de resolutie van de beelden wordt gehalveerd. Dit omdat de stripes van de patronen breed genoeg zijn en het voldoende is om sparse dieptemappen te bekomen. Dit zorgt ervoor dat de complexiteit van de costmatrix van $O(\frac{MN}{4})$ en de complexiteit van het zoeken van het pad van $O(M+N)$ naar $O(\frac{M+N}{2})$.
In hoofdstuk 7 wordt er een overzicht gegeven van de impact van een aantal factoren op de snelheid van het algoritme.

Discussie

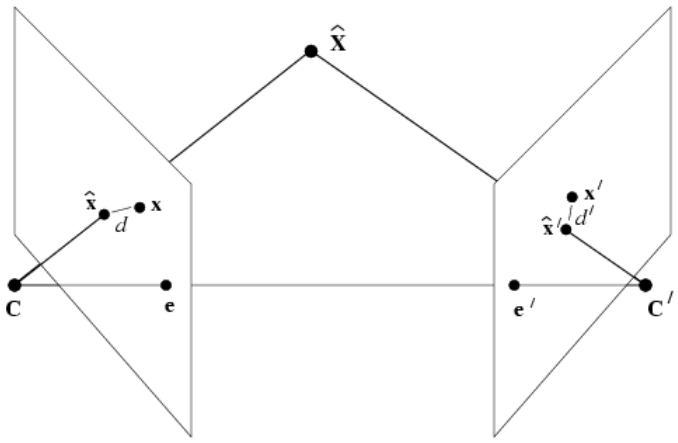
Dynamic programming kan niet goed overweg met discontinuïteiten in de scène. Ook in bovenstaande implementatie is er geen oplossing voor gegeven. Op plaatsen waar een abrupte overgang is van voorgrond naar achtergrond geeft het soms slechte resultaten waardoor de rand soms gaat samensmelten met de achtergrond. Het is mogelijk om het dynamic programming algoritme uit te breiden naar een multi-pass algoritme, zoals voorgesteld door *Zhang et al.* [89]. Deze gaat eerst optimale subpaden zoeken en deze vervolgens correct aan elkaar plakken.

Om het algoritme zo robuust mogelijk te maken is het aan te raden een soort van background substraction toe te passen door het zwart weg te filteren en deze punten niet te gebruiken. Hierdoor gaan enkel maar de pixels waarop een patroon geprojecteerd is worden gematcht. Het is dan ook aan te raden patronen te ontwikkelen zonder zwarte banden. Dit is in deze thesis niet het geval, waardoor er in de dieptemap voor bepaalde banden geen match is gevonden omdat deze punten weggefiterd zijn.

6.6 3D Reconstructie

In de 3D reconstructie stap wordt er getracht om vanuit de dispariteiten uit de vorige stap, 3D posities in wereldcoördinaten te berekenen. Dit is de zogenaamde triangularisatie. Vanuit de twee projectiematrices en de twee corresponderende 2D punten wordt een 3D punt geschat (zie Figuur 6.12). Doordat de backprojections van de twee punten niet ideaal snijden, wordt er gebruik gemaakt van SVD. SVD is een algoritme dat gebruikt kan worden om de meest ideale oplossing te verkrijgen.

Stel twee correspondentiepunten $(x_1, y_1, 1)$ en $(x_2, y_2, 1)$ die hetzelfde punt (X, Y, Z, W) in 3D beschrijven en hebben respectievelijk een projectiematrix P^1 en P^2. Backprojectie



Figuur 6.12: Triangularisatie is het zoeken van een 3D punt in wereldcoördinaten op basis van twee corresponderende 2D image punten (x en y in de Figuur). De backprojectie gaat nooit ideaal een snijpunt teruggeven omdat x en y niet accuraat genoeg kunnen worden berekend. Met behulp van SVD wordt de meest ideale 3D coördinaat berekend. Dit is $\hat{X}$. De triangularisatie error is gedefinieerd als het verschil in afstand van de projectie van $\hat{X}$ op de afbeeldingen. In de Figuur is deze aangegeven met d en d'.

gebeurt met de volgende vergelijkingen:

$$\begin{aligned}
P^1_{00}X + P^1_{01}Y + P^1_{02}Z + P^1_{03}W &= x_1; && (6.4)\\
P^1_{10}X + P^1_{11}Y + P^1_{12}Z + P^1_{13}W &= y_1; && (6.5)\\
P^1_{20}X + P^1_{21}Y + P^1_{22}Z + P^1_{23}W &= 1; && (6.6)\\
P^2_{00}X + P^2_{01}Y + P^2_{02}Z + P^2_{03}W &= x_2; && (6.7)\\
P^2_{10}X + P^2_{11}Y + P^2_{12}Z + P^2_{13}W &= y_2; && (6.8)\\
P^2_{20}X + P^2_{21}Y + P^2_{22}Z + P^2_{23}W &= 1. && (6.9)\\
& && (6.10)
\end{aligned}$$

Deze worden herschreven in de volgende vorm:

$$A \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = 0. \qquad (6.11)$$

Deze uitdrukking wordt herschreven in de volgende vier vergelijkingen:

$$\begin{aligned}
(6.4) &= [(6.4) - x_1(6.6)] && (6.12)\\
(6.5) &= [(6.5) - y_1(6.6)] && (6.13)\\
(6.7) &= [(6.7) - x_2(6.9)] && (6.14)\\
(6.8) &= [(6.8) - y_2(6.9)]\,, && (6.15)\\
& && (6.16)
\end{aligned}$$

ofnog

$$\begin{bmatrix} x_1P^1_{20} - P^1_{00} & x_1P^1_{21} - P^1_{01} & x_1P^1_{22} - P^1_{02} & x_1P^1_{23} - P^1_{03} \\ y_1P^1_{20} - P^1_{10} & y_1P^1_{21} - P^1_{11} & y_1P^1_{22} - P^1_{12} & y_1P^1_{23} - P^1_{13} \\ x_2P^1_{20} - P^2_{00} & x_2P^2_{21} - P^2_{01} & x_2P^2_{22} - P^2_{02} & x_2P^2_{23} - P^2_{03} \\ y_2P^1_{20} - P^2_{10} & y_2P^2_{21} - P^2_{11} & y_2P^2_{22} - P^2_{12} & y_2P^2_{23} - P^2_{13} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = 0. \qquad (6.17)$$

De matrix A gaat met behulp van Singular Value Decomposition (SVD) worden opgedeeld in UDV^t, waarbij U en V orthogonaal zijn; de kolommen van U de eigenvectoren zijn van AA^t; de kolommen van V de eigenvectoren zijn van A^tA en D een diagonaalmatrix met de eigenwaardes van A. De eigenvector met de kleinste eigenwaarde voor A^tA is de beste uitkomst voor de opgestelde vergelijkingen. Deze is de de laatste kolom van V. In het geval van triangularisatie, zal de eigenvector met kleinste eigenwaarde niet het exacte snijpunt zijn van de twee backprojecties, maar wel het midden zijn van het lijnstuk loodrecht op de twee backprojecties.

Voorbeeld

Gegeven een dieptemap zoals Figuur 6.13c. Elk punt in de dieptemap staat voor een dispariteit en is een verschil in coördinaat van twee punten op de twee beelden. Bij een gerectificeerd beeld gaat de x-coördinaat van beide punten overeenkomen en wordt de dispariteit weerspiegeld op de verandering in y-coördinaat. Deze twee punten, tesamen met de projectiematrices van beide camera's vormen de input voor het opstellen van de matrix A. Elk bekomen 3D punt wordt vervolgens gerenderd met OpenGL en krijgt de kleur mee van de pixel uit het camerabeeld. De output van dit algoritme resulteert in een 3D puntenwolk zoals weergegeven in Figuur 6.13. In Figuur 6.14 vindt u een uitgebreider voorbeeld van een scène waar een hoofd wordt bewogen. Indien op de puntenwolken een surface reconstruction algoritme wordt toegepast, bijvoorbeeld het *Ball-Pivoting Algorithm* van *Bernardini et al.* [8], worden deze puntenwolken omgezet in een mesh (zie Figuur 6.15). De rendering is uitgevoerd met *MeshLab* [47].

(a) (b) (c)

Figuur 6.13: Links: inputbeeld; Midden: dieptemap voor reconstructie; Rechts: 3D gereconstruuerde puntenwolk na triangularisatie.