Groep 5
Hidde Brands (2158176)
Ruben Boone (2261602)

Verslag 3D-modelling project

Introductie

In dit verslag over ons project van 3D modelling and Image Based Rendering gaan we over elke stap in de opgave en rapporteren we kort hoe we dit gedaan hebben en of we moeilijkheden mee hadden. Ook geven we op het einde nog een kort overzicht van de problemen die we ondervonden hebben en resultaten die afwijken van de basisoplossing. Ook geven we nog een overzicht van de inhoud van onze zip.

Stap 1:

- Deel 1
  We hebben het voorbeeld van OpenCV gevolgd. We nemen met de package glob alle afbeeldingen hun bestandslocatie en lezen deze nadien in. Zoals aangehaald in de opgave zijn de afbeeldingen verkleind met een scale, zo gebeurt de detectie sneller. Achteraf schalen we de hoekpunten van het checkerboard terug naar gewenste grootte.

- Deel 2
  Uit het vorige deel krijgen we object punten en image punten. Deze geven we mee met OpenCV's functie calibrateCamera. Hieruit krijgen we de matrix K (intrinsieke parameters), en de rotatie en translatie vectors.

- Deel 3
  In de directory “Result” is een subdirectory genaamd “undistorted”.
  Hierin zitten alle checkerboard afbeeldingen die door middel van de voorgaande matrix K ge-undistord zijn met behulp van remapping.

- Deel 4
  Zoals gevraagd hebben we met de rotatie- en translatie vectors de poses van de camera ten opzichte van het chessboard patroon berekend en opgeslagen in een .txt bestand. Het resultaat hiervan staat in ‘Result/camera_parameters.txt’. De visualisatie van de camera's gebeurd in de file ‘Src/step_1_visualise_3d.py’.

Stap 2

- Deel 1
  We hebben de meegeleverde klasse gebruikt om de patronen te genereren. We slaan de patronen op in de directory ‘Result/gce_patterns’
  Waarom gray codes en geen binaire voorstelling?
  Er is een 1 bit afwijking waardoor we makkelijk naburige checks doen. De volgende bit verandert dus slechts 1 bit ten opzichte van de vorige bit. Bij gewoon binair veranderen er meerdere bits per stap.

Groep 5
Hidde Brands (2158176)
Ruben Boone (2261602)

- Deel 2
  We maken een valid mask. Deze mask zal kijken of het verschil groter of kleiner is dan een threshold om zo te beslissen of de pixel valid is of niet. We bekijken het verschil tussen een gray code patroon en zijn inverse. Afhankelijk van de threshold zal een 1 of een 0 worden toegevoegd aan de bitmask. Als alles bekeken is zal de valid mask toegepast worden en houden we een lijst bij met de x, y coördinaat en de id van de gray code.

- Deel 3
  Voor dit deel bekijken we de 2 lijsten met de id's en gaan we punten matchen op basis van hun id's, hieruit krijgen we dan de punten uit alle twee de lijsten en een lijst met matches. Deze kunnen we meegeven aan OpenCV's *drawMatches* functie. Om het zichtbaar te houden pakken we een subset van de matches zodat we kunnen kijken of de overeenkomsten juist zijn.

Stap 3

- Deel 1
  Voor stap 3 geven we de relevante functies van stap 2 mee om aan de basis data te komen. Vervolgens berekenen we de essential matrices voor de verschillende camera's met de OpenCV functie 'findEssentialMat()'

- Deel 2
  Dan gebruiken we recoverPose() van openCV om de rotatie -en translatievectoren te berekenen. Als deze berekend zijn, visualiseren we deze ook op twee manieren: met een axis visualisatie en een pyramid visualisatie.

- Deel 3
  De triangularisatie doen we aan de hand van openCV's functie 'triangulatePoints()' en manueel. De versie met de openCV functie is eenvoudig en gebruikt gewoon simpel de 'triangulatePoints()' functie.
  De manuele versie gebruikt het stelsel gegeven in de opgave om dit te berekenen.

- Deel 4
  We hebben open3D gebruikt voor onze 3D visualisaties. We hebben dit wel gewoon in dezelfde file gedaan aangezien het geen conflicten gaf met de rest zoals aangegeven stond dat kon gebeuren in de opgave.

Stap 4

- Deel 1
  Voor het eerste deel hebben we een werkende implementatie, al is het niet super smooth. Er zijn twee delen. Het eerste scherm dat je te zien krijgt blijft

statisch staan van positie, maar de kleuren veranderen naargelang je weg beweegt of dichterbij beweegt. We hebben dit eerst zo gedaan zodat je een goed beeld op de scene blijft behouden terwijl je de camera ‘verplaatst’. In het tweede beeld kan je wel bewegen met de camera zelf en gaat je view van de scene wel veranderen.

- Deel 2
  De plane sweep van stap 4 is ons niet gelukt. We hebben heel veel geprobeerd en er vele uren aan gespendeerd, maar uiteindelijk hebben we geen werkende oplossing gevonden. We hebben wel een aantal resultaten gekregen waar mogelijk iets op staat dat in de buurt komt. Deze resultaten staan in de map ‘Result/step4’ en de namen van de files komen overeen met de filenamen van de verschillende testfiles die we hebben geprobeerd. Op één van de afbeeldingen in resultaten staat ook een view van de camera’s t.o.v. het scenevlak. Hierop kan je zien dat onze virtuele camera wel op de juiste positie staat. Het gaat dus ergens fout gaan bij het correct verkrijgen van de scene voor die plaats aan de hand van plane sweep.

Stap 5

In de directory “Result” zijn verschillende subdirectories. ‘ownChessImages’ zijn onze eigen chessboard kalibratie afbeeldingen. ‘ownDataSet1’ zijn onze afbeeldingen van de scene met spongebob. Hier hebben we 3 camerastandpunten genomen. In de directory ‘OwnDataset2DiffProjector’ hebben we dezelfde scène met 2 camera standpunten gecapteerd maar met de projector op een ander standpunt. Helaas zijn we hier vergeten om ook foto’s te nemen voor de projector kalibratie.

Stap 6

- Deel 1
  Alle functies van stap 2 en stap 3 zijn aangepast zodat deze ook werken voor meerdere camera standpunten. De functies gaan met behulp van for-loops hun resultaten opslaan in lijsten waarvan de index dan gelijk is aan de index van het standpunt van de camera (0,1,.....).
  Het aantal camera's kan meegegeven worden als argument van het script.
  Volgens ons gaat het matchen van correspondente punten niet juist, mogelijks omdat we afbeeldingen hebben waar de afbeelding redelijk blauw is door verkeerd licht.

- Deel 2
  Zoals eerder vermeld zijn we vergeten foto’s te maken om de projector te kalibreren. Hierdoor konden we deze stap dus niet juist testen. We hebben wel code geschreven die volgens ons zou moeten werken om de projector

kalibratie te laten werken, maar deze hebben we dus niet kunnen testen door het gebrek aan foto's. We nemen de intrinsieke parameters van de camera. We zoeken het chessboard patroon in de geprojecteerde afbeeldingen, en zo zoeken we de 3d punten vanaf de pose berekend. De 3D-wereldcoördinaten van het geprojecteerde patroon worden samen met de 2D coördinaten gebruikt om de in- en extrinsieke te vinden. We hebben bij de vorige stappen geen veranderingen gedaan om dit met de projector te laten werken aangezien we hier geen werkende versie van hebben, maar we nemen aan dat dit niet zo verschillend zou moeten zijn van onze huidige oplossing als we de parameters van de projector weten.

Extra's

We hebben twee extra's geïmplementeerd.
Ten eerste hebben we een mesh gemaakt van onze pointcloud in stap 3. Dit staat in de file van stap 3, helemaal onderaan.
Ten tweede hebben we 3 openCV functies zelf geïmplementeerd. De geïmplementeerde functies zijn grayscale, blur en een Sobel filter. Deze functies staan telkens in hun aparte file 'extra_opencv_<naam>.py'. We hebben telkens ook de vergelijking gemaakt met het resultaat van de openCV functie zelf.

Problemen

We hebben 2 grote problemen ondervonden.
Ten eerste is onze implementatie van 4.2 niet correct. We hebben een deel van de pogingen nog laten staan in de zip file die mogelijk in de buurt komen. Hieraan kan wel gezien worden dat we ons best hebben gedaan, maar we hebben uiteindelijk geen tijd gehad om de juiste oplossing te vinden.
Ten tweede waren we vergeten om bij de eigen dataset foto's te nemen om de projector te kalibreren. Hierdoor hebben we stap 6.2 niet helemaal juist kunnen uitvoeren. We hebben zo goed mogelijk code geschreven die volgens ons zou werken indien we wel de foto's hadden om als input te geven, maar dit hebben we dus niet kunnen testen.
Zoals eerder aangehaald liep het matchen van overeenkomende punten ook niet goed met meer dan 2 camera's.

Overzicht zip

Data/Graycodes: data gegeven bij de opdracht.
Result/: alle bekomen resultaten, inclusief de eigen dataset(s).
Src: Alle code files die nodig waren voor onze oplossingen.

Conclusie

Uiteindelijk hebben we het meeste van het project kunnen implementeren. We hebben alles van de basis buiten stap 4.2 (plane sweep) en de foto's voor stap 6.2

Groep 5
Hidde Brands (2158176)
Ruben Boone (2261602)
(projector gebruikt als camera). Er waren ook een paar kleine onjuistheden bij stap 6.1 bij de reconstructie. Ook hebben we 2 extra's geïmplementeerd. Buiten de paar voorvermelde problemen is het project over het algemeen vlot verlopen.