— Report P&O CW 2019–2020 Task F1.1 —

Department of Computer Science – KU Leuven 28 april 2020

Team 12

Martijn Debeuf
Frédéric Blondeel
Toon Sauvillers
Dirk Vanbeveren
Bert Van den Bosch
Seppe Van Steenbergen

Inhoudsopgave

1 Inleiding

In de huidige applicatie van de *Screencaster* wordt een afbeelding of video weergegeven volgens het perspectief van het master apparaat. Deze wordt vastgelegd tijdens het detecteren van de schermen. Dit verslag bespreekt de mogelijkheid om de beweging van het master apparaat tegenover de schermen te volgen en dus ook het perspectief ten opzichte van de schermen . Dit wil zeggen dat de gebruiker in *real-time* met het master apparaat kan rondlopen en de schermen hier meteen op reageren door het gewijzigde perspectief weer te geven. Zo zal de gebruiker ten alle tijden afbeeldingen of video's vanuit zijn perspectief te zien krijgen.
Er worden twee grote aspecten behandeld, enerzijds kan de data van de sensoren in het master apparaat gebruikt worden (lees sectie 2). Anderzijds kan de camera gebruikt worden. Hierbij kunnen keypoints gezocht worden binnen de frames. Daarna kan op basis hiervan de translatie van het master apparaat afgeleid worden (lees sectie 3). Vervolgens kan de applicatie met behulp van deze twee aspecten een volledige transformatie opstellen om een correct perspectief te behouden voor de gebruiker.

2 Sensoren

Eén manier om beweging van een apparaat te detecteren is met behulp van zijn sensoren. Vandaag de dag zitten smartphones vol met sensoren die allerlei verschillende soorten data verzamelen [1]. Diegene die nuttig kunnen zijn voor het bepalen van de beweging zijn de accelerometer, gyroscoop en magnetometer.

De accelerometer meet de acceleratie die het apparaat ondervindt. Hiermee kan deze sensor de oriëntatie bepalen van het apparaat aan de hand van de zwaartekracht of de beweging in eender welke lineaire richting.

De gyroscoop geeft ook informatie over de oriëntatie van het apparaat maar is nauwkeuriger dan de accelerometer. Ook kan de gyroscoop rotatie meten wat een accelerometer niet kan.

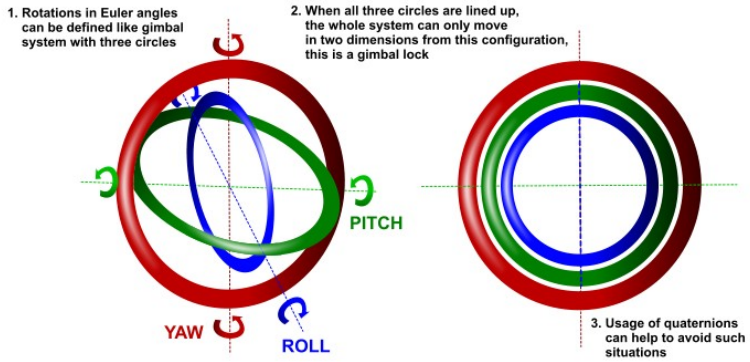
De magnetometer kan magnetische velden detecteren. Dit wordt gebruikt om het noorden te vinden aan de hand van het aardmagnetisch veld.

De beweging van een apparaat kan opgesplitst worden in zijn rotatie en translatie. Dit verslag bespreekt zowel de implementatie als de nauwkeurigheid van deze beiden.

2.1 Rotatie

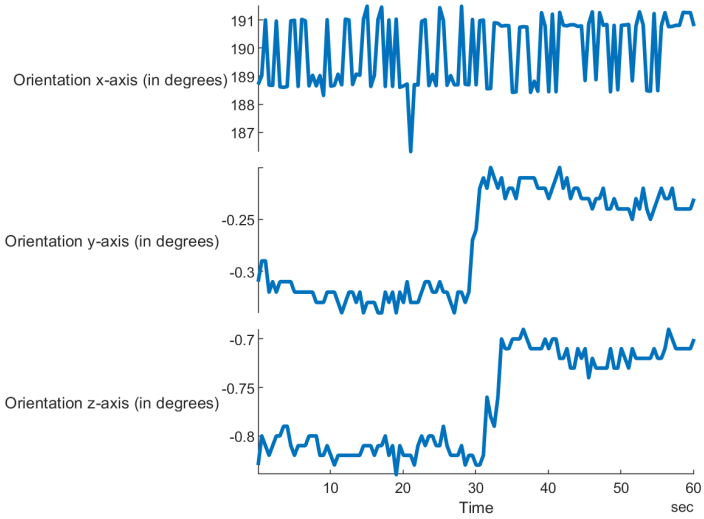
Voor de rotatie wordt de **OrientationSenor** interface gebruikt. Hierbij is nog de keuze tussen de **AbsoluteOrientationSensor** interface en de **RelativeOrientationSenor** interface. Het verschil tussen beiden is het assenstelsel waarin de rotatie gedefiniëerd wordt. Bij de absolute is dit het referentie coördinatensysteem van de aarde. Bij de relatieve is dit een stationair coördinatensysteem. Er wordt voor de relatieve geopteerd omdat deze geen gebruik maakt van de magnetometer voor het bepalen van het referentie coördinatensysteem van de aarde. Deze kan namelijk verstoord worden door magnetische velden en dus de accuraatheid aantasten [2].

Ook is er de keuze tussen Eulerhoeken en quaternionen om de oriëntatie te beschrijven [3]. Hiervoor worden de quaternionen gekozen omdat deze een Gimbal lock vermijden. Indien Eulerhoeken worden gebruikt kan dit voorkomen, waardoor er een vrijheidsgraad verloren gaat en dus informatie over de orïentatie. Zie figuur 1 voor een duiding.



Figuur 1: Illustratie van een Gimbal lock, een vrijheidsgraad gaat verloren [4]

Om een zicht te krijgen op de nauwkeurigheid van de sensoren in een smartphone, is een experiment uitgevoerd met een Samsung Galaxy S9. De smartphone werd gedurende 60 seconden plat op tafel gelegd. In deze tijd werden de accelerometer en gyroscoop uitgelezen aan de hand van de applicatie "AndroSensor"[5].

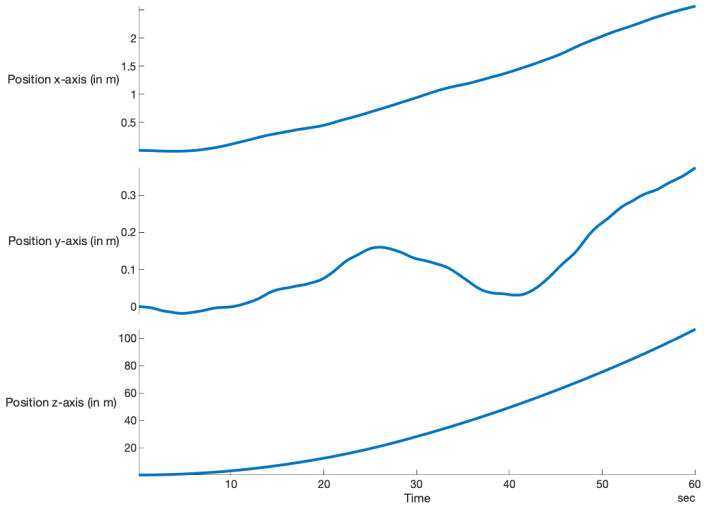


Figuur 2: Oriëntatie apparaat in rust, plat liggend op tafel gedurende 60 seconden volgens de x-as, y-as en z-as met behulp van de **RelativeOrientationSenor** interface.

Zie figuur 2 voor de oriëntatie van het apparaat dat plat ligt op een tafel en niet beweegt. Gedurende deze 60 seconden aan data is te zien dat er altijd een fout zit op de gemeten waarde. Deze is wel beperkt tot maximaal enkele graden, dit maakt het accuraat genoeg voor de *ScreenCaster*.

2.2 Translatie

Voor de translatie moet gekeken worden naar de acceleratie van het apparaat. Hiervoor wordt de **LinearAccelerationSensor** interface gebruikt. Hierbij is de zwaartekracht al van de gemeten waarde afgetrokken. De positie wordt dan berekend aan de hand van een dubbele integraal over de acceleratie omdat de acceleratie de tweede afgeleide is van de positie.



Figuur 3: Positie apparaat in rust, plat liggend op tafel gedurende 60 seconden volgens de x-as, y-as en z-as met behulp van de **LinearAccelerationSensor** interface.

Zie figuur 3 voor de positie van het apparaat dat plat ligt op een tafel en niet beweegt gedurende 60 seconden. Hierop is te zien dat de positie volgens de z-as een kwadratische drift heeft. Na 60 seconden loopt de fout hierdoor al op tot ongeveer 100 meter. Indien voor de positie volgens de x-as een lineaire regressie wordt opgesteld verkrijgt men een R^2 gelijk aan 0.59. Hierdoor kan er verondersteld worden dat deze fout zich lineair zal verder zetten. Na 60 seconden bedraagt de fout hier ook al twee meter. De fout op de positie in de y-as blijft bescheiden maar heeft toch een enigszins steigend verband. Deze bevindingen komen doordat ook bij de accelerometer er een fout zit op de gemeten waarde. Dit zorgt ervoor dat de translatie van een apparaat berekenen aan de hand van sensoren niet mogelijk is voor de applicatie.

2.3 Vinden van rotatie

De rotatie van het apparaat tussen twee momentopnames zal het relatief verschil zijn tussen twee opeenvolgende oriëntaties. De oriëntatie zal eerst ingeladen worden in een *DOMMatrix* [6], deze zal uiteindelijk de transformatiematrix genereren.

3 Visuele aanwijzingen

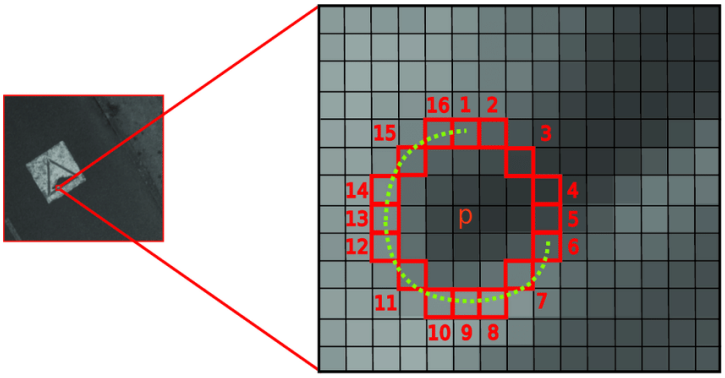
Met behulp van de camera, kan een apparaat een beweging detecteren. Deze beweging vertaalt zich vervolgens in een transformatie. Om beweging te detecteren zijn er punten nodig waarop het algoritme zich kan focussen, dit zijn de **keypoints**. Tussen verschillende frames zullen deze keypoints zich verplaatsen. Het linken van de keypoints tussen deze twee frames zal resulteren in een beweging.

Een algoritme dat in de *Technical Comitees* naar voren kwam is het **ORB** algoritme [7]. ORB staat voor **Oriented FAST and Rotated BRIEF**. In *ORB: an efficient alternative to SIFT or SURF (2011)* bespreken E. Rublee, V. Rabaud, K. Konolige en G. Bradski waarom ORB te verkiezen is boven SIFT of SURF, twee andere keypoint detectiealgoritmes.

Deze sectie bespreekt eerst FAST, een algoritme om keypoints te detecteren. Vervolgens gaat het over naar BRIEF, dit algoritme koppelt de gevonden keypoints aan elkaar.

3.1 FAST

FAST staat voor *Features from Accelerated Segment Test* en werd geïntroduceerd door E. Rosten en T. Drummond in 2005 [8]. FAST zoekt naar hoeken in de afbeelding waar de intensiteit van de omgeving hard verschilt. Rond een pixel p worden alle pixels in een *Bresenham cirkel* met straal drie bekeken. Wanneer er bij twaalf opeenvolgende pixels de intensiteit verschilt met minimaal t, een voorafbepaalde drempelwaarde, wordt p als een *keypoint* beschouwd, zie figuur 4. Om het algoritme snel genoeg te laten verlopen maar niet teveel keypoints te vinden, wordt een drempelwaarde tussen de 20 en 70 gebruikt. Afhankelijk van de omgeving kan deze manueel worden aangepast in de applicatie.

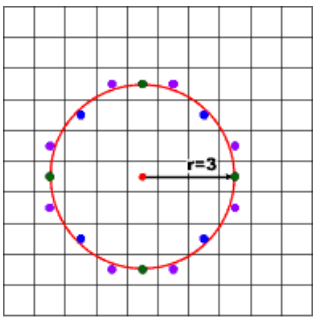


Figuur 4: Pixel p voldoet aan alle voorwaarden om een *keypoint* te zijn. [9]

Intensiteit De intensiteit van een pixel is gelijk aan zijn grijswaarde. Bij de implementatie wordt de foto eerst omgezet naar een grijswaarde foto. De intensiteit zal zo een gewogen waarde worden van de RGB–waarden van de pixel. De coëfficiënten zijn de waarden gebruikt en vastgelegd door de *NTSC* [1].

$$Intensiteit = 0.299 * R + 0.587 * G + 0.114 * B$$

Bresenham cirkel Het Bresenham algoritme is geïntroduceerd in de *Computer Graphics* om lijnen en andere vormen als pixels af te beelden. Door dit algoritme toe te passen zal elke cirkel die genomen wordt, altijd éénzelfde vorm aannemen.[10] Zie afbeelding 5 voor een visualisatie van een Bresenham cirkel met straat drie.



Figuur 5: Bresenham cirkel met straal drie. De aangeduide pixels worden gebruikt om de cirkel te definiëren. [11]

[1] NTSC is *The National Television System Committee*, een analoge norm voor kleurentelevisie geïntroduceerd in 1954.

3.2 BRIEF

BRIEF staat voor *Binary Robust Independent Elementary Features*. Dit algoritme gebruikt de gevonden keypoints volgens FAST (3.1) en probeert voor elk keypoint een match te vinden met een keypoint van een andere set. [12]

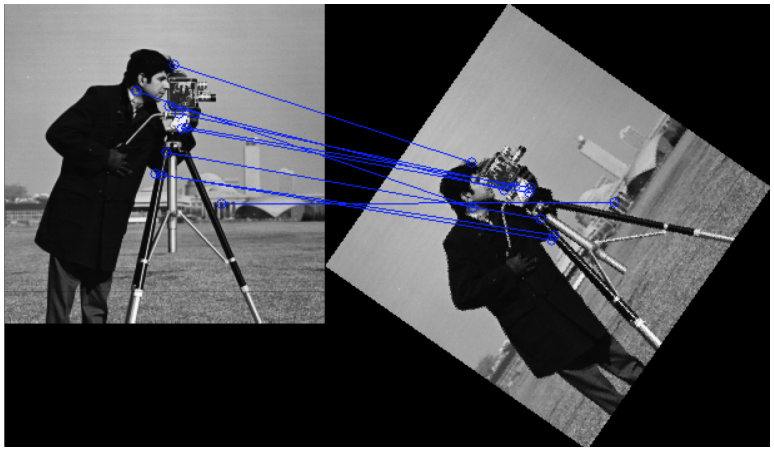
BRIEF koppelt paren van keypoints door een binaire string op te stellen voor elk keypoint van dat paar. Door deze binaire strings te vergelijken aan de hand van de *Hamming Distance* [13] kijkt het algoritme of deze keypoints een match zijn of niet. Daarbij bekijkt het ook hoe nauwkeurig de koppeling is, des te kleiner de *Hamming Distance* des te zekerder het algoritme is dat deze twee keypoints bij elkaar horen.

Keuze voor BRIEF in ORB Verschillende keypoint-matching algoritmes zoals SIFT en SURF beschrijven een keypoint in een vector van 512 bytes. Het aanmaken van al deze beschrijvingen loopt al snel op bij een groot aantal gevonden keypoints en neemt veel geheugen in beslag. Daarenboven duurt het matchen langer hoe meer geheugen er gebruikt wordt voor de beschrijvingen. BRIEF slaat de stap van de beschrijving maken over om keypoints te matchen wat een enorme winst aan geheugen oplevert zonder accurate matching op te geven. [14]

Implementatie De implementatie en parameters zijn volledig gebaseerd op de BRIEF–implementatie van *trackingjs* [15]. Dit is een in de praktijk bewezen correcte en efficiënte implementatie.

3.3 Vinden van translatie

Met behulp van de keypoints, gevonden door *FAST* en gekoppeld door *BRIEF*, kan er een translatiebeweging worden gevonden. Alle gekoppelde keypoints hebben een nauwkeurigheid, deze geeft weer hoe zeker *BRIEF* ervan is dat de twee keypoints bij elkaar horen. Vanaf dat deze nauwkeurigheid boven de zeventig procent ligt, kijkt het algoritme wat de translatie volgens de x- en y-as is. De gemiddelde translatie van alle keypoints die aan de genoemde criteria voldoen, wordt als algemene translatie van het apparaat genomen. Zie figuur 6 voor een illustratie.



Figuur 6: Het matchen van de keypoints in twee frames. [16]

4 Implementatie

De vorige twee secties bespraken de twee verschillende manieren, sensoren en visuele aanwijzingen, om beweging van een apparaat te meten. Bij sensoren is de rotatie zeer accuraat, de translatie daarentegen is niet bruikbaar. Daartegenover staat het *ORB*–algoritme, hierbij is de rotatie niet bruikbaar maar de afgeleide translatie accuraat. Door het samenvoegen van de twee, rotatie door de sensoren en translatie door de visuele aanwijzingen, is het mogelijk om een apparaat te volgen in een 3D omgeving.

4.1 Rotatie

Bij rotatie wordt de afgebeelde foto geroteerd volgens de gegenereerde transformatiematrix. De rotatie wordt gedefinieerd als het verschil tussen de beginoriëntatie en de huidige oriëntatie. Zoals eerder vermeld zal de transformatiematrix gegenereerd worden door een *DOMMatrix*. Om een juiste rotatie voor ieder scherm te bekomen, moet er rekening gehouden worden met de al bestaande transformatie. Aangezien deze een andere oorsprong heeft, moet de rotatie aangepast worden met volgende formule:

$$Rotatie = T^{-1} * R_0 * T$$

Met T de translatie van de rotatie-oorsprong naar de transformatie-oorsprong en R_0 de rotatie van het apparaat.

4.2 Translatie

Bij translatie wordt de afgebeelde foto verplaatst over een afstand gelijk aan het aantal berekende pixels. Ook hier zal de foto de schermen niet

volledig meer overlappen, de applicatie zal de afbeelding niet keer op keer herschalen om de transformaties zo natuurlijk mogelijk aan te doen voelen voor de gebruiker.

4.3 Integratie

In de applicatie zijn de twee, rotatie en translatie, van elkaar gescheiden gehouden. De gebruiker kan kiezen om op een bepaalde plaats te roteren of zonder roteren zichzelf horizontaal of verticaal te verplaatsen. Het samennemen van de twee transformaties zorgt nog voor problemen. Zo zal een rotatie ook als een translatie worden beschouwd door visuele aanwijzingen. Om te zorgen dat deze twee tegelijkertijd gebruikt kunnen worden, is er meer onderzoek nodig naar hoe de twee facetten elkaar beïnvloeden. Schaling is ook buiten beschouwing gelaten in de applicatie, het nauwkeurigste resultaat in deze *'proof of concept'* wordt vooropgesteld.

4.4 Limitatie

Het gebruiken van de sensoren in een apparaat wordt niet standaard door elke browser ondersteund [17]. Momenteel werkt deze feature dus enkel in *Chrome* en *Edge*. Dit wil zeggen dat het master device gebruik moet maken van deze browsers, wat niet geldt voor de slave schermen. Slave apparaten hoeven geen toestemming te geven tot hun sensoren en kunnen dus gebruik maken van de niet ondersteunde browsers. *Mozilla Firefox* bijvoorbeeld, ondersteunt dit momenteel niet omwille van privacy redenen. Hier kan in de toekomst wel verandering in komen [18]. Om veiligheidsredenen moet de verbinding over *HTTPS* gaan, dat momenteel al de standaard is voor de applicatie.

5 Besluit

In de uiteindelijke applicatie zullen de rotatie en translatie niet samen kunnen worden toegepast. De gebruiker kan zelf kiezen welke van de twee hij/zij zal toepassen. De sensoren zoeken de juiste rotatie, met behulp van visuele aanwijzingen van de camera wordt de translatie bepaald. De tweede maakt gebruik van het *ORB*–algoritme. Beiden zouden elkaar in een later stadium kunnen aanvullen. Doordat er nog niet genoeg onderzoek is geweest naar de combinatie, is dit nog niet mogelijk. Met deze aanpassing in de applicatie *ScreenCaster* krijgt de gebruiker de mogelijkheid om zichzelf te bewegen zonder dat het realistisch perspectief verdwijnt.

Referenties

[1] Manisha Priyadarshini. Which Sensors Do I Have In My Smartphone? How Do They Work? https://fossbytes.com/which-smartphone-sensors-how-work/, 2018.

[2] Kenneth Rohde Christiansen and Annsi Kostiainen. Orientation Sensor. https://www.w3.org/TR/orientation-sensor/, 2019.

[3] Moti Ben-Ari. A tutorial on euler angles and quaternions. *Department of Science Teaching Weizmann Institute of Science*, 2014.

[4] Avoid gimbal lock for rotation/direction maya manipulators. https://around-the-corner.typepad.com/adn/2012/08/avoid-gimbal-lock-for-rotationdirection-maya-manipulators.html.

[5] fivasim. AndroSensor for Android. http://www.fivasim.com/androsensor.html, 2015.

[6] MDN Contributors. Dommatrix. https://developer.mozilla.org/en-US/docs/Web/API/DOMMatrix, Jan 2020.

[7] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. *2011 International Conference on Computer Vision*, 2011.

[8] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. *Tenth IEEE International Conference on Computer Vision (ICCV05) Volume 1*, 2005.

[9] Software/ fast corner detection. https://computervisiononline.com/software/1105138582.

[10] Jurg Nievergelt and Claus Hinrichs. *Algorithms and data structures - applications to graphics and geometry*. Textbook Equity Editions, 2014.

[11] Penny Rheingans. Finding pixels that lie on a circle. https://www.csee.umbc.edu/~rheingan/435/pages/res/gen-3.Circles-single-page-0.html, Sep 2003.

[12] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. *Computer Vision – ECCV 2010 Lecture Notes in Computer Science*, page 778–792, 2010.

[13] Nitya Raut. What is Hamming Distance? https://www.tutorialspoint.com/what-is-hamming-distance, 2018.

[14] Abid K. Alexander Mordvintsev. BRIEF (Binary Robust Independent Elementary Features). https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_brief/py_brief.html, 2013.

[15] trackingjs. Source: features/Brief.js. https://trackingjs.com/api/Brief.js.html, 2014.

[16] module' object has no attribute 'drawmatches' opencv python. https://stackoverflow.com/questions/20259025/module-object-has-no-attribute-drawmatches-opencv-python/26227854#26227854.

[17] MDN Contributors. Permissions. https://developer.mozilla.org/en-US/docs/Web/API/Permissions, Sep 2019.

[18] MDN Contributors. Feature-policy. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Feature-Policy, Feb 2020.