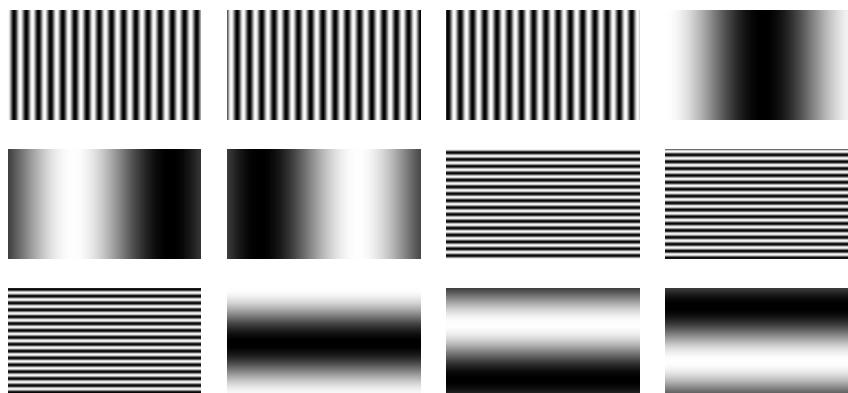


Dense Reconstruction using Structured Light

24 maart 2022

Het doel van deze oefening is het maken van een dense reconstructie van een scène, met behulp van *structured light* scanning. Structured light scanning is een actieve techniek (zie les 5: actieve 3D reconstructie), d.w.z. dat er actief informatie aan de scène wordt toegevoegd om meer correspondenties te bekomen. Dit in tegenstelling tot passieve technieken, zoals b.v. stereo (zie les 4: passieve dichte stereo beeldinterpolatie en dieptereconstructie). Het principe van structured light scanning is simpel. Er worden een aantal patronen (sinuspatronen, Gray Codes¹,...) geprojecteerd op een scène. Na de projectie heeft ieder belicht deel van het tafereel een unieke code, waarmee correspondenties gevonden kunnen worden tussen camerabeelden. In deze opgave zal je structured light in eerste instantie implementeren m.b.v. sinuspatronen.

De oefening gebruikt twee camera's als invoer. Elke camera heeft een reeks van afbeeldingen waarbij 3 structured light patronen worden geprojecteerd op een model van een huisje. Die projectie gebeurt zowel in de horizontale als de verticale richting. In totaal worden er dus 6 patronen geprojecteerd (zie Figuur 1). Figuur 2 toont de overeenkomstige input afbeeldingen van 1 camera.



Figuur 1: Voorbeeld van sinuspatronen.

¹https://en.wikipedia.org/wiki/Gray_code

1 Opgave

Het doel van de opgave is om de invoerafbeeldingen van Figuur 2 te gebruiken om correspondenties te bekomen tussen de twee camera's. De correspondenties worden dan gebruikt om de extrinsieke calibratie-informatie te berekenen (zie nota's hoofdstuk 5). Dit omvat de cameraoriëntatie en het projectiecentrum. Vervolgens kan de extrinsieke calibratie gebruikt worden om de 2D-correspondenties te herprojecteren in 3D.

Week 1

1. Genereer, gebruik makend van de structuur in het gegeven framework, een reeks van sinuspatronen die geprojecteerd kunnen worden op de scène om de invoerafbeeldingen te genereren. Er zijn 3 patronen nodig per frequentie voor zowel de horizontale als de verticale richting. De drie patronen per frequentie zijn 3 sinusgolven, verschoven in fase met 120 graden. Hier kiezen we voor twee frequenties. In totaal moeten er dus 12 patronen gegenereerd worden.



Figuur 2: Voorbeeld invoerafbeeldingen van twee camera's gebruik makend van sinuspatronen.

2. Om correspondenties te vinden tussen het geprojecteerde patroon en het beeld van de camera, is het belangrijk dat iedere pixel een unieke identifier heeft. Bij het door ons gekozen patroon kan de fase op iedere pixelpositie hiervoor gebruikt worden. Het berekenen van de fase kan door de 3 verschoven sinuspatronen te gebruiken om 3 vergelijkingen op te stellen, die vervolgens opgelost kunnen worden naar phi (de fase) [ZHZ06].

$$I_1(x, y) = I'(x, y) + I''(x, y)\cos[\phi(x, y) - 120] \quad (1)$$

$$I_2(x, y) = I'(x, y) + I''(x, y)\cos[\phi(x, y)] \quad (2)$$

$$I_3(x, y) = I'(x, y) + I''(x, y)\cos[\phi(x, y) + 120] \quad (3)$$

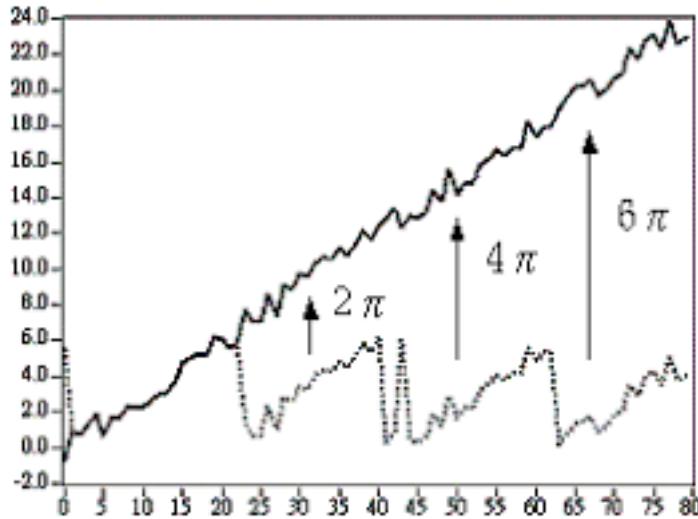
I staat hierbij voor *intensity image*. Uiteindelijk kan de fase berekend worden door de boogtangens van een combinatie van de 3 patronen te berekenen:

$$\phi(x, y) = \arctan\left(\sqrt{3} \frac{I_1 - I_3}{2I_2 - I_1 - I_3}\right) \quad (4)$$

De textuurafbeelding kan berekend worden door de 3 intensiteitsafbeeldingen uit te middelen:

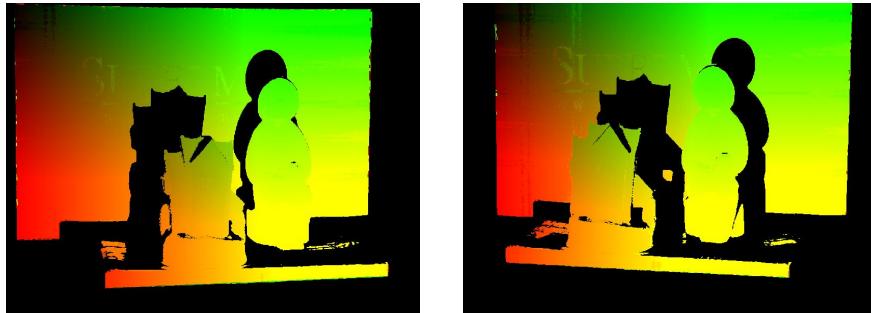
$$I'(x, y) = (I_1 + I_2 + I_3)/3 \quad (5)$$

3. Teken de gevonden fases per pixel uit voor zowel de laag- als hoogfrequente sinusgolf. Je zal zien dat er herhalingen in de afbeelding zitten. Dit komt omdat de fase binnen één periode nauwkeurig te bepalen is, maar om de absolute fase te bepalen over het hele beeld moet er nog een constante factor, vermenigvuldigd met 2π , bij opgeteld worden (Figuur 3). Dit proces noemt men het ontrollen van de fase. Schrijf een functie in je programma die de fase uit de vorige stap ontrolt door ook de laagfrequente sinuspatronen te gebruiken die 1 grote periode doorlopen doorheen het hele scherm. Dit patroon kan je gebruiken als leiddraad om te weten welke factor je moet optellen bij de lokale fase om tot de globale fase te komen.



Figuur 3: Fase ontrollen. De horizontale as duidt de kolom van een pixel aan op een scanline uit het camerabeeld. De verticale as geeft de fasewaarde aan. De niet-ontrolde fase herhaalt zich steeds en moet met een factor 2π vermenigvuldigd worden om een unieke fase te bekomen.

4. Gebruik de identificatie van de vorige stap om iedere pixel een unieke id te geven, d.w.z. maak een false color visualisatie zoals in Figuur 4.
5. Gebruik de checkerboard patronen uit de map *chess* om de intrinsieke calibratie matrix K van de camera te berekenen. Let op, de sequenties van beide standpunten zijn berekend met dezelfde camera, dus er moet maar 1 matrix K berekend worden.

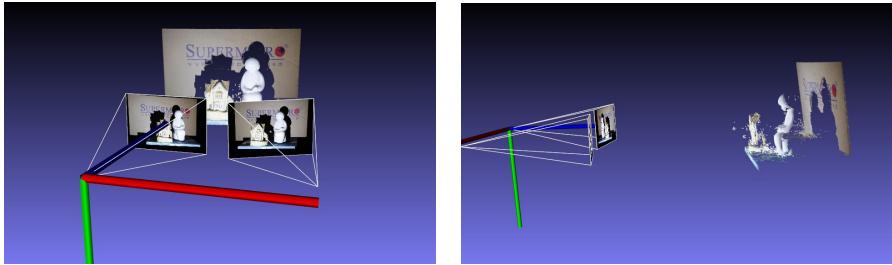


Figuur 4: False color visualisatie van de globale fase voor beide views. Deze globale fase komt eenduidig overeen met beeldcoördinaten van de projector.

6. Verwijder distortie van de invoerbeelden (Tip: *initUndistortRectifyMap*, *remap*,

Week 2

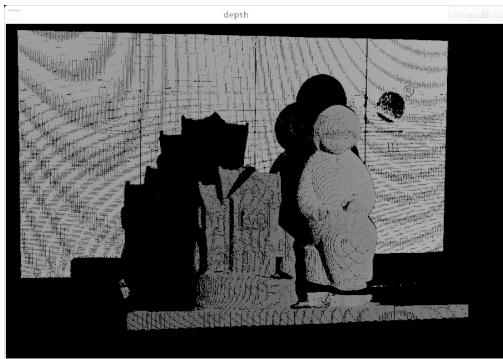
7. Zoek op basis van de globale fase op iedere pixelpositie correspondenties tussen beide camera's. Maak een debug visualisatie om de gevonden matches weer te geven. Je kan bijvoorbeeld de linker- en rechterview tonen en vervolgens in een lus random matches selecteren en aanduiden met een random gekleurd bolletje. Dit is handig om na te gaan of de matches kloppen.
8. Gebruik de bekomen correspondenties om de essentiële matrix te berekenen (Tip: *findEssentialMat*).
9. Bereken de rotatie- en translatierichting aan de hand van de essentiële matrix. Je hebt nu een gok voor de pose van de tweede camera. De eerste camera staat in de oorsprong. (Tip: *recoverPose*).
10. De calibratieinformatie, samen met de 2D-puntcorrespondenties, kunnen nu gebruikt worden om de 3D-coördinaten te bekomen. Doe dit m.b.v. de ingebouwde triangularisatiefunctie *triangulatePoints*. Los nu ook manueel het triangularisatieprobleem op door voor ieder punt een stelsel in de vorm van $Ax = B$ op te lossen. Het stelsel kan opgebouwd worden met behulp van de projectiematrices ($K * R - T$) van camera 1 en 2 en de 2D-correspondenties. Los het stelsel op met SVD (Tip: *cv::SVD*). Meer info kan je lezen in *uitlegTriangulation.pdf* en *mvg_triangulation.pdf*.
11. Visualiseer de puntenwolk en de berekende poses (zie les 6: geometrische cameracalibratie / 3D reconstructie tot puntenwolk). Een voorbeeld is weergegeven in Figuur 5. Merk op: de conventie van het coördinatensysteem is anders (zie nota's OpenGL). Hier wordt een typisch computervisie assenstelsel gebruikt: de kijkrichting komt overeen met de positieve Z-as. X en Y geven de beeldcoördinaten naar rechts en naar onder.



Figuur 5: Resultaten van een gereconstrueerde puntenwolk en de camera poses.

Week 3

12. Maak en visualiseer een dieptemap. Itereer hiervoor over de pixels in het beeld en bereken de afstand tot het getriangulariseerde 3D-punt. Let op, om de dieptemap te kunnen visualiseren zullen de dieptewaarden moeten gemapped worden tussen 0 en 255. Een voorbeeld is weergegeven in Figuur 6.



Figuur 6: Dieptemap.

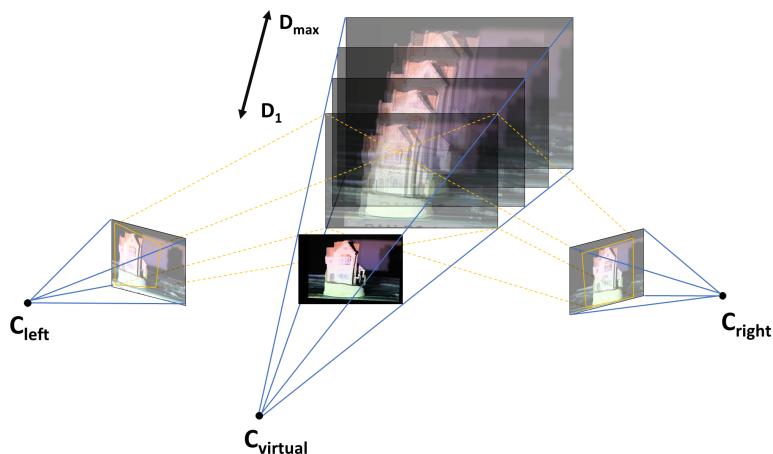
13. Implementeer warping van het beeld naar een nieuw standpunt. Je kan hiervoor de functie `cv::rgbd::warpFrame` gebruiken. Zorg dat je kan switchen naar een nieuw gewarpt standpunt d.m.v. het toetsenbord. Om de tussenliggende camerastandpunten te bepalen kan je de rotatiematrix omsetzen naar een quaternion om vervolgens tussen de rotaties te interpoleren (`Eigen::Quaternion`). Bij een juiste implementatie kan je tijdens dit switchen de parallax in de scène waarnemen. Een voorbeeld is weergegeven in Figuur 7.
14. Implementeer planesweeping (zie Les 4: consensus based rendering). Het berekenen van de tussenliggende camerastandpunten, alsook de visualisatie is dezelfde als bij image warping en kan dus herbruikt worden.
15. De dieptelagen moeten gecentreerd worden rond de scène. Bereken het centrum van de puntenwolk en gebruik de projectiematrix van de virtuele



Figuur 7: Image warping van source image naar destination image of tussenliggende beelden gebruik makend van dieptemap.

camera om de diepte van het centrum te achterhalen. Die diepte kan gebruikt worden om het aantal dieptelagen te beperken.

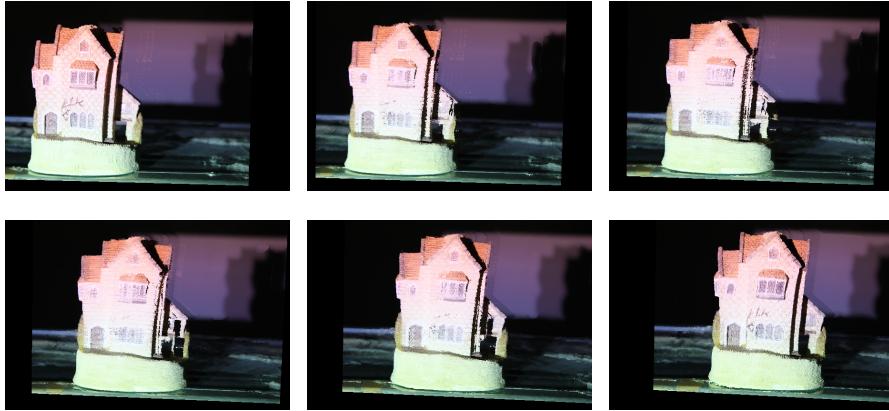
16. Gegeven een virtuele camera en een diepte, bereken het 3D dieptevlak D_i waarop geherprojecteerd moet worden. Deprojecteer de 2D hoekpunten van het image plane van de virtuele camera naar 3D om de 3D hoekpunten van het vlak te bekomen.
17. Herprojecteer de verschillende invoercameras op het dieptevlak D_i . Bepaal de homografie tussen het image plane van de afbeelding en het 3D vlak (tip: `cv::findHomography()`). Om dit te berekenen projecteer eerst de 3D hoekpunten van het vlak op het respectievelijke camerabeeld. Gebruik de vier hoekpunten van de afbeelding en de geprojecteerde 2D punten van het 3D vlak (aangegeven in geel in Figuur 8) om de homografie op te bouwen. Pas de homografie toe op de afbeelding (tip: `cv::warpPerspective()`).



Figuur 8: Overzicht algoritme plane sweeping.

18. Itereer over de verschillende dieptelagen en bereken de error tussen de verschillende herprojecties. Bereken voor iedere pixel de euclidische norm

van het verschil tussen de herprojectie van de linkse camera en de herprojectie van de rechtse camera (tip: `cv::absdiff()`, `cv::compare()`). De pixels waarvoor de error functie minimaal is, worden gebruikt als finaal resultaat in de geïnterpoleerde afbeelding. Figuur 9 toont een aantal resultaten.



Figuur 9: Geïnterpoleerde virtuele standpunten na plane sweep.

extra Gebruik in de error metriek een neighborhood rond iedere pixel (tip: `cv::filter2D()`).

extra Gebruik de puntenwolk uit de *structured light* reconstructie om lokaal het bereik van de dieptevlakken meer accuraat te bepalen.

Week 4

Extra Implementeer structured light ook met graycodes i.p.v. sinuspatronen. Maak een vergelijking tussen beide methodes. Welke sterktes/zwaktes zie je in beide?

Extra Maak een eigen implementatie van de warp functie (zie les 1: warping and image-based rendering, nota's McMillan).

Extra De projector werd tot nu toe enkel gebruikt om een dicht rooster van correspondenties tussen de twee camerabeelden te genereren. Je kan er echter evenveel informatie uit halen als een camera zelf. Inderdaad: veel structured light systemen gebruiken slechts 1 camera en een projector en gebruiken de correspondenties tussen die twee. Pas je oplossing aan tot een drie-aanzicht oplossing, inclusief calibratie van de projector.

2 Modaliteiten

- Het practicum wordt gemaakt in groepjes van twee personen. Je bent vrij in de partnerkeuze. De groepsindeling kan gemaakt worden via de GitHub Classroom, waar je een repository toegekend krijgt per groep.

- De opgave telt mee voor een derde van de examenpunten voor het vak. Zonder tegenindicaties, worden partners in een groep gelijk gekwoteerd.
- Het practicum wordt onafhankelijk van andere groepjes gemaakt. Code van anderen gebruiken, of je code door anderen laten gebruiken, mag enkel indien dat op voorhand en door beide partijen gerapporteerd wordt aan de begeleider, en als dat gecompenseerd wordt door ander werk (in overleg met de begeleider). Je kan best op voorhand niet met andere groepjes discussiëren hoe je de problemen in dit practicum zal oplossen. Onregelmatigheden worden op dezelfde manier gesanctioneerd als spieken op een examen.
- Rapportering: Je programmacode wordt met beknopt verslag ingediend via de GitHub Classroom Assignment. Alle code, resultaten en verslag moet ten laatste op **vrijdag 20 mei om 08h00 gecommit en gepushed worden naar de GitHub Classroom repository van het practicum**. Tussenliggend resultaten en resultaten van de reconstructie moet ook toegevoegd worden aan het ingezonden resultaat. Het verslag beschrijft in het kort (richtlijn: 3 pagina's), wat er in het zip bestand staat, voldoende tussenresultaten voor alle stapjes, en bevat alle andere informatie die relevant kan zijn voor de kwotering van het practicum, zoals bijvoorbeeld: onderdelen van de opgave die je niet geïmplementeerd hebt, eventuele samenwerking met anderen (en hoe die gecompenseerd werd), extra mogelijkheden, etc...
- Demo: op **vrijdag 20 mei vanaf 09h00** zal je je practicum demonstreren. Tijdens de demonstratie kan er gevraagd worden je programma te laten lopen op afbeeldingen van een door ons gegeven voorbeeldscène. Er kunnen vragen gesteld worden over hoe je bepaalde deeltaken hebt opgelost, er kan gepeild worden naar je inzicht in de materie, en je krijgt de kans om eventuele extra's te laten zien. Je kan uiteraard ook toelichten waar je eventuele problemen mee had. De demonstratie zal ca. 20 minuten per groep duren, en is bepalend voor de kwotering.
- Kwotering: een minimale, werkende implementatie van de volledige opgave levert een basisscore van 7 op 10 op. Deze score wordt verminderd of vermeerderd naargelang de kwaliteit van je werk.

Referenties

- [ZHHZ06] Song Zhang, Peisen S. Huang, Peisen S. Huang, and Song Zhangb. A fast three-step phase-shifting algorithm. 2006.