Practicum structured light

Koen Laermans

Joep Stevens

2022

1 Introduction

Voor dit project hebben we alle minimale vereisten geïmplementeerd met enkele ectra's. Alleen vraag 13 is niet gelukt omdat we de functie "cv::rgbd::warpFrame" niet werkende kregen. De code voor de vragen is te vinden in de map code. Elke file "qx.py" kan een aantal vragen beantwoorden. De files beantwoorden de volgende vragen:

- q1.py: 1
- q3.py: 2-3
- q4.py: 4
- q5.py: 5-6
- q7.py: 7
- q11.py: 8-12
- q13.py: 13 (incompleet en werkt niet)
- q14.py: 14-18 + extra neighborhood

In de folder "results" zijn een aantal resultaat afbeeldingen te zien voor elke python file.

Vraag 1

Enkele resultaten van vraag 1 worden getoond in Figuur 1. In deze figuur wordt er links en rechts respectievelijk een hoogfrequente en een laagfrequente sinus geprojecteerd met verschillende verschuivingen van de fase.

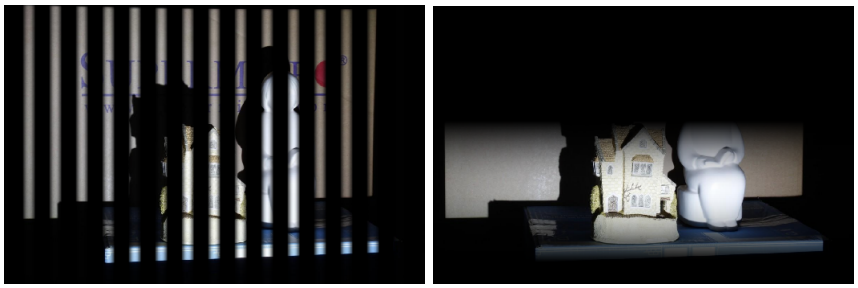

Figure 1: Een hoge frequentie en lage frequentie sinus golf geprojecteerd op de afbeeldingen.

Vraag 2-3

Voor vraag 2 hebben we eerst de fase op iedere pixel berekend voor de hoog- en laagfrequente sinusgolven. Deze worden gevisualiseerd in Figuur 2. De ruis aan de randen van de foto wordt later weggefilterd door een mask te vermenigvuldigen met de foto, waardoor de pixels die niet belicht worden door de projector zwart gemaakt worden.

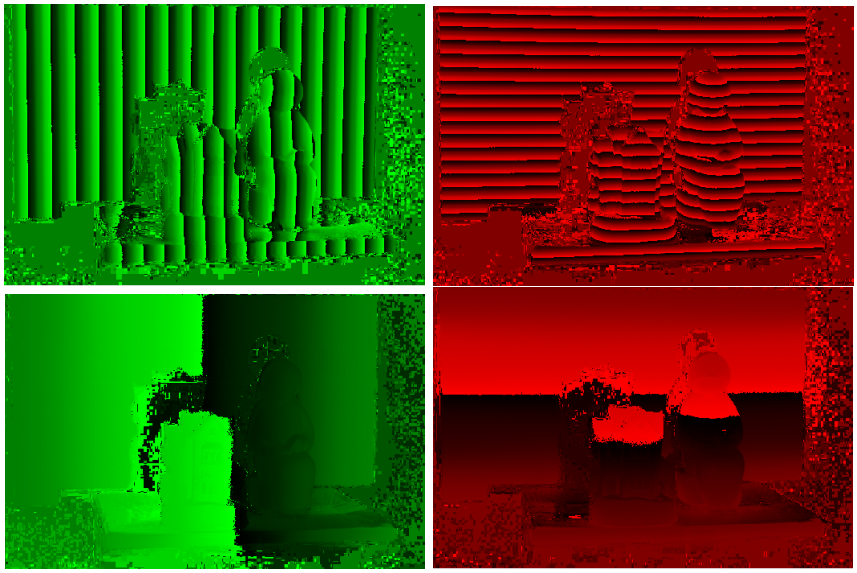

Figure 2: De berekende fases voor de hoge frequentie en lage frequentie afbeeldingen.

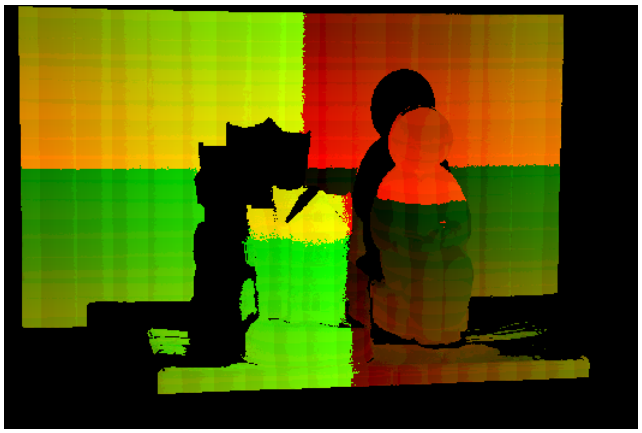

Figure 3: False color representatie van de berekende identifiers.

Vraag 4

Vervolgens hebben we de fase van de hoog- en laagfrequente sinusgolven gecombineerd om een globale fase/unieke identifier te bekomen voor iedere pixel. Deze hebben we omgezet in een false color visualisatie, zoals weergegeven in Figuur 3. In deze figuur zien we enkele artefacten in de vorm van verticale lijnen. Deze geven aan dat er nog iets mis is en dat niet iedere pixel uniek is, maar we hebben dit niet kunnen oplossen. In de matching stap zullen we weergeven hoe accuraat onze identifiers zijn.

Vraag 7

Nu dat we de beelden van de linkse en rechtse camera omgezet hebben in unieke identifiers, kunnen we deze pixel per pixel met elkaar matchen. Het resultaat van deze matching wordt weergegeven in Figuur 4. Hierbij negeren we volledig zwarte pixels die aangeven dat een pixel geen id heeft. We vinden voor de meeste pixels een match, maar er ontbreken een aantal matches op de horizontale en verticale lijnen. Deze komen mogelijks overeen met de artefacten uit de false color visualisatie. Deze ontbrekende lijnen kunnen we wel matchen als we voor 1 id in de rechtse afbeelding meerdere matches toelaten.

Vraag 11-12

Nu dat we de projectiematrices van beide cameras berekend hebben in vraag 8-10, kunnen we deze informatie combineren met de matches tussen beide standpunten om een 3D puntenwolk te berekenen. Het resultaat van de puntenwolk wordt weergegeven in Figuur 5. Links wordt een puntenwolk weergegeven van de door ons gevonden matches. Rechts is de puntenwolk van de correcte matches

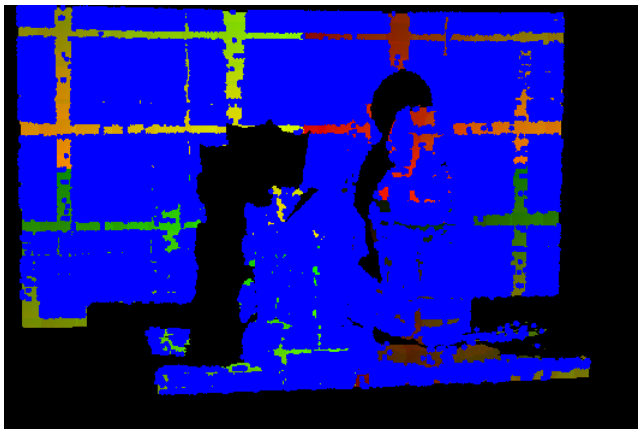

Figure 4: Blauwe pixels in de figuur zijn succesvol gematcht met een id in de linkse en rechte afbeelding.

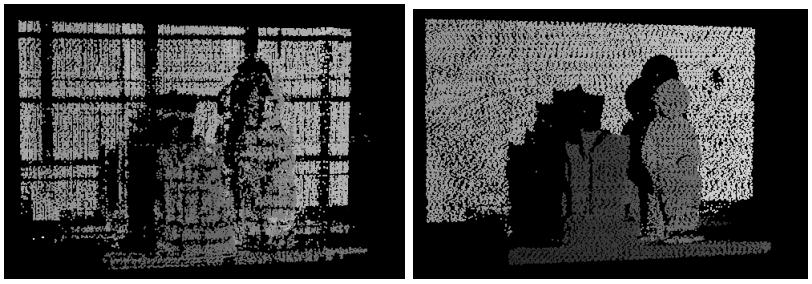

Figure 5: Links de gevonden puntenwolk voor onze matches, rechts de punten wolk voor de gekregen matches.

die wij van de profs gekregen hebben. We hebben de punten in deze puntenwolk zodanig gekleurd dat de diepste pixel wit is en de minst diepe pixel zwart. Voor het weergeven van de puntenwolken hebben we OpenGL gebruikt met een pointsize van 2.

Vraag 14-18

Voor planesweeping hebben we de afbeeldingshoeken geprojecteerd vanuit de virtuele camera om een vlak te bekomen op een specifieke diepte van de virtuele camera. Vervolgens hebben we dit vlak terug geprojecteerd naar de linkse en rechtse camera om een homografie te berekenen tussen beide cameras en het dieptvlak. Deze homografie hebben we gebruikt om de afbeelding van beide cameras te warpen naar het dieptevlak. In Figuur 6 is weergegeven hoe de linkse en rechtse afbeelding worden gewarped naar het dieptevlak op verschillende

Figure 6: Warping de images op verschillende dieptevlakken voor planesweeping.

dieptes. Het resultaat van de reconstructie met planesweeping wordt getoond in Figuur 7. Voor de positie en rotatie van onze virtuele camera interpoleren we tussen de linkse en rechtse camera. Voor de dieptes kiezen we het centrum van de punten wolk -1 en het centrum van de punten wolk +1 en dit delen we op in 200 dieptevlakken. 5 Verschillende standpunten tussen camera 1 en 2 worden getoond.

2 Extra's

2.1 Error Neighborhood

Voor de error metriek in een neighborhood rond iedere pixel te bereken hebben we eerst de errors berekend en daarna convolutie op deze errors toegepast met een 3x3 kernel gevuld met 1. De error wordt dus vervangen door het gemiddelde van de pixels in de buurt. We hebben ook andere kernel sizes geprobeerd, zoals 9x9, en andere waardes in de kernel. Een 3x3 gevuld met enkel 1 gaf bij ons het beste resultaat. De resultaten worden weergegeven in figuur 8.

2.2 Bereik dieptevlakken

Hiervoor hebben we de diepte van het meest diepe punt en het minst diepe punt genomen en deze opgedeeld in 200 vlakken.

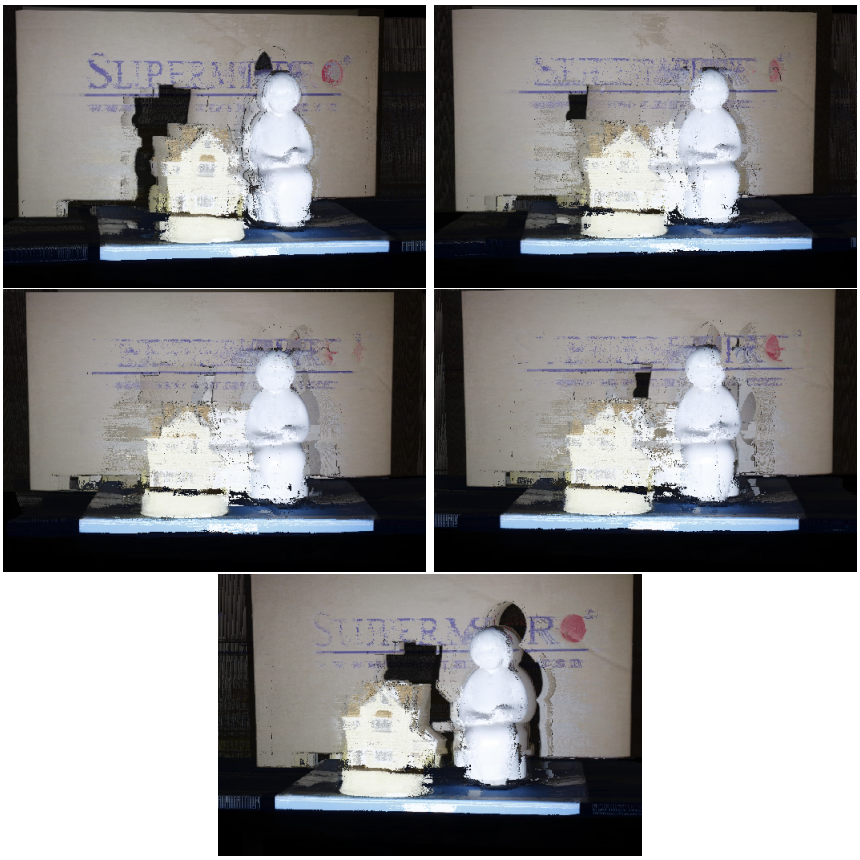

Figure 7: Reconstructie van de afbeelding in tussenliggende standpunten met behulp van planesweeping.

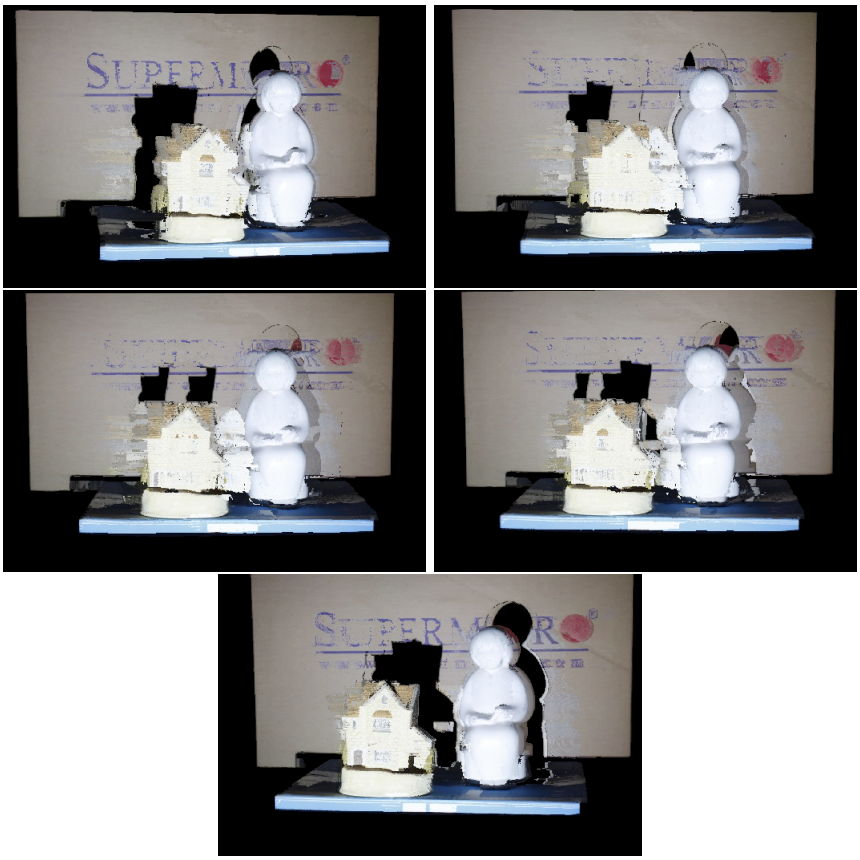

Figure 8: Reconstructie zoals in Figuur 7, maar nu met de extra neighborhood errors en beter minimum en maximum dieptevlakken.

2.3 Graycodes

Wanneer we deze methode vergelijken met sinuspatronen merken we op dat sinuspatronen meer accuraat zijn dan graycodes. Bij graycodes vonden we slechts één derde van het aantal matches bij sinuspatronen. Bovendien zijn er meer afbeeldingen/patronen nodig om een unieke ID te bekomen voor iedere pixel bij graycodes. Een voordeel van gray codes is dat de lichtpatronen conceptueel eenvoudiger zijn om te decoderen, omdat we per pixel enkel moeten kijken of deze belicht is of niet. Het resultaat van onze graycodes wordt weergegeven in figuur 9. In de bovenste 2 afbeeldingen worden de identifiers als kleur weergegeven en in de onderste afbeelding tonen we de pointcloud die we na matching en triangulatie hieruit afleiden.

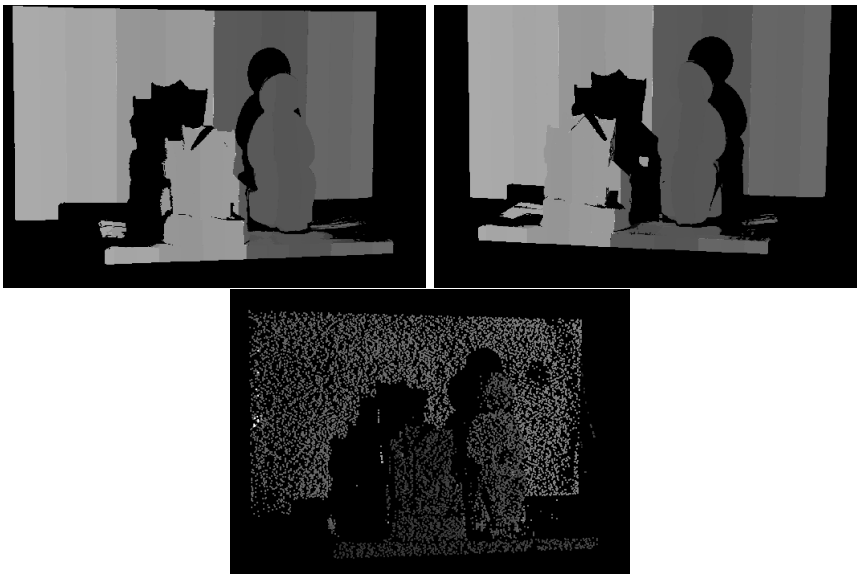

Figure 9: De linkse en rechte id images bekomen met behulp van graycodes en de pointcloud berekent met de matches.