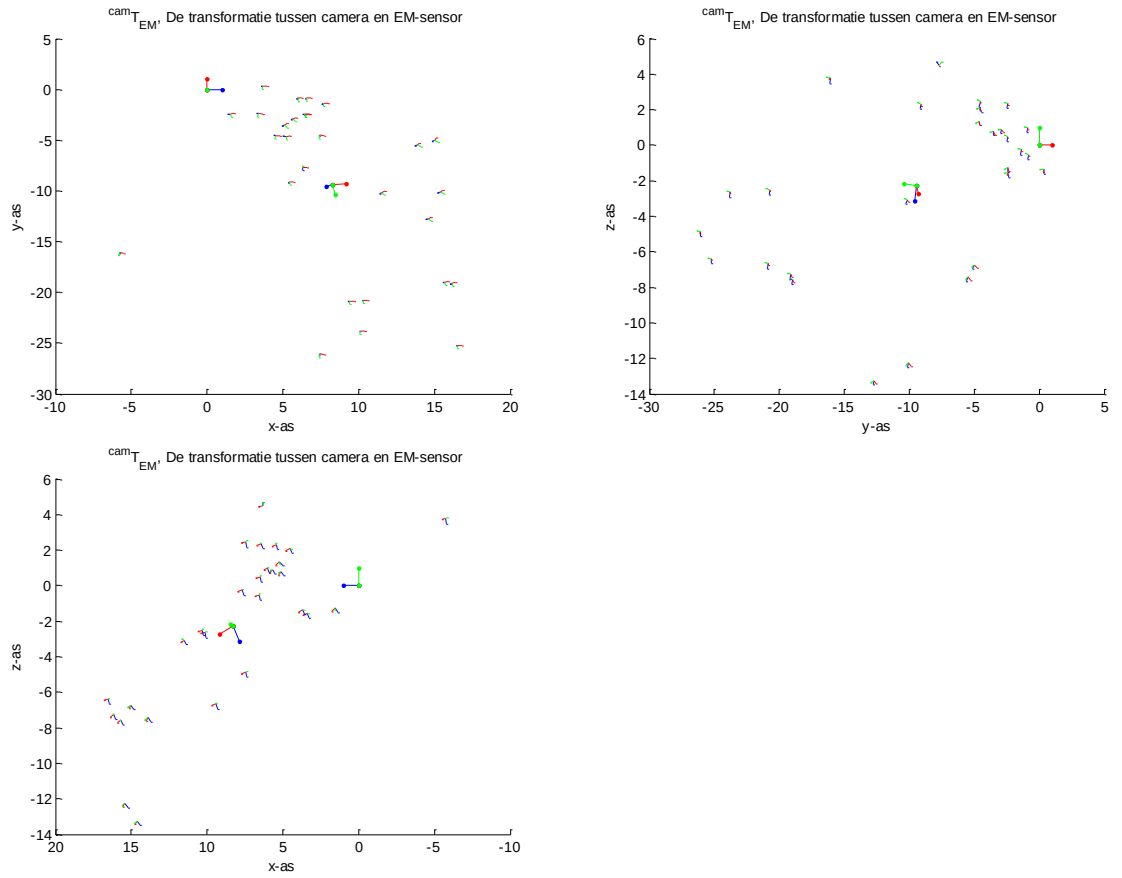
### Stap 4: EM-sensor naar camera

Aan het einde worden de transformatiematrices ${}^{cam}T_{sch}1{:}N$, ${}^{sch}T_{ndi}$ en ${}^{ndi}T_{EM}1{:}N$ bij elkaar genomen en gemaakt tot de transformatiematrix ${}^{cam}T_{EM}1{:}N$. Omdat ze allemaal hetzelfde beschrijven zouden ze allemaal hetzelfde moeten zijn, maar dit is niet zo, omdat er fouten in het systeem zitten. Daarom is met het Kabsch-algoritme **[8-10]** en met gesimuleerde eenheidsassenstelsels een transformatie-matrix gevonden met de kleinste RMS. Voor de kalibratie met het grote schaakbord is dit:

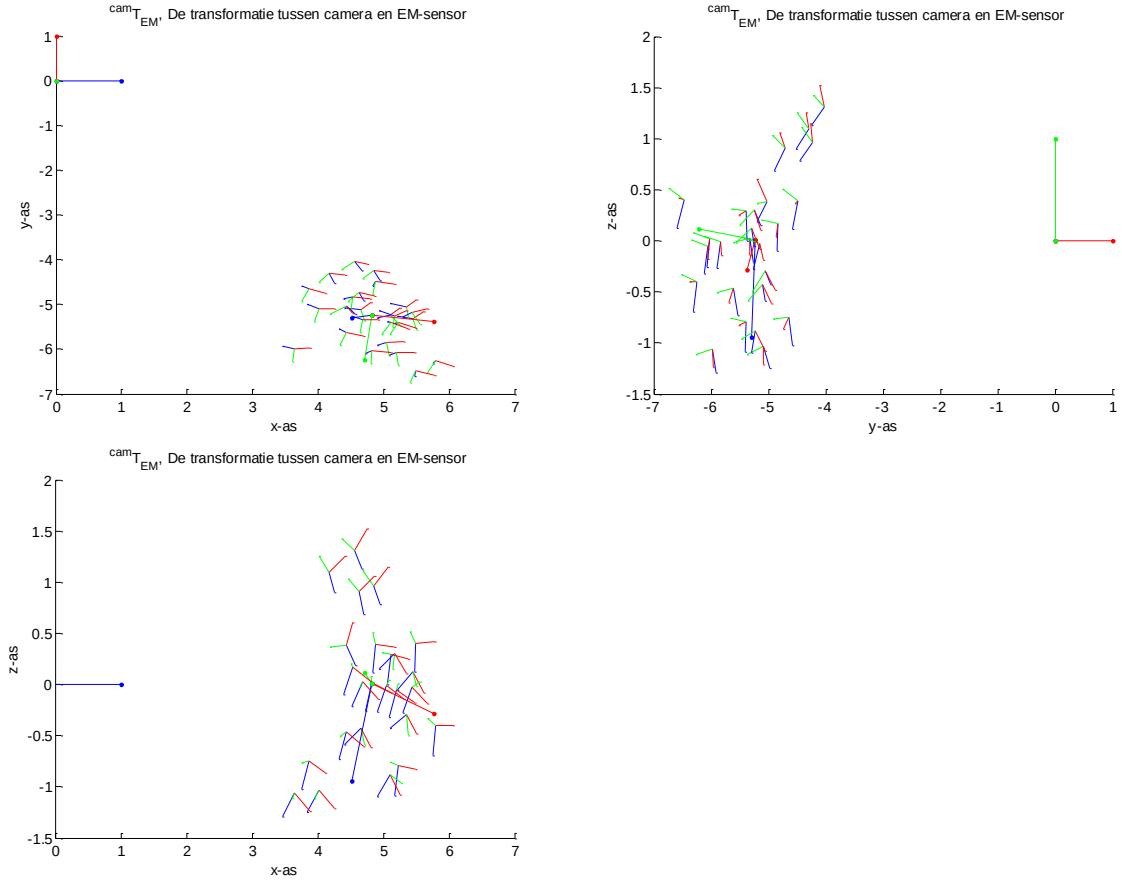
$$ {}^{cam}T_{EM} = \begin{bmatrix} -0.44 & 0.88 & 0.18 & 8.29 \\ -0.16 & 0.12 & -0.98 & -9.41 \\ -0.88 & -0.46 & 0.09 & -2.27 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Voor de kalibratie met het kleine schaakbord is dit:

$$ {}^{cam}T_{EM} = \begin{bmatrix} -0.31 & 0.95 & -0.11 & 4.82 \\ -0.07 & -0.14 & -0.99 & -5.24 \\ -0.95 & -0.29 & 0.10 & 0.01 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$




Figuur 9: Het assenstelsel van de EM-sensor in het assenstelsel van de camera van het grote schaakbord




Figuur 10: Het assenstelsel van de EM-sensor in het assenstelsel van de camera van het kleine schaakbord

In *figuren 9* en *10* is te zien dat de fout bij het kleine schaakbord een stuk kleiner is dan de fout bij het grote schaakbord. De absolute fout van de transformatiematrix ${}^{cam}T_{EM}$ bij het grote schaakbord is 10,8 mm, de absolute fout van de transformatiematrix ${}^{cam}T_{EM}$ bij het kleine schaakbord is 1,05 mm. Als we deze fouten relativeren met de uitgerekende translatie tussen camera en EM-sensor is de relatieve fout bij het grote schaakbord $\frac{10,8}{12,7} * 100\% = 84,5\%$ en de relatieve fout bij het kleine schaakbord $\frac{1,05}{7,12} * 100\% = 14,2\%$

Voor het gemak worden de fouten onder elkaar gezet in *tabel 1*.

**Tabel 1: De absolute en relatieve fouten op een rij**

| | Groot schaakbord 235 x 184 | | Klein schaakbord 37 x 28,2 | |
|---|---|---|---|---|
| | absoluut | relatief | absoluut | relatief |
| Fout ${}^{ndi}T_{EM}$ | 1,01 mm | 0.38 % | 0,24 mm | 0,34 % |
| Fout ${}^{sch}T_{ndi}$ | 1,13 mm | 0.38 % | 0,40 mm | 0,85 % |
| Fout ${}^{cam}T_{sch}$ | 1,34 mm | 0.51 % | 0,45 mm | 0,62 % |
| Fout ${}^{cam}T_{EM}$ | 10,8 mm | 84.5 % | 1,05 mm | 14,2 % |

Opvallend is dat de uiteindelijke fout veel groter is dan de afzonderlijke fouten op zich. De som van de afzonderlijke absolute fouten bij het kleine schaakbord vormen bij elkaar opgeteld de absolute uiteindelijke fout. Dit is niet te zeggen van de absolute fouten bij het grote schaakbord.

De relatieve fout van ${}^{cam}T_{EM}$ is erg groot. Dit komt doordat bij deze fout de absolute fout groter is dan de fouten van de afzonderlijke transformatiematrices en er over een veel kleinere afstand gerelativeerd wordt dan bij de afzonderlijke transformatiematrices.

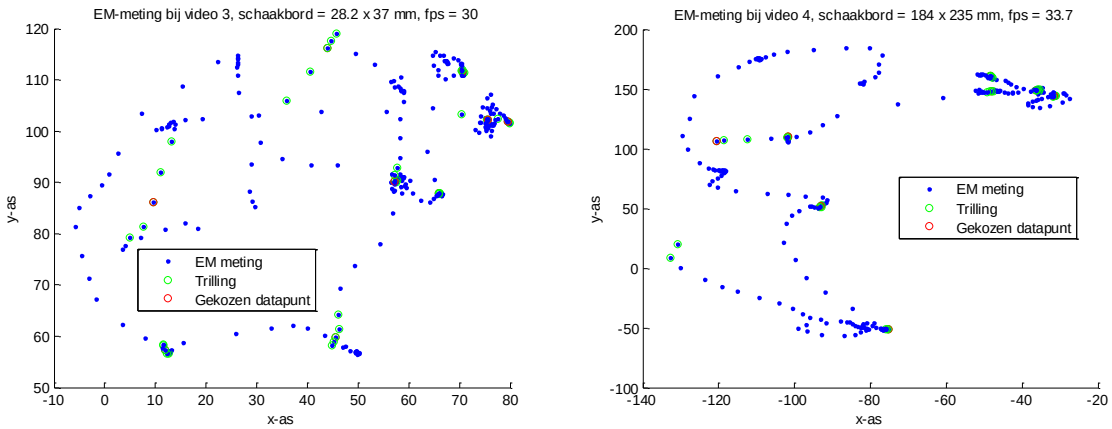
# 6. Conclusie en Discussie

Door de kleine datapool zullen nog geen concrete conclusies getrokken kunnen worden over de bruikbaarheid van de kalibratie, maar de gewonnen data geeft wel een goede indicatie van de complexiteit van het proces. Bovendien laat het zien, dat het erg belangrijk is dat de metingen zeer nauwkeurig uitgevoerd worden.

Omdat er gezocht wordt naar een kleine, maar erg belangrijke transformatie, zijn de relatief kleine fouten van de metingen toch niet zo klein als ze in eerste instantie lijken. Een fout van 0,40 mm op een schaakbord van 37 x 28,2 mm of een fout van 1,13 mm op een schaakbord van 235 x 184 mm lijkt klein, maar op de punt van de endoscoop met een diameter van 10 mm wordt dit al een stuk groter. Als daar nog eens twee fouten in dezelfde orde bovenop komen, is de nauwkeurigheid ver te zoeken.

## Synchronisatie

Er is één fout die nog niet besproken is, maar misschien nog meer effect heeft dan de fouten door trilling, misplaatsing van de pointer of een pixellig beeld. Namelijk de synchronisatie van de twee systemen. De camera neemt beelden op met een bepaald aantal frames per seconde. Tegelijk meet het NDI-systeem de plaats en oriëntatie van de EM-sensor. Maar dit gebeurt niet exact tegelijk. Het NDI-systeem heeft een andere meetfrequentie, dus de meting wordt op een ander moment gedaan dan het beeld wordt opgenomen. Dit is opgelost door de camera voor een tijd stil te houden, zodat de metingen die niet op exact hetzelfde moment gedaan zijn, wel hetzelfde resultaat hebben, met de trilfout natuurlijk. Maar… Het bleek dat het aantal frames per seconden dat opgegeven was in de *imqtool* niet het aantal frames per seconden was waarmee werd opgenomen. In *figuur 11* is goed te zien wat daarvan het effect was.




**Figuur 11: Voorbeelden van fouten door verkeerde synchronisatie**

De gekozen datapunten kwamen in de gedeeltes van de meting te liggen waar de camera verplaatst werd. Bij sommige metingen, zoals video 4 van de kalibratie met het grote schaakbord, is dit nog te zien, ondanks de pogingen het eigenlijke aantal frames per seconden te bepalen.

Om dit te verbeteren zal er met meer zekerheid vastgesteld moeten kunnen worden dat de meting van de EM-sensor en het beeld echt bij elkaar horen. Voor de kalibratie kan de endoscoop met EM-sensor aan een statief bevestigd worden. De camera en sensor bewegen dan niet. Het beeld en de meting van de EM-sensor kunnen dan vlak achter elkaar opgenomen worden. Een belangrijk detail hierin is dat het statief niet van metaal mag zijn. Metaal verandert het magnetisch veld en maakt daardoor de meting onbetrouwbaar. Zo'n statief was niet voor handen op het moment van de

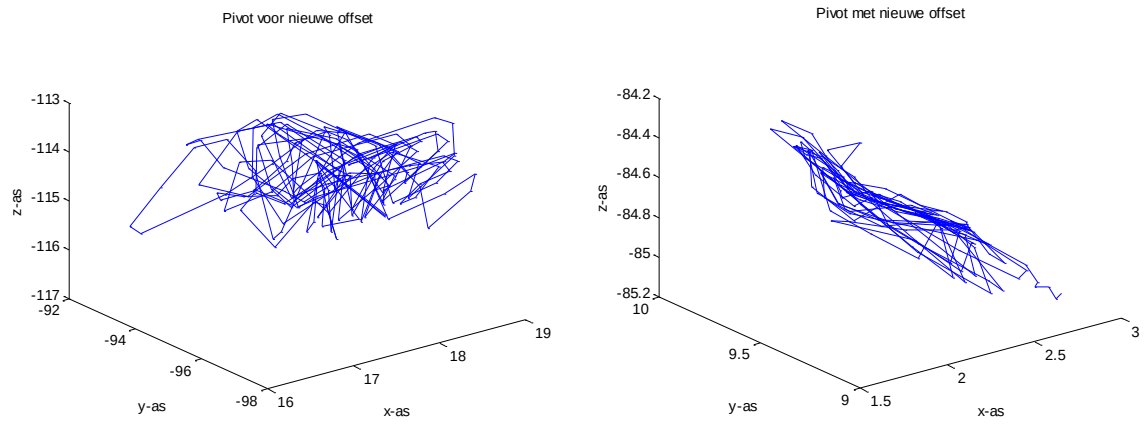
metingen. Deze methode kan ook nog een hoop tijd besparen in het proces, omdat er dan niet meer in de opnames gezocht hoeft te worden naar de stationaire beelden, dit koste namelijk anderhalf uur per kalibratie.

Maar zelfs als de kalibratie dan lukt, moet er nog steeds een betere synchronisatie van de twee meetsystemen plaatsvinden, om het model van N. van der Stap et al. **[6]** te valideren.

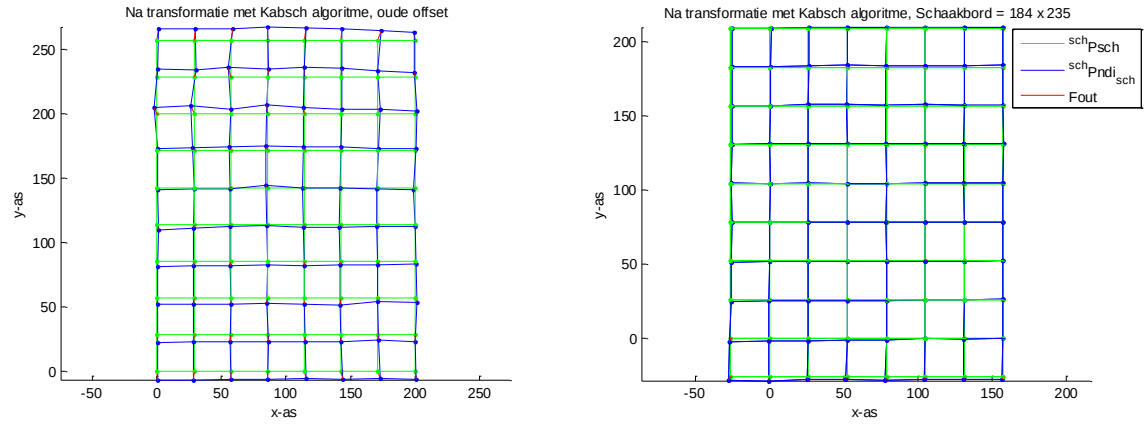
### Offset

Om de plaats van het schaakbord in het assenstelsel van de NDI-fieldgenerator te meten is een pointer EM-sensor gebruikt. Deze pointer EM-sensor geeft de plaats en de oriëntatie van zijn EM-sensor aan het systeem. De EM-sensor in de pointer zit echter niet in de punt, want dat is niet mogelijk, hij zit er dichtbij. De plaats van de EM-sensor ten opzichte van de punt van de pointer heet de offset. De offset was al bekend voor dit onderzoek begon. Deze bleek echter niet te kloppen. Daarom is de offset opnieuw bepaald met de *pivoting tool* van het programma *NDI Track*. De pointer is in een gaatje geplaatst waar de pointer vrij in kon kantelen, maar waar de punt op dezelfde plek bleef. Vervolgens is de *pivoting tool* van het programma *NDI track* gebruikt om de offset te bepalen. De pointer is in een conische ruimte met een hoek in de punt groter dan 30° gedurende 20 seconden bewogen. en het programma heeft de offset berekend. De offset is net iets groter dan de fout van het NDI-systeem, daarom is dit 20 keer gedaan.

Zoals bij een relatief grote fout te verwachten is, waren er grote verschillen in de resultaten, maar na verloop van tijd ontstond toch een patroon. Bij kleine fouten was de offset ongeveer $x = 0.2,\ y = -0.5,\ z = 0.6$. De oude offset was $x = -0.4,\ y = 1.3, z = 2.5$. Om de effecten van deze aanpassing uit te beelden zijn in *figuur 12* de metingen te zien van een meting met dezelfde procedure als de procedure om de offset te bepalen. En in *figuur 13* zijn de metingen van een schaakbord weergegeven voor en na het toepassen van de nieuwe offset.




**Figuur 12: Het verschil tussen de oude en de nieuwe offset, gevisualiseerd met de handelingen van de *pivotingtool***




Figuur 13: Het verschil tussen de oude en de nieuwe offset, gevisualiseerd met het schaakbord

De lengte van de assen die in *figuur 12* te zien zijn, kun je als de omvang van de fout zien. De omvang van de fout is van 5,0 bij 6,0 bij 4,0 mm teruggebracht naar 1,5 bij 1,0 bij 1,0 mm; dit is een verkleining van iets meer dan 4 keer. Het verschil in tussen de twee schaakborden is in *figuur 13* ook goed te zien. In het linker plaatje neemt de fout toe met de afstand tot het midden van het schaakbord. De fout neemt toe vanuit het midden, omdat het Kabsch-algoritme **[8-10]** eerst de translatie tussen de twee evenwichtspunten, in dit geval het midden van het schaakbord, uitrekent, voor de rotatiematrix uitgerekend wordt. Zo wordt een toenemende fout groter waargenomen aan de extremen dan rond het evenwicht van de meting. De fout neemt toe doordat, menselijk als we zijn, de pointer schuiner wordt vastgehouden hoe verder er gerekt wordt met de armen. Als de aangenomen offset niet klopt met de daadwerkelijke offset, ontstaat er een meetafwijking afhankelijk van hoe de pointer vastgehouden wordt. Bovendien zal bij een foute aanname van de offset de transformatie tussen schaakbord en NDI-fieldgenerator niet kloppen met wat de transformatie hoort te zijn. Een paar millimeter of graden is al genoeg om een grote fout te geven. De LRMS van de transformatie voor het schaakbord aan de linker kant was 5,46 mm. Het schaakbord was 200 x 257 mm, dit was een fout van $\frac{5{,}46}{\sqrt{200^2+257^2}} * 100\% = 1.68$ %. De fout van 0,38% is iets meer dan 4 keer kleiner.

## Bolletjes patroon

T.R.F. van Steenbergen et al. heeft de camerakalibratie weten te automatiseren door gebruik te maken van een bolletjespatroon in plaats van een schaakbordpatroon **[12]**. Deze methode vindt het midden van de bollen in het plaatje in plaats van de plek van de kruispunten op het schaakbord. Door een uniek patroon in het midden van het bollejesbord kan de methode uitrekenen waar de oorsprong van het bolletjesbord ligt en hoeft die niet meer handmatig aangeklikt te worden om de kalibratie uit te voeren. De methode met het bolletjespatroon verliest wel wat van de nauwkeurigheid van de camerakalibratie van Bouguet **[7]**. Maar Steenbergen et al. geven aan dat de winst in tijd zwaarder weegt dan het verlies in nauwkeurigheid.

Een mens zal met de pointer EM-sensor de kruispunten van het schaakbord nauwkeuriger aangeven dan het midden van de bolletjes. Dus als het bolletjespatroon gebruikt gaat worden, moet er een keuze gemaakt worden. De extra fout aannemen of een nauwkeurigere methode maken om de plaats van het bolletjespatroon in het assenstelsel van de NDI-fieldgenerator te meten.

Het is zeker zo dat het tijd scheelt als de kalibratie na de metingen volledig geautomatiseerd wordt. Als tijdens de meting de frames van de camera en datapunten van de EM-sensor al gekoppeld zijn en

de camerakalibratie geautomatiseerd is met het bolletjespatroon, kan na de meting de kalibratie bij wijze van spreken met één druk op knop uitgevoerd worden.

## 6.1. Aanbeveling

Helaas is door de kleine datapool nog niet definitief aan te geven of deze methode van kalibratie werkt of niet. De resultaten van de kalibratie met het kleine schaakbord zijn echter hoopvol. Allereerst is het belangrijk om zoveel mogelijk synchronisatie- en trilfouten weg te halen. De trilling is weg te halen door de endoscoop met een statief van hout of kunststof vast te houden in plaats van met de handen. De synchronisatie is te verbeteren, door foto's te maken van het schaakbord en daarnaast één meting van de plaats van de EM-sensor per foto, in plaats van de video-opname en de EM tracking die hierboven gebruikt is. Nadat deze verbeteringen doorgevoerd zijn, moet de foutmarge opnieuw bekeken worden om te bepalen of dit een goede methode is om de kalibratie uit te voeren.

Wanneer deze verbeteringen doorgevoerd zijn en de foutmarge blijkt klein te zijn, kan overwogen worden om de camerakalibratie te automatiseren met behulp van het bolletjespatroon van Steenbergen et al. **[12]**. De methode heeft een iets grotere foutmarge dan de oorspronkelijke camerakalibratie van Bouguet **[7]**.

# 7. Referenties


1. Schippers, E.I., *Kamerbrief invoering bevolkingsonderzoek darmkanker*, W.e.S.V. ministerie van Volksgezondheid, Editor 2011.
2. Wieren S van (RIVM), G.H.R., *Aantal verwachte extra coloscopieën per provincie 2014-2020*, 2013, Nationale Atlas Volksgezondheid.: Volksgezondheid Toekomst Verkenning.
3. Lee, S.H., et al., *An adequate level of training for technical competence in screening and diagnostic colonoscopy: a prospective multicenter evaluation of the learning curve.* Gastrointestinal Endoscopy, 2008. **67**(4): p. 683-689.
4. Spier, B.J., et al., *Colonoscopy training in gastroenterology fellowships: determining competence.* Gastrointestinal Endoscopy, 2010. **71**(2): p. 319-324.
5. Tassios, P.S., et al., *Acquisition of competence in colonoscopy: The learning curve of trainees.* Endoscopy, 1999. **31**(9): p. 702-706.
6. van der Stap, N., et al., *The Use of the Focus of Expansion for Automated Steering of Flexible Endoscopes*, in *2012 4th Ieee Ras & Embs International Conference on Biomedical Robotics and Biomechatronics*, J.P. Desai, L.P.S. Jay, and L. Zollo, Editors. 2012. p. 13-18.
7. Bouguet, J.-Y. *Camera Calibration Toolbox for Matlab*. last updated 2010 29-08-2013]; Available from: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.
8. Kabsch, W., *SOLUTION FOR BEST ROTATION TO RELATE 2 SETS OF VECTORS.* Acta Crystallographica Section A, 1976. **32**(SEP1): p. 922-923.
9. Kabsch, W., *DISCUSSION OF SOLUTION FOR BEST ROTATION TO RELATE 2 SETS OF VECTORS.* Acta Crystallographica Section A, 1978. **34**(SEP): p. 827-828.
10. Kavraki, L. *Molecular Distance Measures*. 2007 29-08-2013]; Available from: http://cnx.org/content/m11608/1.23/.
11. Inc., N.D. *Aurora Accuracy Performance - Planar Field Generator*. 2013 [cited 2013 01-09-2013]; Available from: http://www.ndigital.com/medical/aurora-techspecs.php.
12. T.R.F. van Steenbergen, F.v.d.H., T.J.M. Ruersz, K.F.D. Kuhlmann, and J. Nijkamp, *Laparoscopic Camera Calibration for Surgery using Electromagnetic Tracking.* 2013.

# Bijlagen

## Matlabscripts

### *Kalibratie_script_blanco.m*

```
%% Voorbereiding
clear, clc, close all
sdir = 'Kalibratie\Kalibratie\';% De plek waar alles opgeslagen gaat worden
mkdir(sdir); % De map wordt hier gemaakt
mkdir([sdir, 'KalibratieFrame']); % In deze folder worden de losse frames
opgeslagen
com  = 'com'; % De naam van de COM-poort waar het NDIsysteem is aangesloten
x = ;  y = ;  % Het aantal vakjes van het schaakbord in de x- en y-richting
X = ; Y = ; % De totale lengte van het schaakbord in mm in de x- en y-richting

%% functie 1: Schaakbord in het EM veld
Pndi_sch = maak_Pndi_sch(x,y,com,sdir);

%% functie 1.1: Bij het optreden van een fout
A=1; % Het meetpunt waar de fout is opgetreden, weergegeven in het
     % Command Window
Pndi_sch = maak_Pndi_sch_fout(x,y,com,sdir,Pndi_sch,A);

%% functie 2.1: Tijdens meet de plek van de katetrische EM-sensor: Video-opname
imaqtool

%% functie 2.2: Tijdens video-opname: Meet de plek van de katetrische EM-sensor.
time = 35; % Het aantal seconden dat de meting gaat duren
N = 1;     % !!!Als je vaker deze handeling wilt doen, verander dit nummer!!!
[Pndi_endo, Qndi_endo, EAndi_endo, t, nt] = maak_Pndi_endo(time, com, sdir, N);

%% functie 3: Schaakbord
Psch = maak_Psch(x,y,X,Y,sdir);

%% functie 4: Transformatiematrix sch_T_ndi
[sch_T_ndi, lrms] = maak_sch_T_ndi(Pndi_sch,Psch, sdir);

%% functie 5.1: Selecteer de stationaire beelden.
video = [sdir, 'video_endo_0001.avi'];
implay(video)

%% functie 5.2: Maak plaatjes van de stationaire beelden
N = 1;
video = [sdir, 'video_endo_0001.avi'];
% Noteer hieronder de nummers van de frames van het begin en het eind van
% de stationaire momenten in video N.
begin = ;
eind  = ;
[fnr, M] = maak_frames(video, begin, eind, N, 1, sdir);

%%
% Als de meting in meer dan één video gedaan moest worden.
N = 2;
video = [sdir, 'video_endo_0002.avi'];
% Noteer hieronder de nummers van de frames van het begin en het eind van
% de stationaire momenten in video N.
begin = ;
eind =  ;
[fnr, M] = maak_frames(video, begin, eind, N, M, sdir);

%%
% Als de meting in meer dan twee videos gedaan moest worden.
% Copiëer hier het vorige stuk commands, verander N en de naam van de video.
% Vul weer begin en het eind in en run het stuk.
% herhaal zo vaak als nodig.
```

```
%% functie 6: Vind de datapunten die bij de stationaire frames horen
N = ; % Het aantal opnames wat nodig was om de frames op te nemen
start = []; % Het framenummers wanneer de NDI metingen gestart zijn van iedere
opname.
fps = ; % Het aantal frames per seconde van iedere opname
[Pndi_endo_dpt, Qndi_endo_dpt, EAndi_endo_dpt, dpt] = ...
    maak_datapunt(start, fps, N, X, Y, sdir);

%% voer de cameracalibratie uit met de Camera Calibration Toolbox.

%% Backups ophalen (eventueel)
% voer de voorbereiding nogmaals uit
load([sdir, 'Pndi_sch.mat']);
load([sdir, 'Psch']);
load([sdir, 'sch_T_ndi.mat']);
load([sdir, 'Pndi_endo_dpt.mat']);
load([sdir, 'KalibratieFrame\Calib_Results.mat']);

%% functie 7: maak de transformatiematricen sch_T_cam, ndi_T_EM en cam_T_EM
grens = ; % mm. maximale trilgrote bij het meten van een datapunt.
active_images(((find(trilling_dpt>grens))-mod((find(trilling_dpt>grens)),4))/4)=0;
maak_cam_T_sch_en_ndi_T_EM
maak_cam_T_EM_script
bepaal_fouten
```

### *maak_Pndi_sch.m*

```
function Pndi_sch = maak_Pndi_sch(x,y,com,sdir)
% de functie maak_Pndi_sch meet het schaakbord in het assenstelsel van de
% fieldgenerator.
% input:  x        = het aantal vakjes op het schaakbord in de x-richting
%         y        = het aantal vakjes op het schaakbord in de y-richting
%         com      = de naam van de com-poort waar het NDIsysteem op is
%                    aangesloten
%         sdir     = De plek waar Pndi_sch wordt opgeslagen
% output  Pndi_sch = Het schaakbord in het assenstelsel van de
%                    fieldgenerator verkregen door de metingen
%
% Wanneer er wordt gezegt: 'Meet meetpunt x=... en y=...' zet je pointer op
% dat punt en druk op de spatiebalk.
%
NDI=ndiopen(com);
aantal=(x+1)*(y+1);
Pndi_sch=zeros(4,aantal);
Pndi_sch(4,:)=1;
for a=1:aantal
    [X,Y]=ind2sub([x+1,y+1],a);
    disp(['Meet meetpunt x=', num2str(X-2), ' en y=' num2str(Y-2)])
    pause
    [T,Q,EA,F,err]=ndiread(NDI,1);
    if err==1
        break
    end
    Pndi_sch(1:3,a)=T;
end
if err==1
    disp(['Meting beëindigd: fout in meting nr. ', num2str(a)])
    Pndi_sch=Pndi_sch(:,1:a-1);
    plot3(Pndi_sch(1,:),Pndi_sch(2,:),Pndi_sch(3,:))
    save([sdir, 'Pndi_sch.mat'],'Pndi_sch')
else
    save([sdir, 'Pndi_sch.mat'],'Pndi_sch')
    plot3(Pndi_sch(1,:),Pndi_sch(2,:),Pndi_sch(3,:))
end
xlabel('x-as')
ylabel('y-as')
zlabel('z-as')
title('Pndi_sch')
```

```
ndiclose(NDI)
end
```

### *maak_Pndi_sch_fout.m*

```
function Pndi_sch = maak_Pndi_sch_fout(x,y,com,sdir,Pndi_sch,A)
% De functie maak_Pndi_sch_fout meet verder waar de functie maak_Pndi_sch.m
% of de functie maak_Pndi_sch_fout gestopt is.
% input:  x        = Het aantal vakjes op het schaakbord in de x-richting
%         y        = Het aantal vakjes op het schaakbord in de y-richting
%         com      = De naam van de com-poort waar het NDIsysteem op is
%                    aangesloten
%         sdir     = De plek waar Pndi_sch wordt opgeslagen
%         Pndi_sch = De onvolledige Pndi_sch
%         A        = Het meetpunt waar de meting fout ging
% output  Pndi_sch = Het schaakbord in het assenstelsel van de
%                    fieldgenerator verkregen door de metingen
%
% Wanneer er wordt gezegt: 'Meet meetpunt x=... en y=...' zet je pointer op
% dat punt en druk op de spatiebalk.
%
NDI=ndiopen(com);
aantal=(x+1)*(y+1);
Pndi(:,A:aantal)=zeros(4,aantal-A);
Pndi(4,A:aantal)=1;
for a=A:aantal
    [X,Y]=ind2sub([x,y],a);
    disp(['Meet meetpunt x=', num2str(X), ' en y=' num2str(Y)])
    pause
    [T,Q,EA,F,err]=ndiread(NDI,1);
    if err==1
        break
    end
    Pndi(1:3,a)=T;
end
if err==1
    disp(['Meting beëindigd: fout in meting nr. ', num2str(a)])
    Pndi=Pndi(:,1:a-1);
    plot3(Pndi(1,:),Pndi(2,:),Pndi(3,:))
    save([sdir, 'Pndi_sch.mat'],'Pndi')
else
    save([sdir, 'Pndi_sch.mat'],'Pndi')
    plot3(Pndi(1,:),Pndi(2,:),Pndi(3,:))
end
xlabel('x-as')
ylabel('y-as')
zlabel('z-as')
title('Pndi_sch')
ndiclose(NDI)
end
```

### *maak_Pndi_endo.m*

```
function [Pndi_endo, Qndi_endo, EAndi_endo, t, nt] =...
          maak_Pndi_endo(time, com, sdir, N)
% De functie maak_Pndi_endo meet de plek van de EM-sensor op de tip van de
% endoscoop voor een bepaalde tijd.
% input:  time       = De tijd dat er gemeten gaat worden in seconden
%         com        = De naam van de COM-poort waar het NDI-systeem is
%                      aangesloten
%         sdir       = De plek waar de output wordt opgeslagen
%         N          = Het nummer waarmee de bestandsnaam moet eindigen in
%                      het geval dat er meerdere metingen gedaan worden.
% output: Pndi_endo  = De plaats van de EM-sensor
%         Qndi_endo  = De richting van de EM-sensor in quaternions
%         EAndi_endo = De richting van de EM-sensor in Euler Angles
%         t          = De tijdstippen van de metingen in seconden
```

```
%         nt          = De tijdstippen dat er een fout op is getreden in de
%                       meting
NDI=ndiopen(com);
t=[]; nt=[]; ti=0;
Pndi_endo=[];
Qndi_endo=[];
EAndi_endo=[];
disp('Start meting door de spatiebalk aan te slaan.')
pause
disp('Go!')
tic
while ti<time
    [T,Q,EA,F,err]=ndiread_endo(NDI,1);
    ti=toc;
    if err==0
        t=[t,ti];
        Pndi_endo=[Pndi_endo,T];
        Qndi_endo=[Qndi_endo,Q];
        EAndi_endo=[EAndi_endo,EA];
    elseif err==1
        nt=[nt,ti];
    end
end
Pndi_endo(4,:)=1;
save([sdir, 'Pndi_endo_', num2str(N), '.mat'],'Pndi_endo', 'Qndi_endo',
'EAndi_endo', 't', 'nt')

ndiclose(NDI)
```

### *maak_Psch.m*

```
function Psch=maak_Psch(x,y,X,Y,sdir)
% De functie maak_Psch maakt de referentie Psch voor de transformatiematrix
% sch_T_ndi.
% input:  x     is het aantal vakjes in de x-richting
%         y     is het aantal vakjes in de y-richting
%         X     is de totale lengte van het schaakbord in de x-richting in mm
%         Y     is de totale lengte van het schaakbord in de y-richting in mm
%         sdir  is de (al aangemaakte) map waar Psch wordt opgeslagen
% output: Psch  is de matrix met de posities van de punten op het schaakbord
%               in het assenstelsel van het schaakbord in mm
Psch=[combvec(-1:x-1,-1:y-1);zeros(1,(x+1)*(y+1));ones(1,(x+1)*(y+1))];
Psch(1,:)=Psch(1,:).*(X/7);
Psch(2,:)=Psch(2,:).*(Y/9);
save([sdir,'Psch.mat'],'Psch')
end
```

### *maak_sch_T_ndi.m*

```
function [sch_T_ndi,lrms1]=maak_sch_T_ndi(x,y,Pndi_sch,Psch,sdir)
% Deze functie maakt de transformatiematrix voor de transformatie van het
% assenstelsel van de fieldgenerator naar het assenstelsel van het
% schaakbord.
% input:  Pndi_sch  = Het schaakbord in het assenstelsel van de
%                     fieldgenerator
%         Psch      = Het schaakbord in het assenstelsel van het schaakbord
%         sdir      = De plaats waar de output wordt opgeslagen
% output: sch_T_ndi = De transformatiematrix
%         lrms1     = De least root mean square van de fout na
%                     transformatie
[U,r,lrms1] = Kabsch(Pndi_sch(1:3,:), Psch(1:3,:));
U(4,:)=0; r(4)=1; U(:,4)=r;
sch_T_ndi = U;
save([sdir, 'sch_T_ndi.mat'], 'sch_T_ndi', 'lrms1')

% Laat de resultaten zien
sch_Pndi_sch = sch_T_ndi * Pndi_sch;
```

```
figure(1), hold on, axis equal
plot3(Psch(1,:),Psch(2,:),Psch(3,:),'.g')
    for a=0:y
        h(1)=plot3(Psch(1,(1+a*(x+1)):((x+1)+a*(x+1))),...
            Psch(2,(1+a*(x+1)):((x+1)+a*(x+1))),...
            Psch(3,(1+a*(x+1)):((x+1)+a*(x+1))),'-g');
    end
    for a=0:x
        plot3(Psch(1,[1:(x+1):((x+1)*(y+1))]+a),...
            Psch(2,[1:(x+1):((x+1)*(y+1))]+a),...
            Psch(3,[1:(x+1):((x+1)*(y+1))]+a),'-g')
    end
plot3(sch_Pndi_sch(1,:),sch_Pndi_sch(2,:),sch_Pndi_sch(3,:),'.b')
    for a=0:y
        h(2)=plot3(sch_Pndi_sch(1,(1+a*(x+1)):((x+1)+a*(x+1))),...
            sch_Pndi_sch(2,(1+a*(x+1)):((x+1)+a*(x+1))),...
            sch_Pndi_sch(3,(1+a*(x+1)):((x+1)+a*(x+1))),'-b');
    end
    for a=0:x
        plot3(sch_Pndi_sch(1,[1:(x+1):((x+1)*(y+1))]+a),...
            sch_Pndi_sch(2,[1:(x+1):((x+1)*(y+1))]+a),...
            sch_Pndi_sch(3,[1:(x+1):((x+1)*(y+1))]+a),'-b')
    end
    for a=1:((x+1)*(y+1))
        h(3)=plot3([sch_Pndi_sch(1,a),Psch(1,a)],...
            [sch_Pndi_sch(2,a),Psch(2,a)],...
            [sch_Pndi_sch(3,a),Psch(3,a)],'-r');
    end
    legend(h, '^s^c^hPsch', '^s^c^hPndi_s_c_h', 'Fout', 'Location', 'BestOutside')
xlabel('x-as')
ylabel('y-as')
zlabel('z-as')
end
```

### *Kabsch.m*

```
% Find the Least Root Mean Square distance
% between two sets of N points in D dimensions
% and the rigid transformation (i.e. translation and rotation)
% to employ in order to bring one set that close to the other,
% Using the Kabsch (1976) algorithm.
% Note that the points are paired, i.e. we know which point in one set
% should be compared to a given point in the other set.
%
% References:
% 1) Kabsch W. A solution for the best rotation to relate two sets of vectors. Acta
Cryst A 1976;32:9223.
% 2) Kabsch W. A discussion of the solution for the best rotation to relate two
sets of vectors. Acta Cryst A 1978;34:8278.
% 3) http://cnx.org/content/m11608/latest/
% 4) http://en.wikipedia.org/wiki/Kabsch_algorithm
%
% We slightly generalize, allowing weights given to the points.
% Those weights are determined a priori and do not depend on the distances.
%
% We work in the convention that points are column vectors;
% some use the convention where they are row vectors instead.
%
% Input  variables:
%  P : a D*N matrix where P(a,i) is the a-th coordinate of the i-th point
%      in the 1st representation
%  Q : a D*N matrix where Q(a,i) is the a-th coordinate of the i-th point
%      in the 2nd representation
%  m : (Optional) a row vector of length N giving the weights, i.e. m(i) is
%      the weight to be assigned to the deviation of the i-th point.
%      If not supplied, we take by default the unweighted (or equal weighted)
%      m(i) = 1/N.
%      The weights do not have to be normalized;
%      we divide by the sum to ensure sum_{i=1}^N m(i) = 1.
```

```
%      The weights must be non-negative with at least one positive entry.
% Output variables:
%  U : a proper orthogonal D*D matrix, representing the rotation
%  r : a D-dimensional column vector, representing the translation
%  lrms: the Least Root Mean Square
%
% Details:
% If p_i, q_i are the i-th point (as a D-dimensional column vector)
% in the two representations, i.e. p_i = P(:,i) etc., and for
% p_i' = U p_i + r      (' does not stand for transpose!)
% we have p_i' ~ q_i, that is,
% lrms = sqrt(sum_{i=1}^N m(i) (p_i' - q_i)^2)
% is the minimal rms when going over the possible U and r.
% (assuming the weights are already normalized).
%
function[U, r, lrms] = Kabsch(P, Q, m)
    sz1 = size(P) ;
    sz2 = size(Q) ;
    if (length(sz1) ~= 2 || length(sz2) ~= 2)
        error 'P and Q must be matrices' ;
    end
    if (any(sz1 ~= sz2))
        error 'P and Q must be of same size' ;
    end
    D = sz1(1) ;          % dimension of space
    N = sz1(2) ;          % number of points
    if (nargin >= 3)
        if (~isvector(m) || any(size(m) ~= [1 N]))
            error 'm must be a row vector of length N' ;
        end
        if (any(m < 0))
            error 'm must have non-negative entries' ;
        end
        msum = sum(m) ;
        if (msum == 0)
            error 'm must contain some positive entry' ;
        end
        m = m / msum ;      % normalize so that weights sum to 1
    else                    % m not supplied - use default
        m = ones(1,N)/N ;
    end

    p0 = P*m' ;           % the centroid of P
    q0 = Q*m' ;           % the centroid of Q
    v1 = ones(1,N) ;      % row vector of N ones
    P = P - p0*v1 ;       % translating P to center the origin
    Q = Q - q0*v1 ;       % translating Q to center the origin

    % C is a covariance matrix of the coordinates
    % C = P*diag(m)*Q'
    % but this is inefficient, involving an N*N matrix, while typically D << N.
    % so we use another way to compute Pdm = P*diag(m)
    Pdm = zeros(D,N) ;
    for i=1:N
        Pdm(:,i) = m(i)*P(:,i) ;
    end
    C = Pdm*Q' ;
%   C = P*Q' / N ;        % (for the non-weighted case)
    [V,S,W] = svd(C) ;    % singular value decomposition
    I = eye(D) ;
    if (det(V*W') < 0)    % more numerically stable than using (det(C) < 0)
        I(D,D) = -1 ;
    end
    U = W*I*V' ;

    r = q0 - U*p0 ;

    Diff = U*P - Q ;      % P, Q already centered
```

```
%   lrms = sqrt(sum(sum(Diff.*Diff))/N) ; % (for the non-weighted case)
    lrms = 0 ;
    for i=1:N
        lrms = lrms + m(i)*Diff(:,i)'*Diff(:,i) ;
    end
    lrms = sqrt(lrms) ;
end
```

### *maak_frames.m*

```
function [fnr, M] = maak_frames(video, begin, eind, N, M, sdir)
% maak_frames slaat de middelste frames van de stationaire intervallen op
% in losse .tif bestanden op
% input:  video = de naam van de plaats waar de opname opgeslagen is.
%         begin = de nummers van de frames van het begin van de stationaire
%                 momenten in de opname
%         eind  = de nummers van de frames van het eind van de stationaire
%                 momenten in de opname
%         N     = de N-de video, dit nummer komt achter het bestand te
%                 staan waar de output wordt opgeslagen.
%         M     = De extensie van het eerste plaatje wat gemaakt wordt.
%         sdir  = De plaats waar de plaatjes worden opgeslagen.
% output: fnr   = De nummers van de frames die omgezet zijn tot frames
%         M     = Het nummer van de eerstvolgende extensie uit de volgende
%                 opname.

N=num2str(N);
fnr=begin+round((eind-begin)./2);
save([sdir, 'fnr_', N], 'begin', 'eind', 'fnr')
if N==1
    mkdir([sdir, 'Kalibratieframe'])
end
obj=VideoReader(video);
a=1;
stop = M + length(fnr) - 1;
while M <= stop
    im=read(obj,fnr(a));
    imwrite(im,[sdir, 'KalibratieFrame\Frame_', num2str(M), '.tif'])
    M = M+1; a = a+1;
end
end
```

### *maak_datapunt.m*

```
function [Pndi_endo_dpt, Qndi_endo_dpt, EAndi_endo_dpt, dpt, trilling_dpt] = ...
    maak_datapunt(start, fps, N, X, Y, sdir)
% Maak_datapunt haalt de datapunten van de EM-sensor die samenvallen met de
% gekozen frames uit de meting van de EM-sensor
% input:  start          = Dit is een rij met de frames waar de meting van
%                          de EM-sensor start
%         fps            = Dit is een rij met het aantal frames per seconde
%                          waarmee de opnames zijn gemaakt
%         N              = Dit is het aantal opnames wat nodig was om de
%                          benodigde frames te verzamelen
%         sdir           = Dit is de map waar alle data wordt opgeslagen
% output: Pndi_endo_dpt  = Dit zijn de plaatsen van de EM-sensor op de
%                          momenten van de uitgekozen frames
%         Qndi_endo_dpt  = Dit zijn de oriëntaties van de EM-sensor op de
%                          momenten van de uitgekozen frames in quaternions
%         EAndi_endo_dpt = Dit zijn de oriëntaties van de EM-sensor op de
%                          momenten van de uitgekozen frames in Euler
%                          Angles
% Ook wordt de backups selectie_endo_n.mat aangevuld.

Pndi_endo_dpt = [];
Qndi_endo_dpt = [];
```

```
EAndi_endo_dpt = [];
dpt=[];
trilling_dpt=[];
for n=1:N
    A = num2str(n);
    load([sdir, 'Pndi_endo_', A, '.mat'])
    load([sdir, 'fnr_', A, '.mat'])
    t_ndi = t + (start(n)/fps(n));
    t_fnr = fnr./fps(n);
    dpt_n = zeros(1, length(t_fnr));
    dpt_n_tril = zeros(1, length(t_fnr).*5);
    for a=1:length(t_fnr)
        t_ndi_fnr = abs(t_ndi - t_fnr(a));
        [minumum, ind] = min(t_ndi_fnr);
        dpt_n(a) = ind;
        dpt_n_tril((a*5-4):(a*5)) = ind-2:ind+2;
    end
    Pndi_endo_dpt  = [Pndi_endo_dpt,  Pndi_endo(:,dpt_n) ];
    Qndi_endo_dpt  = [Qndi_endo_dpt,  Qndi_endo(:,dpt_n) ];
    EAndi_endo_dpt = [EAndi_endo_dpt, EAndi_endo(:,dpt_n)];
    dpt = [dpt, dpt_n];

    Pndi_endo_dpt_tril_n = Pndi_endo(:,dpt_n_tril);
    A = Pndi_endo_dpt_tril_n;
    trilling_dpt_n = zeros(1, length(t_fnr).*4);
    for a=1:length(t_fnr) % dit is waarschijnlijk niet efficiënt...
    trilling_dpt_n([a*4-3:a*4]) = [sqrt(sum((A(1:3,a*5-4)-A(1:3,a*5-3)).^2)), ...
                                   sqrt(sum((A(1:3,a*5-3)-A(1:3,a*5-2)).^2)), ...
                                   sqrt(sum((A(1:3,a*5-2)-A(1:3,a*5-1)).^2)), ...
                                   sqrt(sum((A(1:3,a*5-1)-A(1:3,a*5  )).^2))];
    end
    trilling_dpt = [trilling_dpt, trilling_dpt_n];


    figure(n), clf, hold on
    plot3(Pndi_endo(1,20:end), Pndi_endo(2,20:end), Pndi_endo(3,20:end), '.b')
    plot3(Pndi_endo(1,dpt_n_tril), Pndi_endo(2,dpt_n_tril),
Pndi_endo(3,dpt_n_tril), 'og')
    plot3(Pndi_endo(1,dpt_n), Pndi_endo(2,dpt_n), Pndi_endo(3,dpt_n), 'or')
    legend('EM meting', 'Trilling', 'Gekozen datapunt', 'Location', 'Best')
    xlabel('x-as'), ylabel('y-as'), zlabel('z-as')
    title(['EM-meting bij video ', num2str(n), ', schaakbord = ',...
       num2str(X,3), ' x ', num2str(Y,3), ' mm'])
end
save([sdir, 'Pndi_endo_dpt.mat'], 'Pndi_endo_dpt', 'Qndi_endo_dpt',
'EAndi_endo_dpt', 'dpt', 'trilling_dpt')
end
```

### *maak_cam_T_sch_en_ndi_T_EM.m*

```
% Dit script maakt de twee transformatie matrices sch_T_cam en ndi_T_EM
% voor ieder frame/datapunt. Omdat de calib Camera Calibration van ieder
% frame apart een translatie- en rotatievector in een variable opslaat is
% hier gekozen voor een script in plaats van een functie. De benodigde
% input moet dus in de Workspace staan. De output wordt opgeslagen in
% de workspace.
% Input:  Rc_[1:n]        = De rotatievectoren van de camera ten opzichte
%                           van het schaakbord. Het cijfer staat voor de
%                           nummer van de frame.
%         Tc_[1:n]        = De translatievectoren van de camera ten
%                           opzichte van het schaakbord Het cijfer staat
%                           voor de nummer van de frame.
%         P_ndi_endo_dpt  = De plaats van de EM-sensor op de momenten van
%                           de datapunten.
%         EA_ndi_endo_dpt = De oriëntatie in Euler Angles van de EM-sensor
%                           op de momenten van de datapunten.
% Output: cam_T_sch_[1:n] = De transformatiematricen voor de transformaties
%                           van het assenstelsel van het schaakbord naar
%                           het assenstelsel van de camera voor alle frames
```

```
%         ndi_T_EM_[1:n]  = De transformatiematricen voor de transformaties
%                           van het assenstelsel van de EM-sensor naar het
%                           assenstelsel van de fieldgenerator voor alle
%                           datapunten.


% maak de transformatiematrices cam_T_sch
for a = 1:length(image_numbers)
    eval(['Rc=Rc_', num2str(image_numbers(a)), ';']);
    eval(['Tc=Tc_', num2str(image_numbers(a)), ';']);

    cam_T_sch = [Rc, Tc];
    cam_T_sch(4,:) = 0; cam_T_sch(4,4)=1;

    eval(['cam_T_sch_', num2str(image_numbers(a)), ' = cam_T_sch;']);
end

% maak transformatiematrices ndi_T_EM
    rotZ = @(A) [cos(A.*(pi/180)), -sin(A.*(pi/180)), 0; sin(A.*(pi/180)),
cos(A.*(pi/180)), 0; 0, 0, 1];
    rotY = @(B) [cos(B.*(pi/180)), 0, sin(B.*(pi/180)); 0, 1, 0; -sin(B.*(pi/180)),
0, cos(B.*(pi/180))];
    rotX = @(C) [1, 0, 0; 0, cos(C.*(pi/180)), -sin(C.*(pi/180)); 0,
sin(C.*(pi/180)), cos(C.*(pi/180))];
for a = 1:length(Pndi_endo_dpt)
    Rc = rotY(EAndi_endo_dpt(3,a)) * rotX(EAndi_endo_dpt(2,a)) *...
        rotZ(EAndi_endo_dpt(1,a)); Rc(4,:) = 0;
    Tc = Pndi_endo_dpt(:,a);
    ndi_T_EM = [Rc,Tc];
    eval(['ndi_T_EM_', num2str(a), ' = ndi_T_EM;']);
end
```

### *maak_cam_T_EM_script.m*

```
% Dit script maakt de transformatiematrix cam_T_EM. Omdat er te veel
% inputvariablen zijn, is gekozen voor een script met de input in de
% workspace.
% Input:  cam_T_sch_[1:n] = De transformatiematrices van het assenstelsel
%                           van het schaakbord naar het assenstelsel van de
%                           camera voor ieder frame.
%         ndi_T_EM_[1:n]  = De transformatiematrices van het assenstelsel
%                           van de EM-sensor naar het assenstelsel van de
%                           fieldgenerator voor ieder datapunt.
%         sdir            = De map waar de output wordt opgeslagen.
% Output: cam_T_EM        = De transformatiematrix van het assenstelsel van
%                           de EM-sensor naar het assenstelsel van de
%                           camera met de kleinste rms bepaald met het
%                           Kabsch-algoritme
%         lrms2           = De Root Mean Square van de transformatiematrix
%                           cam_T_EM tov de input.

P = [0 0.3 0 0 -0.3 0 0
    0 0 0.3 0 0 -0.3 0
    0 0 0 0.3 0 0 -0.3
    1 1 1 1 1 1 1];
cam_PEM=[];
for a=1:length(image_numbers)
    if active_images(a) == 1
        eval(['cam_T_sch = cam_T_sch_', num2str(a), ';'])
        eval(['ndi_T_EM = ndi_T_EM_', num2str(a), ';'])
        cam_P_EM_a =  cam_T_sch * sch_T_ndi * ndi_T_EM * P;
        cam_PEM = [cam_PEM, cam_P_EM_a];
    end
end
cam_Pcam = zeros(size(cam_PEM));
for a=0:sum(active_images==1)-1
    cam_Pcam(:,[1:7]+a.*7) = P ;
```

```
end

% plot alle afzondelijke getransformeerde eenheidsassenstelsels
figure(1), hold on
for a=0:sum(active_images==1)-1
    plot3([cam_PEM(1,a*7+1), cam_PEM(1,a*7+2)] , [cam_PEM(2,a*7+1),
cam_PEM(2,a*7+2)]...
        , [cam_PEM(3,a*7+1), cam_PEM(3,a*7+2)], '-b')
    plot3([cam_PEM(1,a*7+1), cam_PEM(1,a*7+3)] , [cam_PEM(2,a*7+1),
cam_PEM(2,a*7+3)]...
        , [cam_PEM(3,a*7+1), cam_PEM(3,a*7+3)], '-r')
    plot3([cam_PEM(1,a*7+1), cam_PEM(1,a*7+4)] , [cam_PEM(2,a*7+1),
cam_PEM(2,a*7+4)]...
        , [cam_PEM(3,a*7+1), cam_PEM(3,a*7+4)], '-g')
end

[cam_T_EM, lrms2] = maak_cam_T_EM(cam_Pcam, cam_PEM, sdir);
P = [0 1 0 0
    0 0 1 0
    0 0 0 1
    1 1 1 1];
cam_Pcam=P;
cam_PEM=cam_T_EM*P;
% plot de uiteindelijke transformaties.
figure(1), hold on
plot3([cam_Pcam(1,1), cam_Pcam(1,2)] , [cam_Pcam(2,1), cam_Pcam(2,2)]...
    , [cam_Pcam(3,1), cam_Pcam(3,2)], '.-b')
plot3([cam_Pcam(1,1), cam_Pcam(1,3)] , [cam_Pcam(2,1), cam_Pcam(2,3)]...
    , [cam_Pcam(3,1), cam_Pcam(3,3)], '.-r')
plot3([cam_Pcam(1,1), cam_Pcam(1,4)] , [cam_Pcam(2,1), cam_Pcam(2,4)]...
    , [cam_Pcam(3,1), cam_Pcam(3,4)], '.-g')

plot3([cam_PEM(1,1), cam_PEM(1,2)] , [cam_PEM(2,1), cam_PEM(2,2)]...
    , [cam_PEM(3,1), cam_PEM(3,2)], '.-b')
plot3([cam_PEM(1,1), cam_PEM(1,3)] , [cam_PEM(2,1), cam_PEM(2,3)]...
    , [cam_PEM(3,1), cam_PEM(3,3)], '.-r')
plot3([cam_PEM(1,1), cam_PEM(1,4)] , [cam_PEM(2,1), cam_PEM(2,4)]...
    , [cam_PEM(3,1), cam_PEM(3,4)], '.-g')

title('^c^a^mT_E_M, De transformatie tussen camera en EM-sensor')
xlabel('x-as')
ylabel('y-as')
zlabel('z-as')
```

### *maak_cam_T_EM.m*

```
function [cam_T_EM,lrms2]=maak_cam_T_EM(EM_Pcam,EM_PEM,sdir)
% Deze functie maakt de transformatiematrix voor de transformatie van het
% assenstelsel van de EM-sensor naar het assenstelsel van de camera.
% input:  EM_Pcam   = De camera in het assenstelsel van de EM-sensor met
%                     bijbehorende punten voor de x- y- en z-richtingen. De
%                     lengte van deze matrix is 7 keer het aantal bekeken
%                     frames
%         EM_PEM    = De EM-sensor met bijbehorende punten voor de x- y- en
%                     z-richtingen De lengte van deze matrix is 7 keer het
%                     aantal bekeken frames
%         sdir      = De plaats waar de output wordt opgeslagen
% output: cam_T_EM  = De transformatiematrix
%         lrms2     = De least root mean square van de fout na
%                     transformatie
[U,r,lrms2] = Kabsch(EM_Pcam(1:3,:), EM_PEM(1:3,:));
U(4,:)=0; r(4)=1; U(:,4)=r;
cam_T_EM = U;
save([sdir, 'cam_T_EM.mat'], 'cam_T_EM', 'lrms2')
end
```

### *bepaal_fouten.m*

```
% Bepaal_fouten
% Dit script bepaald de root mean square van de fouten die zijn opgetreden
% en van de uiteindelijke fout en slaat deze op in fouten.mat
% input:  De hele workspace.
% output: fout_rms_sch_T_ndi  =
%                               de rms van de fout bij het bepalen van de
%                               transformatiematrix van het assenstelsel
%                               van de NDI Fieldgenerator naar het
%                               assenstelsel van het schaakbord.
%         fout_rms_cam_Tc_sch =
%                               de rms van de fout van de translatie tussen
%                               het assenstelsel van het schaakbord en het
%                               assenstelsel van de camera. De fout in de
%                               rotatie is niet gegeven door de
%                               camerakalibratie!!!
%         fout_rms_ndi_Tc_EM_trilling =
%                               De rms van de translatie tussen 5 punten
%                               rond de genomen punten uit de EM meting.
%                               Dit kan gezien worden als de fout
%                               veroorzaakt door een trillende hand.
%         fout_rms_cam_T_EM =
%                               De rms van de uiteindelijke fout bepaald
%                               door het Kabsch algoritme

fout_rms_sch_T_ndi = lrms1;
perc_fout_sch_T_ndi = (fout_rms_sch_T_ndi / ((X.^2+Y.^2).^0.5)) *100; %

B=[]; D=[];
for a=1:length(active_images)
    if active_images(a)==1
        eval(['A=Tc_error_', num2str(a), ';'])
        B=[B, (A(1).^2 + A(2).^2 + A(3).^2).^0.5];
        eval(['C=Tc_' num2str(a),';'])
        D=[D, (C(1).^2 + C(2).^2 + C(3).^2).^0.5];
    end
end
fout_rms_cam_Tc_sch = sqrt(mean(B.^2));

A=[]; C=[];
for a=1:length(active_images)
    if active_images(a)==1
        A=[A,trilling_dpt(a*4-3:a*4)];
        C=[C,Pndi_endo_dpt(:,a)];
    end
end
fout_rms_ndi_Tc_EM_trilling = sqrt(mean(A.^2));
[~,A]=size(C);
B=combvec(1:A,1:A);
maxdPndi_endo_dpt=max(( ((C(1,B(1,:)) - C(1,B(2,:))).^2) + ...
                        ((C(2,B(1,:)) - C(2,B(2,:))).^2) + ...
                        ((C(3,B(1,:)) - C(3,B(2,:))).^2)).^0.5);
perc_fout_ndi_Tc_EM_trilling = ...
    fout_rms_ndi_Tc_EM_trilling / maxdPndi_endo_dpt * 100; %
perc_fout_cam_Tc_sch = fout_rms_cam_Tc_sch / maxdPndi_endo_dpt * 100; %

fout_rms_cam_T_EM = lrms2;
Tc_cam_EM = (cam_T_EM(1,4).^2 + cam_T_EM(2,4).^2 + cam_T_EM(3,4).^2).^0.5;
perc_fout_cam_T_EM = fout_rms_cam_T_EM / Tc_cam_EM * 100; %

save([sdir, 'fouten.mat'], 'fout_rms_sch_T_ndi', 'perc_fout_sch_T_ndi', ...
    'fout_rms_cam_Tc_sch', 'perc_fout_cam_Tc_sch',...
    'fout_rms_ndi_Tc_EM_trilling', 'perc_fout_ndi_Tc_EM_trilling',...
    'fout_rms_cam_T_EM', 'perc_fout_cam_T_EM')
```