Mesostructure from Specularity using a Raster Display and a Digital Camera

Jo Gielis

Promotor: Prof. dr. Philippe Bekaert
Advisor: Yannick Francken

Thesis voorgedragen tot het behalen van de graad van licentiaat in de informatica / doctorandus in de kennistechnologie, afstudeervariant informatica/multimedia.

Diepenbeek, May 23, 2007

The most beautiful experience we can have is the mysterious. It is the fundamental emotion that stands at the cradle of true art and true science. Whoever does not know it and can no longer wonder, no longer marvel, is as good as dead, and his eyes are dimmed.

Albert Einstein

Abstract

This thesis describes a method for efficient acquisition of fine-scale surface details, or *mesostructure* surfaces, where a single camera is used to capture the mesostructure illuminated by a raster display.

When adopting a typical camera-screen setup where a camera is placed on top of a raster display, a *camera calibration* step is inevitable. Camera calibration allows mapping from camera pixel coordinates to rays in the scene, and from points in the scene to camera pixel coordinates. To establish the relation between the screen and the camera, different screen calibration methods are discussed, using a planar or spherical mirror.

A concise overview of popular *shape recovery* techniques is presented, including shape from shading, photometric stereo and structured light. Because these techniques are usually inadequate to reconstruct highly specular fine-scale details, shape reconstruction using specularities is examined. Starting from the general shape reconstruction from specularities the more specialized case of mesostructure from specularity is discussed.

Mesostructure from specularity requires a high directional light resolution resulting in a high acquisition time. We propose a new *multiplexed illumination* scheme using *gray code patterns*. This technique exploits specularity properties, allowing for a fast and inexpensive shape recovery. The acquisition of N light sources is reduced to $O(\log_2 N)$. A raster display functions as a controllable planar illuminant which makes it easy to emit patterned illumination.

Further the implementation of the discussed methods is presented. Results obtained from our proposed multiplexed illumination scheme are thoroughly discussed, presenting both synthetic and real-world examples.

Acknowledgements

I would like to thank Philippe Bekaert for giving me the opportunity to work on this thesis. My advisor, Yannick Francken, for his guidance, stimulating support and proofreading and to whom I could come any time with all my questions, problems and achievements.

Special thanks go to my friend Johan Huysmans for some very valuable remarks, motivation skills and detailed proofreading and to my friend Nicolas Barbier for proofreading.

Special thanks also go to my girlfriend Katrijn for motivating me and for her loving support, my parents for giving me the opportunity to study and all my friends and family who, in one way or another, stimulated me to finish this thesis.

Nederlandse samenvatting

Het ultieme doel van computer visie is om het menselijk zicht te kunnen simuleren met een computer, dit betekent reageren op visuele invoer en op basis van deze invoer acties uitvoeren [GW02].

Een deel van dit doel bestaat uit het *scannen* van reële objecten, om zo informatie over de vorm en kleur van het object te verkrijgen. Deze informatie kan dan gebruikt worden om een drie dimensionale (3D), digitale voorstelling van het object te construeren. Deze techniek wordt *3D-scanning* genoemd en kent vele toepassingen, zoals de filmindustrie, de reclamewereld, de medische wereld, computerspellen, enzovoorts. In deze thesis bestuderen we het *3D-scannen* van mesostructuur oppervlakten gebruik makend van een camera-scherm opstelling, waarbij de camera geplaatst wordt op het scherm. Een mesostructuur oppervlakte is een oppervlakte die zeer gedetailleerd is maar waarvan de structuur toch nog door het blote oog kan worden onderscheiden. Voorbeelden hiervan zijn o.a. de schil van een sinaasappel, een muntstuk, leder, enzovoorts. Het scannen van een mesostructuur oppervlakte m.b.v. een enkele camera en een scherm kan worden opgedeeld in drie deelproblemen.

Camera calibratie is het eerste deelprobleem. De functie hiervan is zorgen voor de transformatie van pixel coördinaten op het beeldvlak van de camera naar stralen in de scène, en van punten in de scène naar pixel coördinaten op het beeldvlak van de camera. Deze transformatie kan worden voorgesteld door een matrix, genaamd de *camera matrix*. Om deze matrix op te stellen moeten de parameters van de camera gevonden worden. Deze parameters bestaan uit *interne* en *externe* parameters. De interne parameters stellen de interne oriëntatie van de camera voor:

- De brandpuntsafstand (*focal length*), dit is de afstand van het brandpunt tot het beeldvlak van de camera.
- Het hoofdpunt (*principal point*), dit zijn de pixel coördinaten van het punt waar de diepte-as en het beeldvlak intersecteren.

De externe parameters of de externe oriëntatie van de camera bestaan uit rotatie- en translatieparameters. Deze relateren het *wereld coördinatensysteem* aan het *camera coördinatensysteem*. Om deze camera parameters te berekenen, wordt gebruik gemaakt van *calibratie patronen*. Een calibratie patroon is typisch een zwart-wit schaakbord patroon of een patroon bestaande uit stippen met gelijke grootte en gelijke tussenafstanden. Vervolgens worden een aantal foto's genomen van het calibratie patroon, telkens met een andere oriëntatie

van het calibratie patroon t.o.v. de camera. Gegeven de grootte van de vierkanten (of stippen) en de tussenafstanden kunnen, m.b.v. de vervormde patronen waargenomen op de foto's, de camera parameters berekend worden. De bekomen interne parameters stellen de interne oriëntatie van de camera voor en zijn dus camera specifiek. De externe parameters beschrijven de externe oriëntatie, welke verschillend zijn voor elk van de invoerfoto's. Met andere woorden, de locatie van een object t.o.v. de camera is bepaald door deze externe camera parameters. Er zijn allerhande hulpmiddelen beschikbaar die via deze methode de parameters kunnen berekenen [Lab, JY, Sto]. Dit betekent dat we de locatie van een reëel object t.o.v. de camera kunnen bepalen door een calibratie patroon op het object te plaatsen. Wanneer de locatie van het scherm t.o.v. de camera bepaald moet worden is dit echter niet mogelijk aangezien het scherm niet in het gezichtsveld van de camera ligt. De camera en het scherm zijn namelijk beiden gericht naar de scène. Er worden twee methodes besproken om dit probleem op te lossen.

Als eerste methode wordt scherm calibratie m.b.v. een vlakke spiegel voorgesteld. Er wordt een calibratie patroon op de spiegel geplaatst en ook op het scherm wordt een calibratie patroon getoond. Vervolgens wordt de spiegel zo geplaatst zodat zowel het volledige calibratie patroon op de spiegel als het volledige calibratie patroon op het scherm direkt zichtbaar zijn voor de camera. Eens de juiste positie van de spiegel is bepaald, wordt er een foto genomen van de spiegel. Aangezien deze foto beide patronen bevat, is de volgende stap het scheiden van deze patronen. Dit kan door een markering tussen beide patronen te plaatsen tijdens de opname. Gesteld dat de interne parameters van de camera reeds gekend zijn, kunnen de externe parameters geassocieerd met elk van de patronen bepaald worden. Merk op dat de externe parameters geassocieerd met het calibratie patroon op scherm de locatie van de *gespiegelde* versie van het scherm bepalen. De originele scherm positie kan worden gevonden door te spiegelen over het vlak bepaald door de spiegel [Fun05, FY07].

Een tweede optie is scherm calibratie m.b.v. een bolvormige spiegel [FHB07, TLGS05]. Het grote van deze methode is dat ze grotendeels geautomatiseerd is. De bolvormige spiegel wordt in het gezichtsveld van de camera geplaatst waardoor de camera een gespiegelde versie van het scherm te zien krijgt. Het scherm dient een enkele kleur uit te zenden. Vervolgens worden een aantal foto's genomen van de bolvormige spiegel telkens met de spiegel op een verschillende locatie. In elke foto wordt de bolvormige spiegel gelocaliseerd. Dit gebeurt door eerst een achtergrond substractie stap uit te voeren gevolgd door een aantal morfologische operaties. De omtreklijn van de spiegel, een ellips, wordt gedetecteerd m.b.v. een aangepaste RANSAC [FB81] benadering. De volgende stap is het bepalen van de coördinaten van de omtreklijn t.o.v. de camera. Er wordt een *kegel* gevormd met als top de camera en met als grondvlak de omtreklijn van de bolvormige spiegel. Gesteld dat de interne parameters van de camera en de straal van de bolvormige spiegel gekend zijn, kan de afstand camera-spiegel (afstand top-grondvlak in de kegel) en het centrum van bolvormige spiegel berekend worden. De locatie van de spiegel t.o.v. de camera is nu bepaald. Er wordt nu gezocht achter het scherm binnen de omtreklijn van de spiegel, dit gebeurt opnieuw door een reeks van morfologische operaties. Nadat het scherm gelocaliseerd is, wordt er gezocht achter de hoekpunten. Francken et al. [FHB07] stellen hier voor om de omtreklijn van het scherm te projecten naar een beter gepast coördinatensysteem, waardoor het detectie probleem sterk vereenvoudigd wordt. Eens de hoekpunten zijn gedetecteerd, kunnen de reflectiestralen berekend worden. Tenslotte kan de 3D positie van het scherm berekend worden door de verzameling van reflectiestralen per hoekpunt, gegeven door de verschillende invoer foto's, te beschouwen als een kleinste-kwadraten probleem.

Een tweede deelprobleem is *vorm reconstructie*, het bepalen van de vorm van een object gegeven één of meer foto's van het object. Er wordt een overzicht gegeven van een aantal populaire vorm reconstructie technieken zoals *shape from shading* [Hor70, ZTCS99], *photometric stereo* [Woo78, Woo89] en *structured light* [J$^+$01]:

Shape from shading heeft als invoer slechts één afbeelding nodig. De vorm wordt gereconstrueerd door analyse van graduele veranderingen van shaduw. Er worden twee aannames gemaakt, namelijk dat de lichtbron positie gekend is en dat het object in kwestie voldoet aan het Lambertiaanse reflectiemodel. Er kan ook gebruik gemaakt worden van technieken die een afschatting maken van de lichtbron positie indien deze niet gekend is [ZC91, LR85, PEN82].

Photometric stereo gebruikt meerdere foto's als invoer, deze foto's zijn opgenomen vanuit hetzelfde gezichtsveld maar met een verschillende belichtingsrichting. Vervolgens wordt de vlak oriëntatie gekoppeld aan de radiantie voor elke pixel. De verzameling van deze koppels wordt een *reflectiemap* genoemd. Voor elke foto is het mogelijk om een reflectiemap te berekenen. Met deze verzameling van reflectiemappen kan de normaal per pixel berekend worden. Er worden twee aannames gedaan, namelijk dat de lichtbron positie gekend is en dat het object in kwestie voldoet aan het Lambertiaanse reflectiemodel.

Structured light projecteert een aantal gekende lichtpatronen op de te meten scène. De scene wordt vervolgens vastgelegd door één enkele camera. De nodige 3D-informatie kan bemachtigd worden door de vervormingen van de lichtpatronen op de scène te analyseren.

Deze technieken hebben echter problemen wanneer ze te maken krijgen met sterk reflecterende objecten, omdat het actieve signaal (bijvoorbeeld het geprojecteerde lichtpatroon bij *structured light*) hoofdzakelijk gereflecteerd wordt langs de speculaire richting zodanig dat bijna geen licht de camera bereikt.

Fleming et al. [FTA04] laten zien dat wel degelijk 3D-informatie geobserveerd kan worden, gegeven een perfect speculair gerendered object. Er wordt een overzicht gepresenteerd van onderzoek omtrent vorm reconstructie van sterk reflecterende objecten [ZFA97, ZM00, SP01, SP02, BS03, TLGS05]. Meest recent is het werk van Chen et al. [CGS06] waarin mesostructuur oppervlakten worden gereconstrueerd door gebruik te maken van speculaire glanspunten, ook wel *mesostructure from specularity* genoemd. De wet van reflectie zegt dat een speculair glanspunt uniek bepaald is door de inkomende lichtrichting, de oppervlakte normaal en de kijkrichting. Dit betekent dat de oppervlakte normaal berekend kan worden indien lichtrichting en kijkrichting gegeven zijn. *Mesostructure from specularity* buit deze observatie uit als volgt: er wordt een reeks foto's genomen telkens met dezelfde kijkrichting (camera is statonair) maar met steeds een verschillende lichtrichting. Voor elke foto wordt een speculaire reflectiemap berekend door elke pixelintensiteit te vergelijken met een drempelwaarde. Indien de pixelintensiteit groter is dan de drempelwaarde dan is er een speculaire reflectie aanwezig. Aangezien voor elke foto de lichtbron en camera positie gekend zijn, kan de oppervlakte normaal voor elke pixel, waar een speculaire reflectie gedetecteerd is, berekend worden. Hoe meer foto's met verschillende lichtrichting, hoe meer speculaire glanspunten gedetecteerd kunnen worden en dus ook hoe meer normalen gevonden worden. Deze techniek is uiterst geschikt voor het reconstrueren van speculaire mesostructuur oppervlakten. In tegenstelling tot voorgaande technieken is *mesostructure from specularity* niet onderhevig aan *subsurface scattering* (bij *doorzichtige* materialen waar onder het *initiële* oppervlakte nog meer reflecties plaatsvinden). Merk wel op dat er (extreme) oppervlakte normalen zijn die niet gedetecteerd kunnen worden doordat de oppervlakte het licht zelf blokkeert, ongeacht het bereik van de lichtbron.

Het laatste deelprobleem dat deze thesis beschouwt, is *gecontroleerde belichting*. Er wordt gebruik gemaakt van een raster scherm om de scène te belichten. De meest voorkomende raster schermen zijn CRT en LCD schermen. Er is een belangrijk verschil tussen beide dat zeker dient opgemerkt te worden. LCD schermen lijken helderder als deze worden bekeken vanuit het centrum van het scherm, dan als ze worden bekeken vanuit een bepaalde hoek. Dit effect wordt verklaard door de constructie van LCD schermen. CRT schermen zijn niet onderhevig aan dit effect. Een ander belangrijk effect dat geldt voor beide soorten schermen is dat de schermen altijd licht uitzenden, ook als het scherm enkel *zwarte pixels* toont. Dit kan in rekening gebracht worden door een opname te maken van de scène belicht met enkel zwarte pixels en deze aftrekken van de opname van de scène belicht door de gewenste pixels.

Wanneer speculaire reflecties worden gebruikt voor vorm reconstructie is het belangrijk dat er voldoende opnames zijn vanuit verschillende lichtrichtingen. Dit heeft uiteraard tot gevolg dat de opnametijd hoog kan oplopen. Om de opnametijd te verminderen, stellen wij een aangepast gemultiplext belichtingsschema voor. Merk op dat de licht*richtingen* worden gemultiplext. Een typisch gemultiplext belichtingsschema gaat meerdere lichtbronnen tegelijk gebruiken per opname zonder dat de elementen (reflecties, shaduwen, etc.) die worden verkregen bij het belichten met een enkele bron verloren gaan. Dit betekent echter niet dat er minder opnames nodig zijn. Ons voorstel is om gebruik te maken van gecodeerde patronen waardoor de opnametijd wel verminderd kan worden. Deze techniek laat toe N lichtbronnen te simuleren door $O(\log_2 N)$ patronen waardoor de opnametijd drastisch verminderd wordt. Door elke pixel waar een speculair glanspunt is waargenomen een codewoord toe te kennen, kan de bijhorende individuele lichtbron makkelijk achterhaald worden. Het coderen van de pixels gebeurt als volgt:

- Elke individuele lichtbron krijgt een index toegewezen, dit kan met $\log_2 N$ bits.
- Er worden $\log_2 N$ patronen getoond. Deze patronen worden gevormd door de indices van de individuele lichtbronnen tegelijk door te lopen. De indices van de lichbronnen

worden binair voorgesteld en worden bit voor bit doorlopen, startend bij de MSB (meest significante bit). Een patroon wordt dus voorgesteld door N bits, voor elke bit gelijk aan 0 wordt de lichtbron in kwestie uitgezet, voor elke bit gelijk aan 1 wordt de lichtbron in kwestie aangezet. Merk op dat de eerste lichtbron geïndexeerd wordt met 0 als al de lichtbronnen met $\log_2 N$ bits geïndexeerd worden. Dit betekent dat voor elk patroon deze lichtbron een 0-bit zal hebben, dus nooit aangezet zal worden. Om dit op te lossen worden $\log_2 N + 1$ bits gebruikt om de lichtbronnen te indexeren. Aan elke lichtbron index wordt een bit toegevoegd die steeds op 1 staat, deze bit wordt de MSB. Het eerste patroon zal dus alle lichtbronnen aanzetten.

- Elke pixel krijgt N bits toegewezen. Bij elk patroon wordt gekeken of er speculaire glanspunten aanwezig zijn. Indien een glanspunt op een pixel wordt gedetecteerd, wordt voor die pixel, de bit op de plaats gelijk aan de index van het patroon gelijk gezet aan 1.
- Als alle patronen doorlopen zijn, zal voor elke pixel waarop een glanspunt gedetecteerd is, het codewoord verwijzen naar de index van de individuele lichtbron.

Voor de implementatie wordt er gebruik gemaakt van twee verschillende camera-scherm opstellingen. Bij de eerste opstelling wordt de camera boven, onder of langs het scherm geplaatst. Deze opstelling is in bijna elke huiskamer terug te vinden, ook laptops maken gebruik van deze opstelling. Het mesostructuur oppervlakte wordt voor scherm en camera geplaatst. De tweede opstelling die gebruikt wordt, is een vereenvoudigde opstelling. In deze opstelling wordt er gebruik gemaakt van een tweede scherm. De camera wordt tussen de twee schermen geplaatst zodanig dat camera en schermen *parallel* staan t.o.v. elkaar. Het mesostructuur oppervlakte wordt ook parallel voor camera en schermen geplaatst. Deze opstelling heeft het grote voordeel dat de locaties van schermen en mesostructuur t.o.v. de camera manueel bepaald kunnen worden en calibratiestap vermeden kan worden.

Om het calibreren van de camera te vergemakkelijken wordt er gebruik gemaakt van *tclcalib* [Lab], een verzameling van hulpmiddelen om camera calibratie te vereenvoudigen, en is volledig geïntegreerd in de implementatie. *Tclcalib* geeft zowel de interne als externe parameters terug, welke gebruikt worden om de camera matrix te construeren. De calibratie van het scherm wordt gedaan m.b.v. een vlakke spiegel. Hiervoor wordt er een patroon op de spiegel geplaatst en ook wordt er een patroon op het scherm getoond. De spiegel wordt vervolgens zo geplaatst zodat beide patronen in het gezichtsveld van de camera liggen, welke hiervan een foto neemt. De volgende stap is het scheiden van beide patronen. Dit wordt gedaan door twee rode markeringen te plaatsen tussen de twee patronen voor de opname. Deze markeringen worden gedetecteerd d.m.v. een *thresholding* stap. Eens de markeringen gedetecteerd zijn, worden ze verbonden en kunnen de patronen gescheiden worden. De gescheiden patronen kunnen nu gebruikt worden om de locatie van spiegel en een *gespiegelde* versie van het scherm t.o.v. de camera te berekenen. Om de echte locatie van het scherm te vinden, wordt de gespiegelde versie gespiegeld over het vlak bepaald door de vlakke spiegel.

Gemultiplexte belichtingsmodellen worden doorheen de volledige implementatie gebruikt. Deze modellen zijn nodig om de lichtpatronen te genereren, voor het decoderen van foto's die opgenomen zijn met lichtpatronen en voor het bepalen van de posities van de individuele lichtbronnen (pixel of verzameling van pixels) op het scherm. De modellen zijn gebaseerd op het *algemeen* gemultiplexte belichtingsprincipe. Door de modellen algemeen te definieren kunnen meerdere modellen ondersteund worden. Momenteel wordt het door ons geïntroduceerde model ondersteund, alsook het Hadamard-gebaseerde model van Schechner et al. [SNB03].

De implementatie laat toe een normaalmap te berekenen gegeven een mesostructuur oppervlakte. Dit kan gebeuren op twee manieren. Voor de eerste methode zijn er N invoerfoto's nodig, met N het aantal lichtbronnen. Deze invoerfoto's kunnen ofwel rechtstreeks opgenomen worden ofwel *gedecodeerd* worden uit de foto's opgenomen m.b.v. de lichtpatronen. Voor het decoderen wordt gebruik gemaakt van de *demultiplex matrix*, deze is gegeven door het gekozen gemultiplext belichtingsmodel. Een demultiplex matrix definieert de relatie tussen de foto's opgenomen met de individuele lichtbronnen en de foto's opgenomen met de lichtpatronen. Gegeven deze matrix en de foto's, opgenomen met de lichtpatronen, kunnen schattingen van de foto's, opgenomen met de individuele lichtbronnen, bekomen worden. Als eens de invoerfoto's opgenomen of berekend zijn, worden deze gecontroleerd op speculaire reflecties. Dit gebeurt m.b.v. een *thresholding* stap. Vervolgens kunnen de normalen berekend worden voor de pixels waar een speculaire reflectie gevonden is. We nemen aan dat een calibratiestap is uitgevoerd en de locatie van scherm, en dus ook de individuele lichtbronnen, en mesostructuur oppervlakte t.o.v. de camera gekend zijn.

De tweede manier maakt gebruik van pixel codificatie. Dit algoritme heeft enkel de foto's nodig opgenomen met de lichtpatronen. Elke invoerfoto wordt geanalyseerd en de pixels krijgen een code toegekend. M.b.v. deze code kan de lichtbron, die de speculaire reflectie op de pixel in kwestie veroorzaakte, gevonden worden. Dit algoritme is dus in staat een normaalmap te berekenen in slecht $O(\log_2 N)$ stappen i.p.v. N stappen, met N het aantal individuele lichtbronnen.

Als laatste vermelden we dat de implementatie in staat is synthetische beelden te genereren, gegeven een normaalmap. Zowel beelden belicht met de inviduele lichtbronnen als beelden belicht met de lichpatronen kunnen worden gegenereerd. Om de intensiteit per pixel te berekenen, wordt gebruik gemaakt van het Phong model voor speculaire reflectie [Pho73].

We hebben in deze thesis de reconstructie van mesostructuur oppervlakten m.b.v. speculaire reflecties onderzocht. Bij de camera-scherm opstelling waar de camera boven, onder of langs het scherm geplaatst wordt, is een calibratiestap noodzakelijk. Uit onze experimenten is gebleken dat de geïmplementeerde calibratiestap niet robuust genoeg is, als gevolg is er een vereenvoudigde opstelling gebruikt. Deze maakt gebruik van twee, langs elkaar geplaatste, LCD schermen waarbij de camera tussen deze twee schermen wordt geplaatst. Deze opstelling heeft naast haar geometrisch voordeel ook nog het voordeel dat het bereik van de mogelijke normalen die verkregen kunnen worden, groter is. Desondanks zijn er extreme oppervlakte normalen waarbij het licht door de oppervlakte zelf geblokkeerd wordt en dus niet verkregen kunnen worden ongeacht het bereik van de lichtbron.

Verder werd het door ons geïntroduceerde gemultiplexte belichtingsmodel uitvoerig getest, zowel op synthetische als reële objecten. Deze techniek is momenteel beperkt tot oppervlakten met spiegelachtige reflecties. Bij glanzende reflecties zijn de speculaire glanspunten meer uitgesmeerd waardoor de unieke mapping tussen pixel en lichtbron beïnvloed wordt, wat tot dubbelzinnige informatie kan leiden. Desondanks zijn de door ons verkregen resultaten zeer positief, zowel bij de synthetische als bij de reële geteste mesostructuur oppervlakten. Ook wordt de berekeningstijd drastisch verminderd, N lichtbronnen worden gesimuleerd door slechts $O(\log_2 N)$ patronen.

Uiteraard is er plaats voor verbetering. Als alternatief voor de minder robuuste scherm calibratie met vlakke spiegel, stellen we voor om experimenten uit te voeren gebruik makend van scherm calibratie m.b.v. een bolvormige spiegel. Deze techniek vereist minder interactie met de gebruiker waardoor een meer robuuste oplossing aangeboden kan worden.

Om de resultaten te verbeteren bekomen met de huidige vorm reconstructie techniek, kan deze techniek uitgebreid worden door een *smoothness* restrictie op te leggen. Het gebruik van deze restrictie zal meer uniforme resultaten opleveren.

Het gebruik van een scherm, een vlakke lichtbron, voor belichting zal altijd het bereik van mogelijke normalen die verkregen kunnen worden beperken. Een halfronde lichtbron zou gebruikt kunnen worden om dit bereik te vergroten. Desondanks zullen er echter steeds extreme normalen zijn die niet gevonden kunnen worden ongeacht het bereik. Een mogelijke oplossing is het gebruik van een combinatie van vorm reconstructie technieken. Vorm reconstructie m.b.v. speculaire reflecties kan dan gebruikt worden voor de pixels waar een speculair glanspunt gevonden wordt en vorm m.b.v. *photometric stereo* kan dan gebruikt worden voor de pixels die beneden de drempelwaarde vallen voor speculaire glanspunten.

Contents

List of Figures

List of Tables

1
Introduction

The ultimate goal of computer vision is to use computers to emulate human vision, including learning and being able to make inferences and take actions based on visual inputs [GW02].

A part of this goal is to gather information about the shape and the color of real-world objects. The information can then be used to reconstruct the real-world object into a three dimensional, digital form. This process is called *3D-scanning* and has a wide variety of applications, i.e. the movie industry (special effects, etc.), advertisements, textile, the medical world, computer games, prototyping and more [Eye].

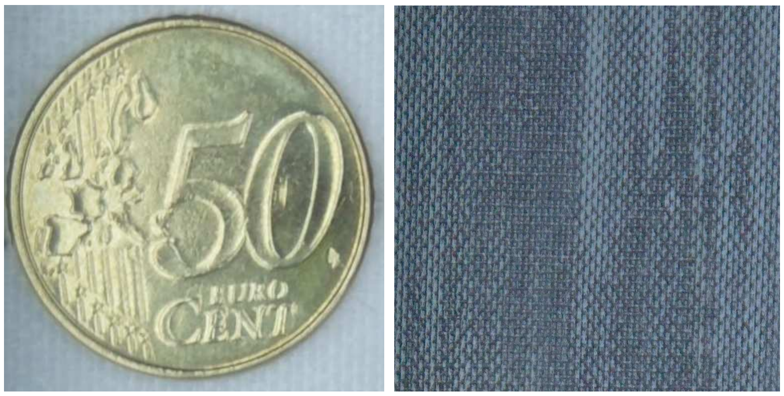
This thesis will focus on gathering information about the shape of fine-scale surface details, or *mesostructure* surfaces, using a single digital camera to capture and a raster display to illuminate the mesostructure surface.

1.1 Mesostructure Surfaces

A mesostructure surface has geometric details that are relatively small but still individually visible such as bumps or dents. It is one of the key components of three dimensional texture and contributes strongly to the complex surface appearance of real-world objects [CGS06]. Some example mesostructure surfaces are shown in figure 1.1.

1.2 Problem Statement

To present the problem statement, the keywords in the title of this thesis are explained.



(a) (b)

Figure 1.1: Examples of two mesostructure surfaces: (a) a coin and (b) tape.

1.2.1 Mesostructure from Specularity

Recovery of fine-scale details is still a difficult problem, especially for highly specular or translucent mesostructure surfaces. Recently Chen et al. [CGS06] proposed the use of specularities to reconstruct mesostructure surfaces, based upon their work we will present a technique for mesostructure surface acquisition.

1.2.2 Raster Display

A raster display, i.e. a CRT (Cathode Ray Tube) monitor or a LCD (Liquid Crystal Display) screen will function as a controllable planar illuminant [FY07]. Each pixel or group of pixels on the raster display will function as a single controllable light source.

To take full advantage of this controllable illuminant we propose a new multiplexed illumination scheme using gray-code patterns. This illumination scheme exploits specularity properties, allowing for a fast and inexpensive shape recovery. The raster display will make it easy to emit patterned illumination.

1.2.3 Digital Camera

A single digital camera will be used to capture the mesostructure surface. A camera calibration step will be needed to acquire the positions and orientations of the raster display and the mesostructure surface with respect to the camera. The adopted camera-screen setup is shown in figure 1.2.

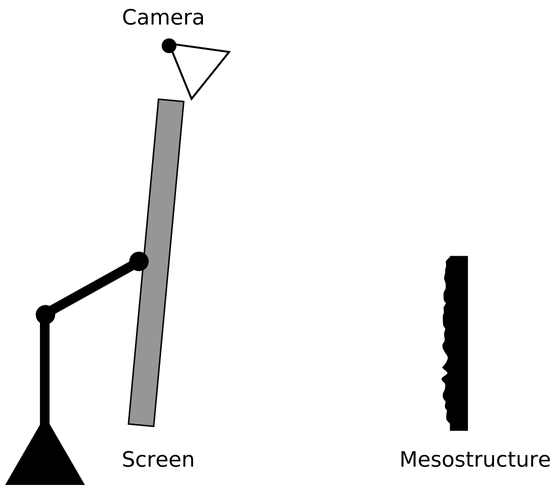


Figure 1.2: Camera-screen setup

1.3 Applications

Most mesostructure acquisition systems are complex systems and are rarely able to capture fine-scale details of real-world objects with translucency or highly specular reflection. In contrast this thesis presents a simple mesostructure acquisition system using an inexpensive setup that is able to reconstruct translucent or highly specular mesostructure surfaces like human skin, leather, coins, etc. Even an average desktop camera-screen setup can be used to scan fine-scale surface details.

1.4 Thesis Outline

A summary of the subjects that will be discussed in each chapter.

- **Chapter 2** discusses calibration. It starts with camera calibration, which is required to map camera pixel coordinates to rays in the scene, and points in the scene to camera pixel coordinates. In addition, to establish the relation between the screen and the camera, different screen calibration methods will be discussed, using a planar or spherical mirror.
- **Chapter 3** presents an overview of popular shape recovery methods. The focus lies on shape recovery using specularities. Starting from the general shape reconstruction from specularities the more specialized case of mesostructure from specularity will be discussed.
- **Chapter 4** proposes a new multiplexed illumination scheme using gray-code patterns. This technique exploits specularity properties, allowing for a fast and inexpensive shape recovery. The acquisition of N light sources is reduced to $O(\log_2 N)$.

The advantages and disadvantages of using a raster display as a controllable planar illuminant will be discussed too in this chapter.

- **Chapter 5** gives an overview of the implementation.
- **Chapter 6** presents the results from our implementation, along with a discussion.
- **Chapter 7** summarizes the achieved results.
- **Chapter 8** gives an outline for possible future.

2 Calibration

Camera calibration consists of two processes, *geometric* calibration and *radiometric* calibration. Geometric calibration is the process of determining the geometric deformation of the image. This process allows mapping from image pixel coordinates to rays in the scene, and from points in the scene to image pixel coordinates. Radiometric calibration relates the pixel intensity values to the irradiance on the image sensor. This chapter will focus on geometric calibration. Radiometric calibration is not further discussed, the interested reader is referred to the work of Funk et al. [Fun05]. Throughout this text the use of the term (camera) calibration will refer to geometric calibration exclusively.

The aforementioned mapping can be expressed as a matrix. This matrix is referred to as the *camera matrix*. The first part of this chapter will discuss the construction of the camera matrix. The second part of this chapter presents the problem and possible solutions of locating the screen with respect to the camera.

2.1 Camera Calibration

Camera calibration is required to allow mapping from camera pixel coordinates to rays in the scene, and points in the scene to camera pixel coordinates. This mapping or transformation can be expressed as a matrix, which we refer to as the *camera matrix*. Based on the work of Hartley and Zisserman [HZ04] it is shown how to construct the camera matrix.

2.1.1 Camera Matrix

The camera matrix will be constructed step by step. To start a point in space (a point from the scene) is projected to the image plane of the camera (Figure 2.1). For now central

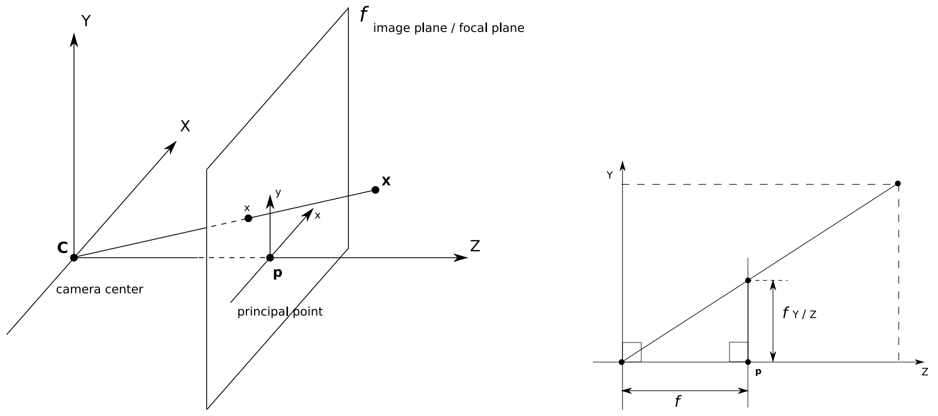


Figure 2.1: C is the camera center and p the principal point, which is the intersection between the principal axis (Z) and the image plane (f).

projection is assumed. Let the center of projection be the origin of a Euclidean coordinate system and consider the plane f, which is called the *image plane* or *focal plane* of the camera.

Note: *For now we will make the assumption that both the points on the image plane and the points in space are expressed in the same Euclidean coordinate frame, the camera coordinate frame.*

The projection of a point $\mathbf{X} = (X, Y, Z)^T$ in space on the image plane (Figure 2.1) is a mapping from Euclidean 3-space $\mathbb{R}^3$ to Euclidean 2-space $\mathbb{R}^2$:

$$\mathbb{R}^3 \mapsto \mathbb{R}^2 : \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{pmatrix}. \tag{2.1}$$

The center of projection is the *camera center* (C). The ray from the camera center perpendicular to the image plane is called the *principal axis* or *principal ray* of the camera. The intersection between the principal axis (Z) and the image plane f is called the principal point (p).

Using homogeneous coordinates, the central projection can be expressed as a lineair mapping. This means equation (2.1) can be rewritten in terms of a matrix multiplication:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.2}$$

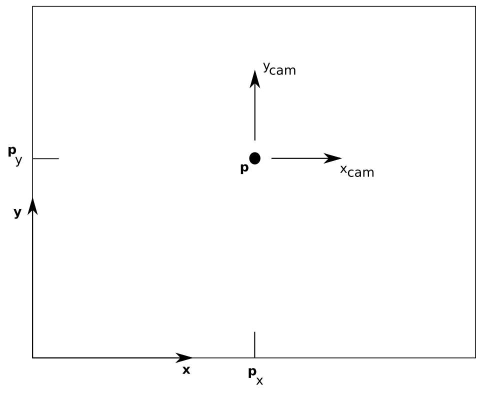


Figure 2.2: Mapping of the principal point to the origin.

where a point in space $\mathbf{X}$ is represented by the homogeneous vector $(X, Y, Z, 1)^T \in \mathbb{R}^4$ and an image point $\mathbf{x}$ is represented by the homogeneous vector $(fX, fY, Z)^T \in \mathbb{R}^3$.

We will now introduce $P \in \mathbb{R}^{3\times4}$, the homogeneous *camera projection matrix*, whereby equation (2.2) can be rewritten as:

$$\mathbf{x} = P\mathbf{X} \tag{2.3}$$

Until now the origin of the image plane was assumed to be at the principal point. This is not necessarily true and thus has to be taken into account. Equation (2.1) becomes:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} f\frac{X}{Z} + p_x \\ f\frac{Y}{Z} + p_y \end{pmatrix} \tag{2.4}$$

with $(p_x, p_y)^T$ are the coordinates of the principal point (Figure 2.2). When written using homogeneous coordinates this becomes:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.5}$$

Introducing $K \in \mathbb{R}^{3\times3}$, the *camera calibration matrix*:

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}, \tag{2.6}$$

equation (2.5) can be written simply as:

$$\mathbf{x} = K \left[\ I \quad | \quad 0\ \right] \mathbf{X}_{\text{cam}} \tag{2.7}$$

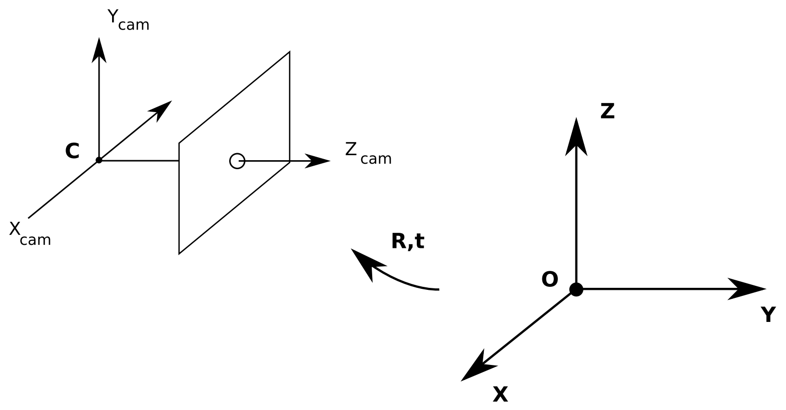


Figure 2.3: The Euclidean transformation between the world and camera coordinate frames.

where $K \in \mathbb{R}^{3\times3}$ is the camera calibration matrix and $[I \mid 0]$ represents a matrix divided up into an identity matrix $I \in \mathbb{R}^{3\times3}$ plus a column vector, here the zero vector ($\in \mathbb{R}^3$). $(X, Y, Z, 1)^T$ is written as $\mathbf{X}_{\text{cam}}$ to emphasize that the camera is assumed to be located at the origin of a Euclidean coordinate system with the principal axis of the camera pointing straight down the Z-axis, and the point $\mathbf{X}_{\text{cam}}$ is expressed in this coordinate system. This coordinate system is referred to as the *camera coordinate frame*.

As noted earlier, the assumption was made that both the points on the image plane and the points in space are expressed in the same Euclidean coordinate frame. In general, points in space will be expressed in terms of a different Euclidean coordinate frame, the *world coordinate frame*. The camera coordinate frame and the world coordinate frame are related through a rotation and a translation (Figure 2.3). Let $\widetilde{\mathbf{X}}$ be a non-homogeneous 3-vector representing the coordinates of a point in the world coordinate frame, and $\widetilde{\mathbf{X}}_{\text{cam}}$ the same point in the camera coordinate frame. $\widetilde{\mathbf{X}}_{\text{cam}}$ can be written as $R\left(\widetilde{X} - \widetilde{C}\right)$ where $\widetilde{C}$ represents the coordinates of the camera center in the world coordinate frame, and R is a 3×3 rotation matrix representing the orientation of the camera coordinate frame. Using homogeneous coordinates this becomes:

$$\mathbf{X}_{\text{cam}} = \begin{bmatrix} R & -R\widetilde{C} \\ \mathbf{0} & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\widetilde{C} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{X} \tag{2.8}$$

using equation (2.6) this results to:

$$\mathbf{x} = KR\left[\, I \mid -\widetilde{C} \,\right]\mathbf{X} \tag{2.9}$$

The parameters contained in K are called the *internal* or *intrinsic parameters*, or the *internal orientation* of the camera. The parameters of R and $\widetilde{C}$, which relate the camera coordinate frame to the world coordinate frame are called the *external* or *extrinsic parameters* or the *external orientation*.

The derived model assumes that the image coordinates are Euclidean coordinates having equal scales in both axial directions. This model is referred to as the *pinhole camera model*. In the case of CCD (*Charge-Coupled Device* or *Color-Capture Device*) cameras there is the possibility of having pixels with unequal scales. In case the image coordinates are measured in pixels, this has the effect of introducing unequal scale factors in each direction. To take this into account the camera calibration matrix K is adapated to

$$K = \begin{bmatrix} \alpha_x & & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{2.10}$$

with α_x equal to fm_x and α_y equal to fm_y. m_x and m_y are the number of pixels per unit distance in image coordinates.

If we represent the world to image transformation as $\widetilde{\mathbf{X}}_{cam} = R\widetilde{\mathbf{X}} + \mathbf{t}$, the *camera matrix* becomes

$$P = K\left[\begin{array}{c|c} R & \mathbf{t} \end{array}\right] \tag{2.11}$$

where $\mathbf{t} = -R\widetilde{C}$.

We have shown how to construct the camera matrix with the internal and external parameters of the camera. These parameters can be computed with the use of *calibration patterns*, usually checkerboard patterns or dotted patterns. A number of pictures of a pattern are taken with the camera, changing the orientation of the pattern for each picture. Assuming the number of squares (or dots), the size and distance between the squares (or dots) are known, the deformations of the captured patterns can be analyzed and used to retrieve the parameters. The obtained internal parameters describe the internal orientation of the camera. The external parameters describe the external orientation, from this follows that each picture yields different external parameters. In other words, the location of a real-world object with respect to the camera is defined by these external parameters.

Various tools are available that are able to compute these parameters with the use of *calibration patterns*. *tclcalib* is a collection of various tools for camera calibration, uses dotted calibration patterns and is maintained by the Stanford Computer Graphics Laboratory [Lab]. Bouguet created a calibration toolbox for Matlab and uses checkerboard calibration patterns [JY]. Stoyanov created an application for calibration based on Bouguet's toolbox [Sto].

2.2 Screen Calibration

Since a raster display will be used to illuminate the scene and a single camera to capture the scene, they will be both directed to the scene. The result being that the display will be out of the field of view of the camera. This poses a problem when computing the mapping from screen to camera. This mapping is needed to determine the location of the light

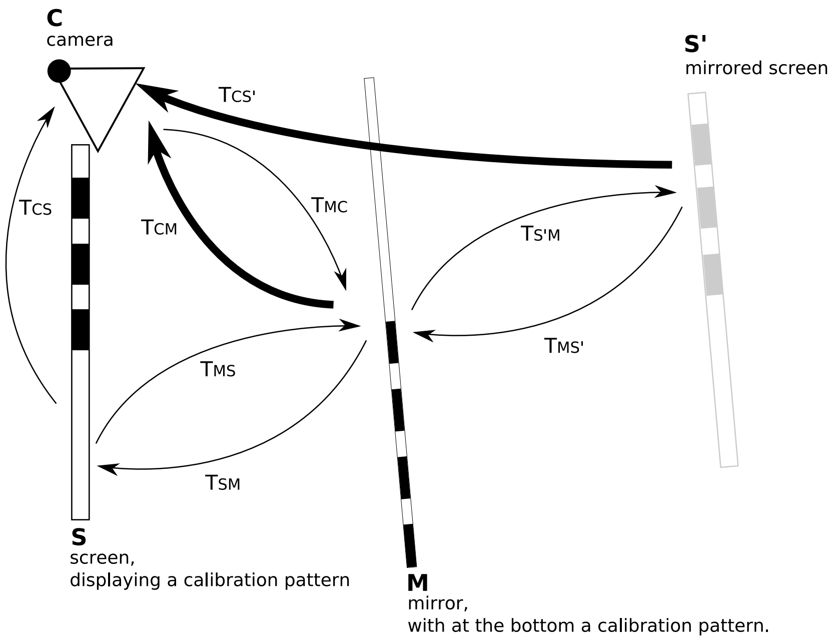

C
camera
S'
mirrored screen
TCS'
TCS
TCM
TMC
TS'M
TMS'
TMS
TSM
S
screen,
displaying a calibration pattern
M
mirror,
with at the bottom a calibration pattern.

Figure 2.4: Screen calibration using a planar mirror.

source, being the display. Two possible solutions for this problem are presented, the first one uses a planar mirror, the second one a spherical mirror.

2.2.1 Screen Calibration using a Planar Mirror

When doing a geometric calibration of a camera, the needed parameters to construct the camera matrix are obtained with the use of a calibration pattern *hartley2004mvg, tclcalib, cctm, cct*. The external parameters describe the location of the pattern with respect to the camera.

Note: *It is assumed that the internal parameters of the camera are already known.*

From this follows that placing a planar mirror in the field of view of the camera and provided a calibration pattern is attached to it, the external parameters and thus the location with respect to the capturing camera can be computed. Consequently the screen, displaying a calibration pattern, can be placed in such a way that a *mirrored* version is visible to the capturing camera. The location of the mirror and a mirrored version of the screen with respect to the capturing camera can be computed in this manner. How to compute the location of the *real* screen with this information will now be explained.

Note: *A 4×4 transformation matrix from reference frame A to reference frame B will be noted as T_{BA}*

An overview of screen calibration with the use of a planar mirror is shown in figure 2.4. The known transformations are shown in bold: T_{CM}, the tranformation from the mirror (M) to the camera (C), and $T_{CS'}$, the transformation from the mirrored screen (S') to the camera. T_{CS} is the required transformation from screen to camera and can be written as

$$T_{CS} = T_{CM}T_{MS} \tag{2.12}$$

where T_{CM} is known and T_{MS} is the transformation from screen to mirror. It is possible to recover T_{MS} from $T_{MS'}$, the transformation from mirrored screen to mirror. The mirror surface is the x-y plane in the local coordinate frame of the mirror. Consequently the mirroring is accomplished by inverting the sign of the z component. Let

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

such that

$$T_{MS} = MT_{MS'} = MT_{MC}T_{CS'} = MT_{CM}^{-1}T_{CS'}. \tag{2.14}$$

Using equation (2.14), equation (2.12) can be rewritten as

$$T_{CS} = T_{CM}MT_{CM}^{-1}T_{CS'}. \tag{2.15}$$

The tranformation matrix from screen to camera is now expressed in the two known tranformations, yielding a solution [Fun05, FY07].

2.2.2 Screen Calibration using a Spherical Mirror

Another way to accomplish screen calibration is using a spherical mirror [FHB07, TLGS05]. This technique aims to present a user-friendly way to obtain the location of the screen with respect to the camera. This is accomplished by a fully automated technique where the only needed user intervention is that of moving a spherical mirror around. An overview of this screen calibration technique is shown in figure 2.5. The process consists of two stages:

1. Detecting the position of the spherical mirror, using only the camera image of the sphere and its radius, illustrated by the first two upper-left images of figure 2.5.
2. Computing the reflection vectors associated with each individual corner for each input image. By combining sets of reflection vectors the 3D-location of the screen surface is recovered. This is illustrated by the two following upper-right images and the four lower images of figure 2.5.

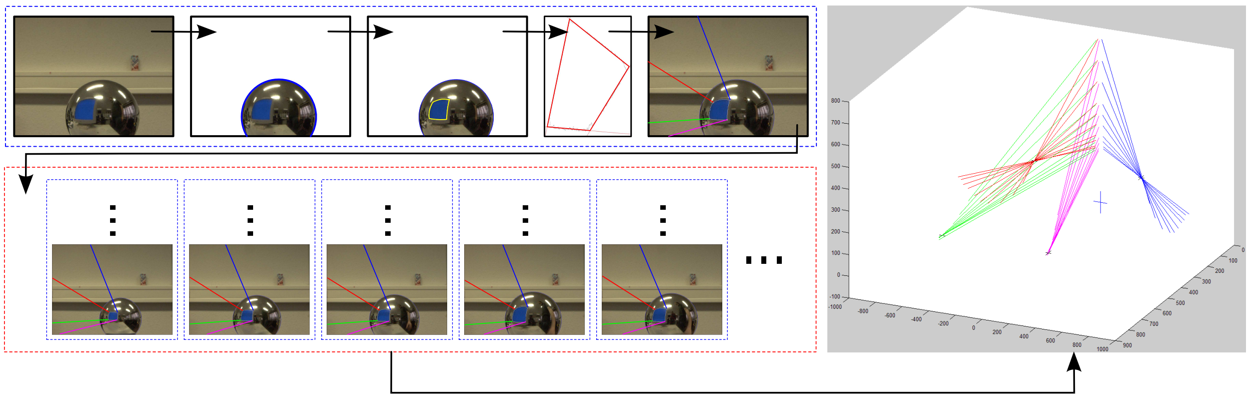


Figure 2.5: Screen calibration using a spherical mirror. Image courtesy of Francken et al [FHB07].

Locating the Spherical Mirror

Locating the spherical mirror means getting the 3D-coordinates of the sperical mirror-image contour in the camera reference frame. This is accomplished by a background subtraction step, resulting in the pixels associated with the spherical mirror to be identified. A set of morphological operations are utilized to eliminate unwanted pixel noise and bridging minor lapses yielding the actual contour. For actual ellipse detection a RANSAC-based [FB81] approach is used. Given the contour of the sphere, the next step is the location of the sphere in the camera reference frame. The camera is placed at the origin. Next a cone is defined, the top is located at the camera and the base is located at the recovered contour of the sphere. To simplify calculations this cone is aligned with the Z-axis of the camera reference frame by a simple rotation. Assuming the radius of the sphere is known, the geometrical properties of the cone are used to compute the camera-sphere (top-base) distance and the center of the sphere (base).

Locating the Screen

By letting the screen emit a constant luminance value in a single color channel the screen-pixels in the sphere contour can be detected. A set of morphological operations is used to extract the edge of the screen, similar to the approach that was used to locate the sphere contour. Due to the non-linearity of the four screen edges and possible distortions, commonly used corner detectors are unable to properly locate the screen corners. This problem can be solved by transforming the contour pixels to a better suited coordinate space as proposed by Francken et al. [FHB07]. This is illustrated by figure 2.6. This transformation reduces the problem of locating the screen corners to line detection and the choice of appropriate intersection points. A RANSAC-based approach is used to detect the four edges and appropriate intersection points are chosen. This results in good estimates of the screen corners. Using the reflection equation, the camera position, the distance sphere-

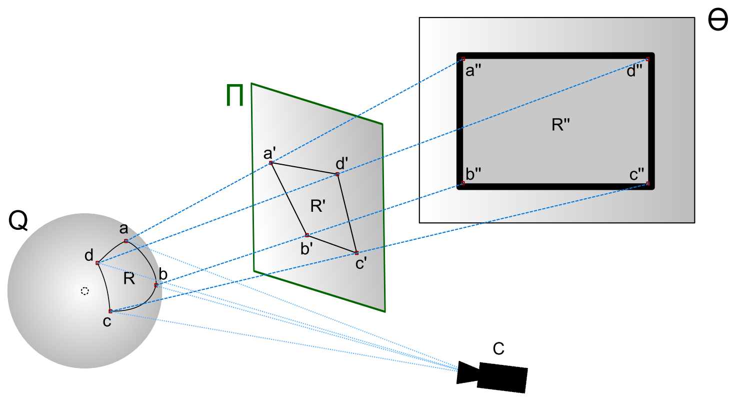


Figure 2.6: By projecting the reflections of the screen contour on an appropriately parametrized plane, the problem of corner detection is reduced. Image courtesy of Francken et al. [FHB07].

camera and the sphere radius, it is possible to compute a reflection vector or ray for each corner. See figure 2.5. When the screen corners are located for a number of input images, all sets of reflection rays associated with each individual corner can be combined into a least squares problem. Once the 3D-position of the individual screen corners is computed, all pixels located on the screen surface can be mapped onto their 3D-coordinates in the camera reference frame by bilinear interpolation.

3
Shape Recovery

Shape recovery is a classic problem in computer vision. The techniques that try to recover shape are often called *Shape-from-X* techniques, where *X* can be shading, stereo, motion, texture, specularity, etc. The goal is to reconstruct a three dimensional scene description given one or more two dimensional images.

A concise overview of popular shape recovery techniques will be presented, including shape from shading, photometric stereo and structured light. Because these techniques are usually inadequate to reconstruct highly specular fine-scale details we will examine shape reconstruction using specularities. Starting from the general shape reconstruction from specularities the more specialized case of mesostructure from specularity will be discussed.

3.1 Shape from Shading

Shading plays an important role in human perception of surface shape [ZTCS99]. Based on this idea shape from shading tries to recover the shape from a gradual variation of shading in a single image. These techniques assume that the light source direction is known. If the direction is not known, algorithms which estimates the light source direction are used [ZC91, LR85, PEN82]. Another assumption is that of Lambertian reflectance, where the observed brightness is the same from all angles. Shape from shading can be subdivided into four groups: minimization approaches, propagation approaches, local approaches and linear approaches. An overview of these approaches is given:

Minimization approaches obtain a solution by minimizing an energy function [FC88, Hor90]. This function can involve:

- The brightness constraint which requires that the brightness of the reconstructed shape is the same as the input image at each surface point.

- The smoothness constraint which ensures a smooth surface reconstruction.
- The integrability constraint which ensures valid surfaces.
- The gradient constraint which requires that the intensity gradient of the reconstructed image comes close to the intensity gradient of the input image.
- The unit normal constraint which forces the recovered surface normals to be unit vectors.

Propagation approaches start from a single reference surface point, or a set of surface points where the shape either is known or can be uniquely determined, and propagate the shape information across the whole image. This approach was used by Horn when he introduced shape from shading for the first time [Hor70].

Local approaches derive shape based on the assumption of surface type. They use the intensity derivative information and assume spherical surface.

Linear approaches compute the solution based on the linearization of the reflectance map. The idea is based on the assumption that the lower order components in the reflectance map dominate.

For the interested reader we refer to the work of Zhang et al. [ZTCS99].

3.2 Photometric Stereo

Photometric stereo was introduced by Woodham [Woo78, Woo89]. Similar to shape from shading the assumption is made that light source directions are known and visible surfaces are Lambertian. The algorithm takes as input a set of images, each taken from the same viewpoint but with the scene lit under different lighting conditions, as shown in figure 3.1. Photometric stereo is a local method, the calculations are performed independently at every pixel. Woodham based the original formulation of photometric stereo on the idea of reflectance maps. A reflectance map $R(p, q)$ maps a specific surface orientation (p, q) to a surface radiance value. The reflectance map combines the BRDF (bidirectional reflectance distribution function) and light source vector into one function. Using multiple input images, multiple reflectance maps, surface normals are computed.

The basic photometric stereo algorithm has been extended since it was firstly introduced by Woodham. Tagare and deFigueiredo [Td91] were one of the first to extend the basic algorithm to work with non-Labertian surfaces. Although their approach was not general enough to account for all possible surface and illumination conditions, a first step was taken towards a theory of non-Lambertian photometric stereo. They also showed that it is inappropriate to attempt recovery of non-Lambertian surface under the Lambertian assumption. More recently Hertzmann and Seitz [HS05] developed a technique for computing the geometry of objects with general reflectance properties from images. They use

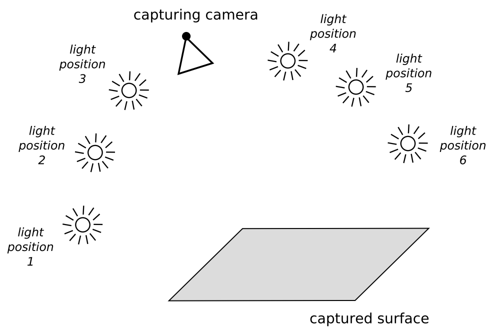


Figure 3.1: Photometric stereo, the input into the algorithm is a set of images, each taken from the same viewpoint but with the scene lit under different lighting conditions.

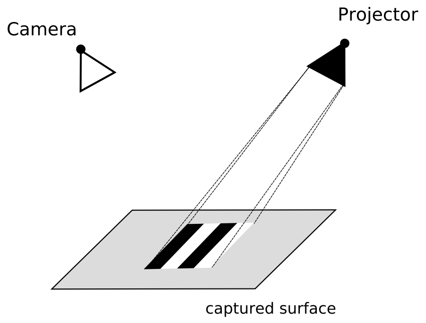


Figure 3.2: Typical structured light setup.

objects with known geometry with the same or similar BRDF as the object to be recovered, referred to as *example-based photometric stereo*.

The traditional photometric stereo algorithm assumes that all the light sources are distant point sources at known locations. Note that the last mentioned technique also operates under more general or unknown lighting conditions. There are also techniques that assume Lambertain reflectance but operate with less restricting lighting assumptions. Basri et al. [BJK07] show how to perform photometric stereo assuming that all lights in a scene are distant from the object but otherwise unconstrained.

3.3 Structured Light

Structure light operates by projecting known patterns of light on the scene to be measured. A single camera captures the lit scene. Figure 3.2 shows a typical structured light setup. The required 3D information can be obtained by analyzing the deformations of the imaged pattern with respect to the projected one. With knowledge of relevant camera and projector geometry, the depth of the measuring scene can be calculated by triangulation.

Joaquin et al. presented a classification of coding strategies for structured light sys-

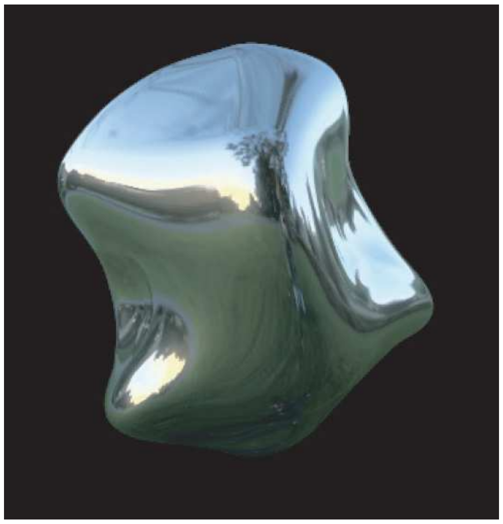


Figure 3.3: A computer-generated image of a perfectly specular surface. Image courtesy of Fleming et al [FTA04].

tems [J$^+$01]. When using pattern codification the patterns are specially designed so that codewords are assigned to a set of pixels. Every coded pixel has its own codeword, so there is a direct mapping from the codewords to the corresponding coordinates of the pixel in the pattern. For example Boyer and Kak [BK87] proposed a technique which uses a pattern formed by vertical slits coded with the three basic colors (red, green and blue) and separated by black bands. The sequence of colored slits was designed so that if the pattern is divided into subpatterns of a certain length, none is repeated. The most interesting thing about this work is the decoding stage. Boyer and Kak realized that the morphology of the measuring surface acts as a deviation applied to the projected pattern (which acts like a signal), so the received pattern can contain disorders or even deletions of the slits.

Scharstein and Szeliski [SS02] provided a taxonomy of existing stereo algorithms that allows the dissection and comparison of individual algorithm components design decisions. They also provided an online test bed[1] for the quantitative evaluation of stereo algorithms.

For the interested reader we refer to the work of Zhang et al. [ZCS03], Scharstein and Szeliski [SS03] and Waschbusch et al. [WWC$^+$05] for further reading.

3.4 Shape from Specularity

The rest of this chapter will be devoted to shape recovery with the use of specular reflections. First we will show that specular reflections are in fact a reliable cue for three-dimensional shape recognition. A computer-generated image of a perfectly specular surface is shown in figure 3.3. Most observers perceive the three-dimensional shape of the object

[1]http://www.middlebury.edu/stereo

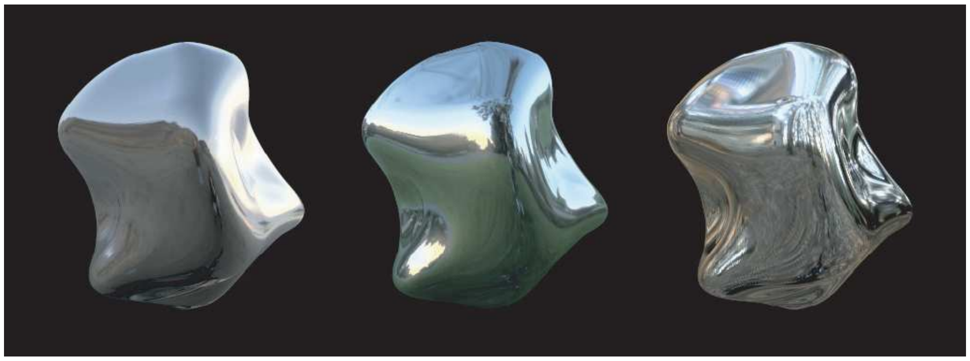


Figure 3.4: The image consists of nothing more than a distorted reflection of the world surrounding the object. Image courtesy of Fleming et al [FTA04].

[FTA04]. This is surprising given that many of the traditionally cues for shape perception are absent. Cues that the techniques mentioned before use. Figure 3.3 contains no shading in the traditional sense of the word, so no shape from shading information. It is only a single image so no shape from stereo or motion information is available. Nevertheless a three dimensional shape can be recovered.

3.4.1 Interpreting Specular Reflections

Unlike texture markings or shadows, specularities slide over the surface and change shape whenever the object, viewer, or environment moves. Specular surfaces inherit their appearance solely from their environment: every visible feature belongs to the world surrounding the object rather than to the object itself. See figure 3.4.

This means that a perfectly smooth object can appear to have dents or bumps simply by distorting the scene, and the observer would have no way of knowing that it is the environment rather than the shape that changed, because the image is indentical. Consequently, many possible combinations of shape and scene are consistent with a given image, as shown in figure 3.5.

Theory and practice show that objects with mirroring surface properties are outside the scope of almost all current standard approaches for high quality acquisition. In all active scanning methods (e.g. structured light systems) the problem is that the active signal (e.g. the projected pattern) is mainly reflected along the specular direction so that almost no reflected light reaches the camera. In addition, specular reflections of an object are effected by the surface normal and can therefore lead to the effect that the projected pattern is perceived on the wrong position. Shape from shading and photometric stereo based approaches require the surface to be Lambertian and hence do not work for glossy, especially not for mirroring surfaces. A solution to this problem is to paint to approximate Lambertian surface behaviour. But this is clearly invasive and unpractical for many applications.

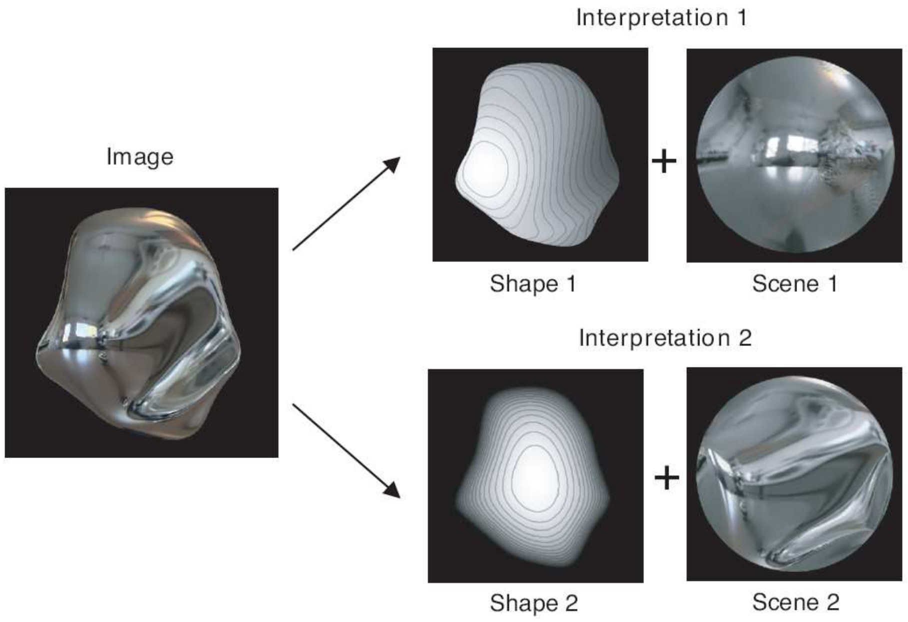


Figure 3.5: A given image of a mirrored object is consistent with many different shapes. For example, the same image could be created by placing Shape 1 in Scene 1, or by placing Shape 2 in Scene 2. Image courtesy of Fleming et al. [FTA04].

3.4.2 Recent Work on Mirroring Object Acquisition

Zheng et al. [ZFA97] rotated a specular object on a turntable under the illumination of two linear fluorescent lamps and estimated normal and geometry by the motion of the reflection during the rotation. Later Zheng and Murata [ZM00] extendeded this approach, using circular-shaped light sources, so that, if there are no occlusions due to the object shape, a highlight can be observed on all surface points regardless of their normal, thus increasing the number of possible shapes that can be recovered. While they can derive the complete geometry of an object in a single scan, the accuracy of the generated models is limited and special hardware is required.

Savarese and Perona [SP01, SP02] used a 2D pattern consisting of points identified by the intersection of three coplanar lines, and manage to recover from their reflection some attributes of the mirroring surface, i.e. relationships between depth and normal. This can be used to locally reconstruct the surface itself, if an approximation of its distance is provided, or some a priori knowledge of the shape of the mirror (a sphere or a planar mirror) is known. Still, the detection and labelling of elements of the patterns is far from automatic and indeed quite problematic for not trivially shaped mirror. More important, the measurements are very sparse. Typically their number in around a dozen, so no assumption of point-to-point continuity is possible. Therefore, the technique can be useful as auxiliary rather that substitute for typical range scanning technique.

Following a totally different approach, Bonfort and Sturm [BS03] showed that the stan-

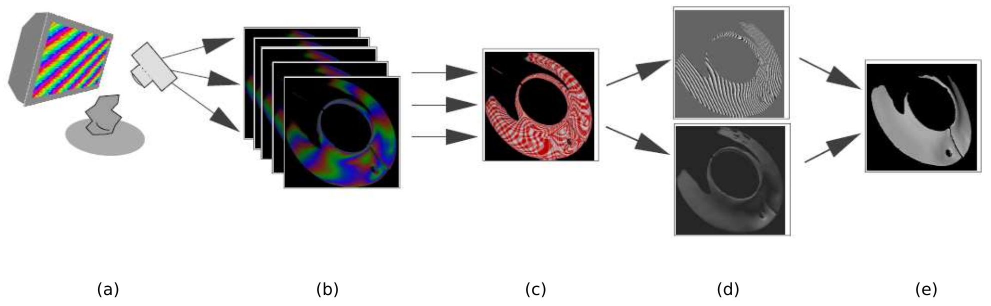


Figure 3.6: An overview of shape from distortion: (a) the object to be measured is captured while reflecting some pattern displayed by a screen, (b) the captured images are used to compute (c) an environment matte from which (d) a normal map together with a depth map is extracted which are embedded into (e) the final 3D range scan. Image courtesy of Tarini et al. [TLGS05].

dard voxel carving techniques can be adapted to the case of mirroring objects. A mirroring object is photographed in front of a known surrounding environment from different point of views. For every viewpoint, surface normals are associated to the voxels traversed by each projection ray formed by the reflection of a scene point. A decision process then discards voxels whose associated surface normals are not consistent with one another. The output of the algorithm is a collection of voxels and surface normals in 3D space, whose quality and size depend on user-set thresholds.

Most recently Tarini et al. [TLGS05] presented a new approach, *shape from distortion*, to reconstruct the surface of a mirroring object with high precision. This approach consists of two steps:

1. an improved environment matte extraction that uses the interference of patterns of different frequencies in order to obtain sub-pixel accuracy relative to the projected pattern,

2. followed by a step that converts the matte into a normal and a depth map by exploiting the self coherence of a surface when integrating the normal map along different paths.

See figure 3.6. In fact this is an extension of earlier work done by Zongker et al. [ZWCS99]. They introduced environment mattes which capture how light impinging from the environment is reflected by the object.

For the rest of this chapter we will be focusing on the recovery of specular mesostructures using specular highlights, based on the work of Chen et al. [CGS06].

3.4.3 Specular Highlights

Specular highlights can be used to recover surface normals. The law of reflection states that every specular highlight on a surface is precisely defined by the incident light direction

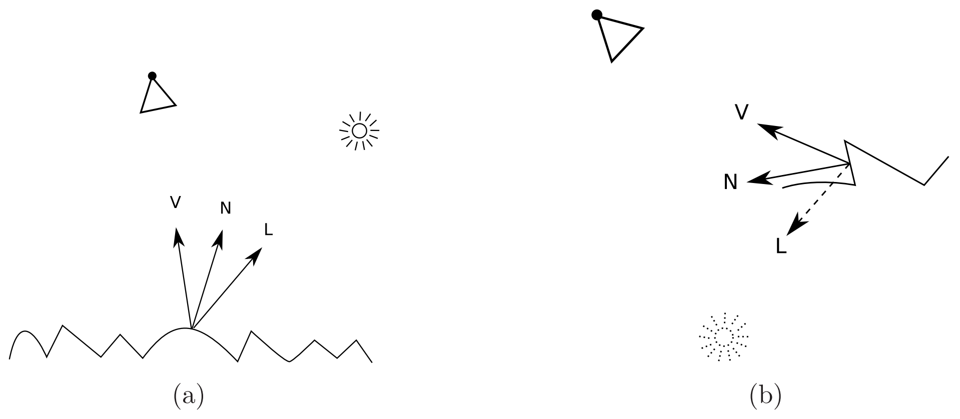


Figure 3.7: Specular reflection.

L, the surface orientation N and the viewing direction V. The angle of incidence equals the angle of specular reflection. See figure 3.7. The exact relation between these parameters is given by the following equation:

$$V = 2\left(N^T L\right) N - L. \tag{3.1}$$

So, given a specular highlight we can easily reconstruct the corresponding normal as

$$N = \frac{V + L}{\|V + L\|}. \tag{3.2}$$

Using a calibration step to locate the light source, the capturing camera and the surface to be measured, the corresponding normal can be easily reconstructed.

Note that there is a restriction on the recovery of surface normals. Not all normals can be recovered using a single capturing camera and a single light source. Figure 3.7(b) illustrates this.

3.4.4 Mesostructure from Specularity

A mesostructure surface has geometric details that are relatively small but still individually visible such as bumps or dents. It is one of the key components of three dimensional texture and contributes strongly to the complex surface appearance of real-world objects. But even for the most advanced and expensive laser scanning systems, mesostructure reconstruction of highly specular or translucent objects is still a difficult problem. The preceding discussions showed that specular reflection is a reliable visual cue for shape perception and unlike traditional methods based on shape from shading or photometric stereo, specularities are not liable to subsurface scattering. This makes the use of specularities for shape recovery interesting to use with translucent materials such as human skin and fruit.

Specular highlights will be used to recover mesostructure shape based on the work of Chen et al. [CGS06]. For a detailed reconstruction we want as much specular highlights found as possible. To accomplish this a high directional light source resolution is needed. This means capturing the scene lit from many different directions, so that a large number of specular highlights can be captured.

The next step will be to recover the captured specular highlights. Efficient and robust separation of diffuse specular components or surface reflection for arbitrary materials, especially for translucent or refractive materials, is still an open problem. Chen et al. [CGS06] used a simple histogram thresholding method to extract specular reflection component. The thresholding is based on the intensity (I) of each pixel [GW02]:

$$I = \frac{R + G + B}{3} \tag{3.3}$$

where the color of the pixel is expressed in the RGB color model. A pixel contains a specular highlight when the intensity computed for that pixel is greater than or equal to a given threshold t. Given an image a specularity field F_s can be recovered:

$$F_s = \begin{cases} 1, & \text{specular highlight was observed at } (x, y) \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

When processing multiple input images at once, the given threshold value t might be less appropriate for some of the images. To solve this problem we introduce an extra parameter r, which we refer to as the *range* parameter. We now say that a pixel contains a specular highlight when the intensity computed for that pixel is greater than or equal to the given threshold t *and* is within $[h - r, h]$, where h is the highest intensity observered for the current image.

We assume that the positions and orientations of the individual light sources with respect to the capturing camera are known by the use of a calibration step. This means that once the specular highlights are found, the surface normal can be recoverd using equation (3.2). When more than one specular highlight is found at a particular pixel throughout the different input images, the light source associated with the highest intensity pixel is used.

4 Controlled Illumination

When using specular highlights to recover shape, a high directional light source resolution is an important requirement. This means the scene to be measured needs to be lit from many different directions consequently the acquisition time will be long. We propose a multiplexed illumination technique that will strongly reduce the acquisition time while keeping a high directional resolution. A raster display will be used as the illumination source, where each pixel or a set of pixels will function as an individual light source.

4.1 Raster Display as Illumination Source

The most common raster display devices are CRT monitors and LCD screens. A CRT monitor contains a cathode ray tube in which three electron beams are used to light up phosphors on the display surface. The light from LCD displays originates from a backlight which is covered with a layer of liquid crystals between two polarization filters. The liquid crystals are controlled to rotate the polarized light and thereby adjust the amount of light passing through that layer. The screen is divided up into pixels which are again divided up into red, green and blue cells (or subpixels). These are controlled individually to adjust the perceived color.

It would be wrong to assume that the light radiance from the pixels of any raster display is constant in all directions. In contrast with CRT monitors, most LCD screens appear brighter when viewed from the centre than when viewed at an angle. This radiance-direction dependency is an effect of the physical construction of a LCD screen. Light sources that do not emit light equally in all directions are referred to as *directional*. Figure 4.1 illustrates the directional property of LCD screens in contrast with CRT monitors. Another difference between LCD screens and CRT monitors is image flickering, CRT monitors are

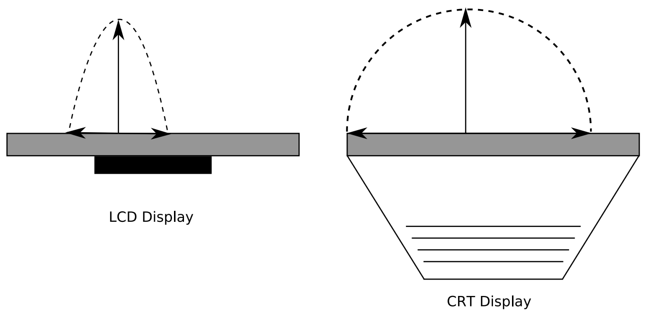


Figure 4.1: LCD versus CRT displays, the viewing angle for CRT displays is far greater than the viewing angle for LCD displays.

prone to flicker in contrast to LCD screens who are unaffected by this effect. Both displays will also always emit a certain amount of light, even when all pixels are set to *black*. If one wants to examine the lighting from only a few pixels on a LCD screen or CRT monitor, it is necessary to compensate for the additional light from all other pixels which are set to black. This can be achieved by taking one image of the scene lit with the entire screen set to black, and a second image with the desired pixels on. Then, by subtracting the first image from the second, the contributions of the desired pixels on the scene can be determined. This also eliminates potential contributions from ambient lighting of the scene. It does however require that the ambient lighting conditions do not change from the first image to the second [Fun05].

4.2 Multiplexed Illumination

In contrast to structured light, which projects patterns onto the object and looks at the deformation of the pattern on the object, we will multiplex different *directions* from which the object is illuminated [SNB03]. See figure 4.2.

Lighting objects from a single direction will usually result in underexposed images. Longer exposure times would solve this but this would lengthen the acquisition time, and a very stable setup would be essential. The multiplexing principle offers a better solution. The object is illuminated by multiple directions while all features obtained with a single direction source like specularities, shading, etc. are preserverd.

4.2.1 Formal Approach

We will start with a formal approach of multiplexed illumnation based on the work of Schechner et al. [SNB03]. Let the vector $\vec{\theta}$ parameterize the direction from which the light source illuminates an object point. The vector $\vec{\theta}$ is measured in the global coordinate system of the illumination system. Let $i_{\vec{\theta}}(x, y)$ denote the value of a specific image pixel (x, y) under a single, narrow light source at $\vec{\theta}$. We require this value for a range of lighting

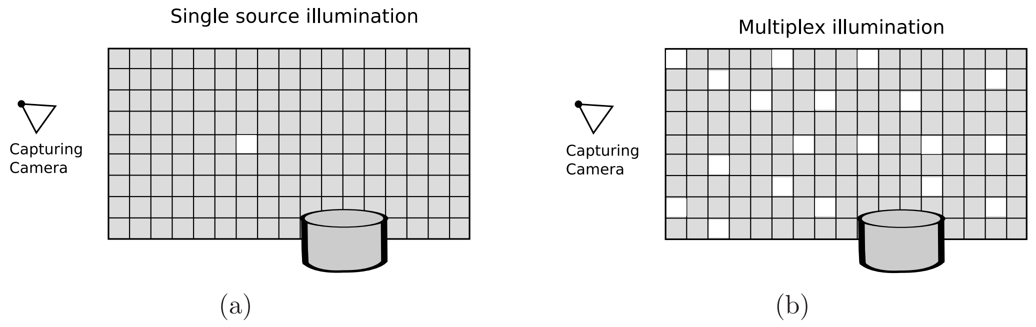


Figure 4.2: (a) Single-source illumination versus (b) multiplexed illumination.

directions, thus $\mathbf{i}\left(x,y\right)$ denotes the vector of values of the image irradiance at that single pixel, with varying $\vec{\theta}$. The length of $\mathbf{i}\left(x,y\right)$ is n, corresponding to the number of individual light source directions. We denote $\mathbf{a}\left(x,y\right)$ the vector of measurements acquired under different lighting settings (typically with multiple simultaneous illumination sources). The number of acquired images (the length of $\mathbf{a}\left(x,y\right)$) equals n. We note that, while we deal with image intensity measurements, they are equivalent to the object radiance.

The images acquired with different lighting conditions represent light energy distributions. Therefore $\mathbf{i}\left(x,y\right)$ and $\mathbf{a}\left(x,y\right)$ are additive quantities, and are related to each other by a lineair superposition:

$$\mathbf{a}\left(x,y\right) = W\mathbf{i}\left(x,y\right) \tag{4.1}$$

where W is a weighting matrix or the *multiplex matrix*. Each row m in the matrix W denotes which of the light sources are "on" and which are "off" when the image is captured. Typical W will consist of ones and zeros, where a one will be associated with an "on" light source and a zero with an "off" light source. Each column s in this matrix corresponds to a specific illumination source, equivalent to a specific $\vec{\theta}$. We estimate $\mathbf{i}\left(x,y\right)$ by

$$\widehat{\mathbf{i}}\left(x,y\right) = W^{-1}\mathbf{a}\left(x,y\right) \tag{4.2}$$

where W^{-1} is the *demultiplex matrix*. When only a single light source is "on" at any time, $\widehat{\mathbf{i}}\left(x,y\right)$ is equal to a raw measured value. Thus $W = I$, where I is the identity matrix. However, the intensities per illumination direction can be multiplexed in the acquired measurements, i.e., W can be general. Then, each measurement $\mathbf{a}\left(x,y\right)$ simultaneously acquires enegery corresponding to lighting from multiple directions. Thus, the energy in $\mathbf{a}\left(x,y\right)$ can be made larger than in single-source lighting, potentially increasing the quality of the estimated $\widehat{\mathbf{i}}\left(x,y\right)$.

4.2.2 Specular Highlights

We will now see what the effect of multiplexing is on specular highlights. Each measurement is described by a row m in equation (4.1). The acquired value is

$$a_m(x,y) = \sum_{s=1}^{n} w_{m,s} i_s(x,y)\,. \tag{4.3}$$

We represent the intensity $i_s(x,y)$ as a sum of a diffuse component and a specular component:

$$a_m(x,y) = \sum_{s=1}^{n} w_{m,s}\left[i_s^{diffuse}(x,y) + i_s^{specular}(x,y)\right]. \tag{4.4}$$

The captured image is composed of such components too:

$$a_m(x,y) = a_s^{diffuse}(x,y) + a_s^{specular}(x,y) \tag{4.5}$$

where

$$a_m^{diffuse}(x,y) = \sum_{s=1}^{n} w_{m,s} i_s^{diffuse}(x,y)\,. \tag{4.6}$$

A highlight due to specular reflection at a pixel (x,y) does not occur for most light source directions $\vec{\theta}$. Rather, it occurs if the illumination comes from a very narrow solid angle around a single direction. For a highly specular surface we thus say that, only one source $\tilde{s}(x,y)$ produces a specular highlight at (x,y) which can be seen from the position of the camera. Therefore,

$$i_s^{specular}(x,y) = i_s^{specular}(x,y)\,\delta\left[s, \tilde{s}(x,y)\right]. \tag{4.7}$$

It follows that

$$i_m^{specular}(x,y) = w_{m,\tilde{s}_{(x,y)}} i_{\tilde{s}_{(x,y)}}^{specular}(x,y)\,. \tag{4.8}$$

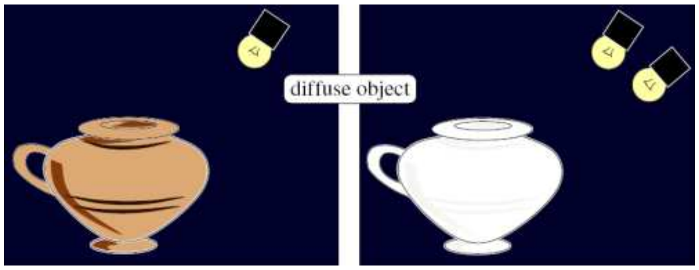
Suppose that in a single-source image, the light source is "on" in a direction corresponding to the highlight, i.e., $w_{m,s} = \delta(s, [\tilde{s}(x,y)])$. The captured image is then

$$a^{single}(x,y) = i_{\tilde{s}}^{diffuse}(x,y) + i_{\tilde{s}}^{specular}(x,y) \gg i_{\tilde{s}}^{diffuse}(x,y)\,. \tag{4.9}$$

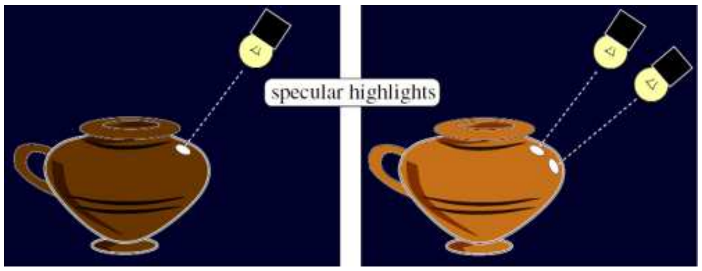
In such cases, we get the familiar situation in which the specular highlight at (x,y) is much brighter than most of the image pixels, which measure only the diffuse component. This creates a problem of dynamic range. In contrast, using multiplexed illumination, when the light source corresponding to the highlight is "on", an amount of other light sources, which do not create a highlight in (x,y) are "on" as well. Then,

$$a^{multiplexed}(x,y) \quad [r]\, i_{\tilde{s}}^{diffuse}(x,y) + i_{\tilde{s}}^{specular}(x,y) \tag{4.10}$$

with r the amount of light sources that are "on", being far greater than one in a multiplexed scheme. The diffuse component in the captured image $a^{multiplexed}$ is significantly brighter than in a a^{single}, while the specular component is (almost) not amplified. This greatly reduces the dynamic range problem. See figure 4.3.



(a)



(b)

Figure 4.3: Multiplexed illumination and specular highlights: (a) Multiplexing does not apply to *bright* diffuse objects, which may saturate. (b) A highlight forces low exposure settings to avoid its saturation. Multiplexing brightens most of the image, but does not saturate the highlights, thanks to the fact that a highlight is uniquely defined by a single light source. Images courtesy of Schechner et al. [SNB03]

4.3 Our Approach

Mesostructure from specularity requires a high directional resolution. This means that the scene has to be lit with as many individual directional light sources as possible. This can be a tedious and time-consuming job, therefore a method that allows simulation of n individual directional light sources with only $\log_2 n + 1$ patterns is proposed. Let us recall equation (4.1)

$$\mathbf{a}(x, y) = W\mathbf{i}(x, y)$$

where $W \in \mathbb{R}^{n \times n}$. In our approach, W will not be a square matrix since n individual directional light sources are simulated with $\log_2 n + 1$ patterns. These patterns are in fact the rows of W. Let the number of patterns (the rows of W) be noted with m (equal to $\log_2 n + 1$) and the new weighting matrix or multiplex matrix with S. Let us now rewrite equation (4.1)

$$\mathbf{a}(x, y) = S\mathbf{i}(x, y) \tag{4.11}$$

where $S \in \mathbb{R}^{m \times n}$. This poses a new problem when calculating an estimate of $\mathbf{i}$ $(\widehat{\mathbf{i}})$. Before $\mathbf{i}$ was estimated with equation (4.2)

$$\widehat{\mathbf{i}}(x, y) = W^{-1}\mathbf{a}(x, y).$$

But since S is not a square matrix this becomes more of a challenge. This is solved by introducing an alternative to the identity matrix, but first a convenient property of specular highlights will be shown. Only one source $\tilde{s}(x, y)$ produces a specular highlight at (x, y) which can be seen from the position of the camera. This means that specular highlights contained in a single source image are unaffected when another (different) single source image is subtracted. Recall equation (4.9)

$$a^{single}(x, y) = i_{\tilde{s}}^{diffuse}(x, y) + i_{\tilde{s}}^{specular}(x, y) \gg i_{\tilde{s}}^{diffuse}(x, y).$$

This means that the specular component is far greater than the diffuse component. Consequently the intensity at (x, y) will be nearly unaffected when the value of (x, y) from another image is subtracted because the latter can not contain a specular highlight at (x, y).

Note: *It should be noted that we make this assumption since we are only interested in specular highlights.*

Using this property, an alternative to the identity matrix can be defined:

$$\widehat{I} = TS = \begin{bmatrix} 1 & \leq 0 & \dots & \leq 0 \\ \leq 0 & 1 & \dots & \leq 0 \\ \vdots & \vdots & \ddots & \vdots \\ \leq 0 & \leq 0 & \dots & 1 \end{bmatrix}. \tag{4.12}$$

with $T \in \mathbb{R}^{n \times m}$. The determination of T will be done by dividing the problem into a number of linear least-squares problems. A linear least-squares problem is given by:

$$\underset{x}{min} \ \frac{1}{2} \|Cx - d\|^2 \tag{4.13}$$

such that

- $Ax \leq e$
- $Bx = f$

where A, B and C are matrices and b, f and d are vectors, and are defined for each row i of T as:

- $C = S^T \in \mathbb{R}^{n \times m}$ (S being the multiplex matrix $\in \mathbb{R}^{m \times n}$ with m the number of patterns / rows and n the number of light sources)
- $d^T \in \mathbb{R}^n$ containing only zeros except for element i which is equal to 1
- $A \in \mathbb{R}^{(n-1) \times m}$ is equal to C (S^T) minus row i
- $e^T \in \mathbb{R}^{n-1}$ containing only zeros
- $B \in \mathbb{R}^m$ equal to row i of C (S^T)
- $f \in \mathbb{R}$ equal to 1.

This will be illustrated by the special case of 4 light sources. The corresponding multiplex matrix $S \in \mathbb{R}^{3 \times 4}$ will be:

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \tag{4.14}$$

consequently $C \in \mathbb{R}^{4 \times 3}$ becomes:

$$C = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}. \tag{4.15}$$

The computation of the first row ($i = 1$) of $T \in \mathbb{R}^{4 \times 3}$ will now be elaborated. $d^T \in \mathbb{R}^4$ becomes

$$d^T = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \tag{4.16}$$

and the other variables become

$$A = \begin{bmatrix} \cancel{1} & \cancel{1} & \cancel{1} \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \tag{4.17}$$

$$e^T = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}, \tag{4.18}$$

$$B = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{4.19}$$

and $f = 1$, where $A \in \mathbb{R}^{3\times6}$, $e^T \in \mathbb{R}^3$ and $B \in \mathbb{R}^3$. Let us note T as

$$T = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \\ x_{4,1} & x_{4,2} & x_{4,3} \end{bmatrix}. \tag{4.20}$$

Next we will look at the conditions of equation (4.13):

- $Ax \leq e$, A contains all rows of C except for the row that is being computed (i). Hence the results of this multiplication correspond to the '≤ 0'-elements in equation (4.12). Applied to our example this becomes (for the first row):

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{1,1} \\ x_{1,2} \\ x_{1,3} \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{4.21}$$

- $Bx = f$, B contains the row number equal to the row of T that is being computed. Hence the results of this multiplication correspond to the '1'-elements in equation (4.12).

$$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_{1,1} \\ x_{1,2} \\ x_{1,3} \end{bmatrix} = [1]. \tag{4.22}$$

The notation in previous examples might be confusing, this will be clarified by rewriting equation (4.12)

$$\widehat{I} = S^T T^T = \begin{bmatrix} 1 & \leq 0 & \dots & \leq 0 \\ \leq 0 & 1 & \dots & \leq 0 \\ \vdots & \vdots & \ddots & \vdots \\ \leq 0 & \leq 0 & \dots & 1 \end{bmatrix}. \tag{4.23}$$

By using equation (4.20) and equation (4.23) this results in

$$S^T T^T = \begin{bmatrix} \mathit{1} & \mathit{1} & \mathit{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} x_{1,1} & x_{2,1} & x_{3,1} & x_{4,1} \\ x_{1,2} & x_{2,2} & x_{3,2} & x_{4,2} \\ x_{1,3} & x_{2,3} & x_{3,3} & x_{4,3} \end{bmatrix} = \begin{bmatrix} \mathit{1} & \leq 0 & \leq 0 & \leq 0 \\ \leq \mathbf{0} & 1 & \leq 0 & \leq 0 \\ \leq \mathbf{0} & \leq 0 & 1 & \leq 0 \\ \leq \mathbf{0} & \leq 0 & \leq 0 & 1 \end{bmatrix} \tag{4.24}$$

with equation (4.21) shown bold and equation (4.22) shown slanted. This shows how to construct T by representing each row i of T as a linear least-squares problem. By empirical

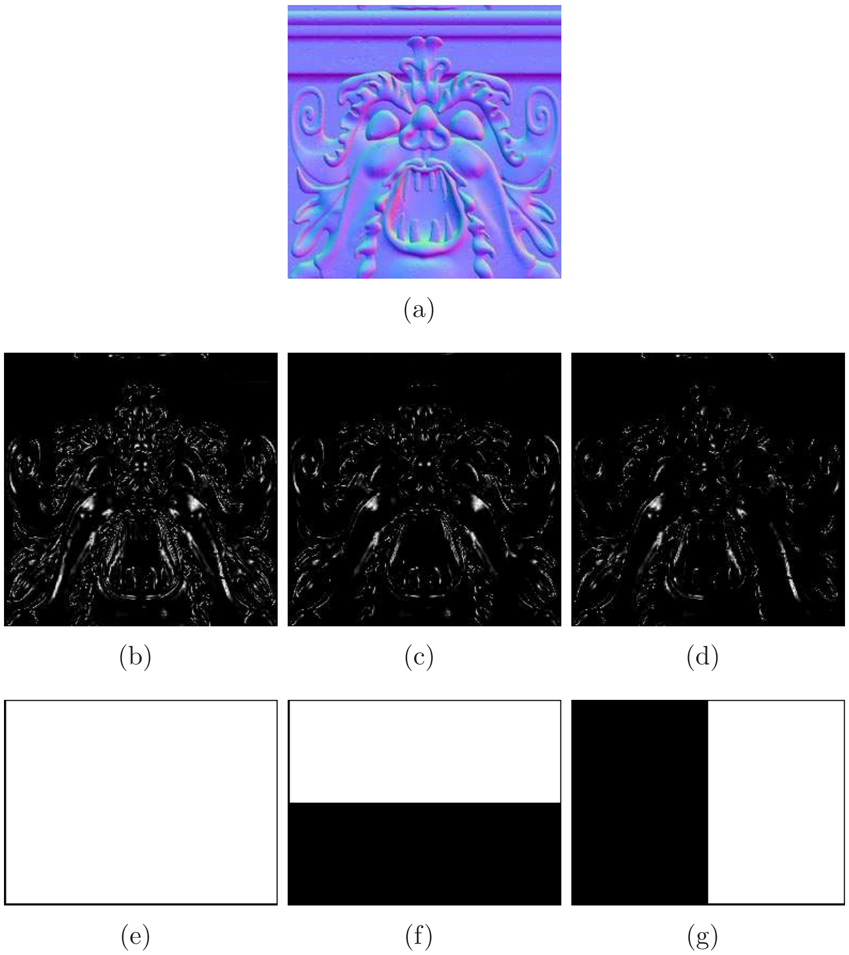


Figure 4.4: Images captured using patterns: (a) The original normal map, used to generate the images, (b-d) images captured while lit by (e-g) the gray code patterns. The patterns correspond to the images in the same column.

evaluation, it was found out that it is possible to rewrite T as:

$$T = 2\mathbf{S}^T - 1_{n\times m} + \mathbf{S}^T \begin{bmatrix} -1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & \dots & 0 \end{bmatrix}^{\in\mathbb{R}^{m\times m}} \begin{bmatrix} 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \dots & 0 \end{bmatrix}^{\in\mathbb{R}^{m\times n}} \tag{4.25}$$

consequently an estimate of $\mathbf{i}$ is given by

$$\widehat{\mathbf{i}}\,(x,y) = T\mathbf{a}\,(x,y)\,. \tag{4.26}$$

The formal approach that was given earlier has now been extended to our approach.

Each row of S represents a *light pattern*. To be able to emulate n light sources with $\log_2 n+1$ (m) patterns, *gray code* patterns are used. This is a technique used in structured light systems [J+01]. This will be illustrated with the special case of 4 light sources. The number of patterns needed is equal to 3 ($\log_2 4 + 1$). For this special case equation (4.11) becomes

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix}. \tag{4.27}$$

This is illustrated by figure 4.4 and shows why $\log_2 n + 1$ instead of $\log_2 n$ patterns are needed. Although $\log_2 n$ bits are enough to encode n binary numbers, S in equation (4.27) would become:

$$S = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \tag{4.28}$$

resulting in one column with only zeros. In this case no measurement of light source 4 will be captured. To avoid this, an extra bit is added. This extra bit is always “on” (set to 1) and becomes the MSB (most significant bit).

Equation (4.25) and (4.26) are used to compute an estimate for the 4 *individual* light sources:

$$\begin{bmatrix} \widehat{i_1} \\ \widehat{i_2} \\ \widehat{i_3} \\ \widehat{i_4} \end{bmatrix} = \begin{bmatrix} -1 & 1 & 1 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}. \tag{4.29}$$

This is illustrated by figure 4.5. Equation (4.29) shows how to recover the single light source images from the multiplexed light source images.

4.3.1 Pixel Codification

This section will discuss the use of pixel codification to allow for a fast recovery of the single source images. The use of gray coded patterns was presented earlier. These patterns encoded every single light source with an unique binary code. Let us recall example 4.27, the binary code for each of the 4 single light sources is given by:

$$\begin{aligned} i_1 &= 111 \\ i_2 &= 110 \\ i_3 &= 101 \\ i_4 &= 100. \end{aligned}$$

Now for each pattern each pixel in the measuring scene is assigned a one or a zero. A one is assigned when a specular highlight is detected at that pixel, a zero otherwise. By doing this each pixel is assigned a binary codeword, the first pattern will determine the MSB, the last pattern the LSB (least significant bit). This is illustrated in figure 4.6. After this process for each detected specular highlight the corresponding light source can be computed. The following equation relates the binary code with the light source number:

$$\text{light source number} = 2^{(\text{number of patterns})} - \text{decimal notation of the binary code.} \tag{4.30}$$

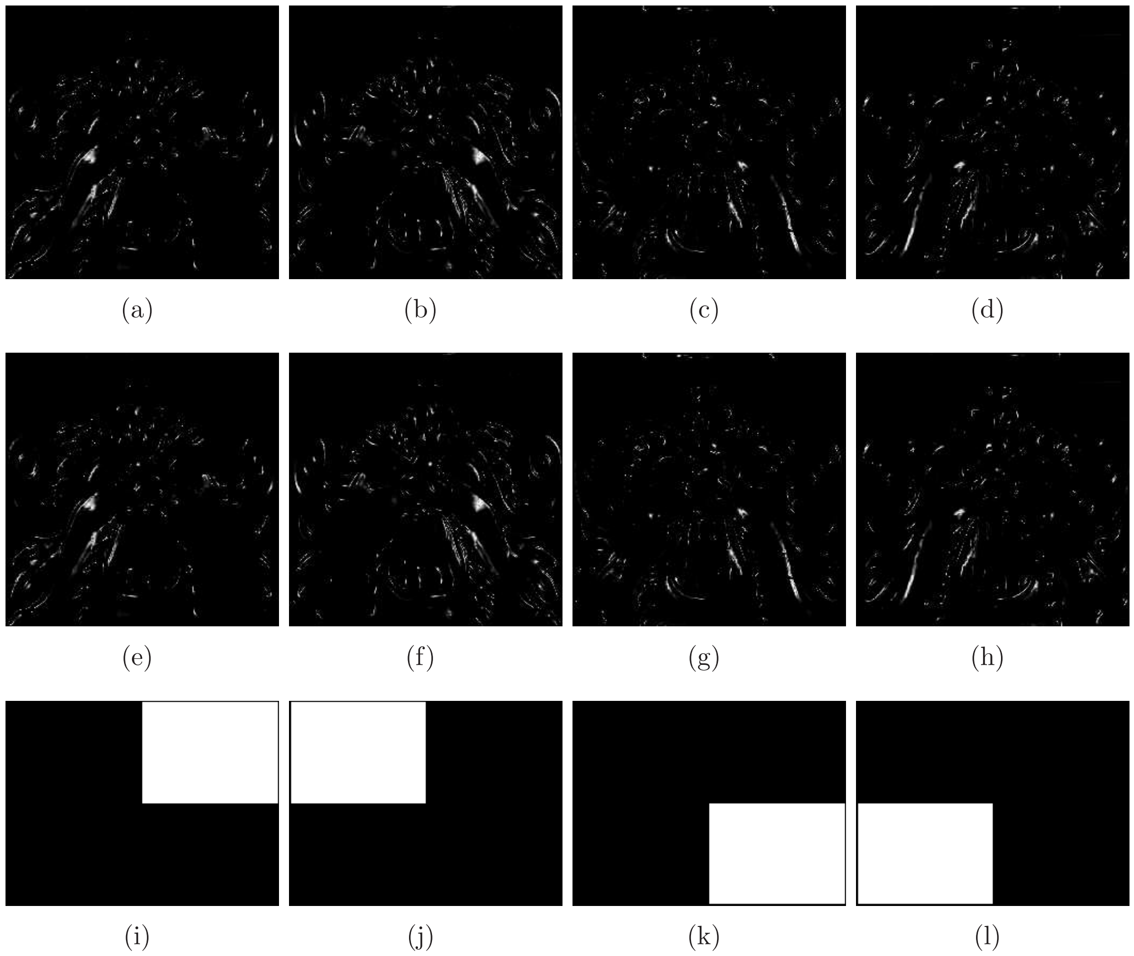


Figure 4.5: Decoding: (a-d) the decoded single source images, (e-h) the original single source images, (i-l) the single light sources. The patterns correspond to the images in the same column.

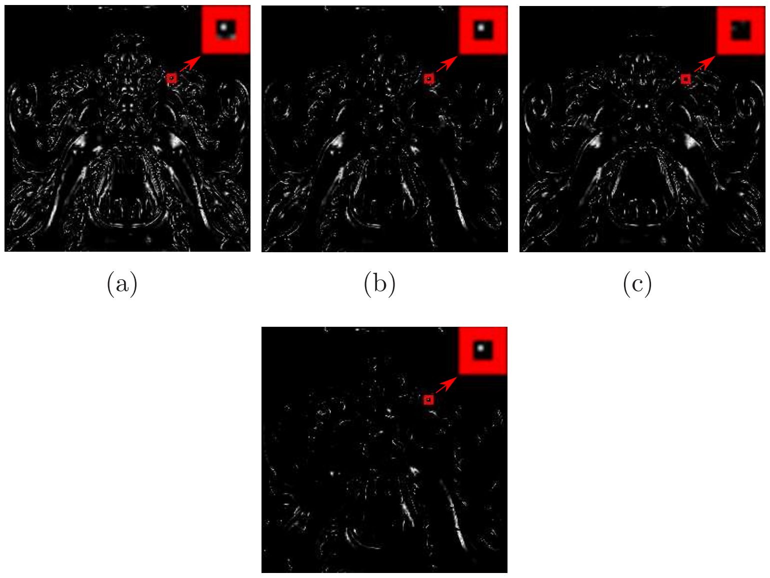


Figure 4.6: Pixel codification: the first two images (a) and (b) show a specular highlight, yielding codeword 110, which is light source number 2 and indeed the specular highlight is found on the second single source image (c).

When applied to our example:

$$\begin{array}{ccccccccccc} i_1 & = & 2^3 & - & 2^2 & + & 2^1 & + & 2^0 & = & 1 \\ i_2 & = & 2^3 & - & 2^2 & + & 2^1 & & & = & 2 \\ i_3 & = & 2^3 & - & 2^2 & & & + & 2^0 & = & 3 \\ i_4 & = & 2^3 & - & 2^2 & & & & & = & 4. \end{array}$$

The use of pixel codification quickly relates detected specular highlights to their corresponding light sources.

5 Implementation

In this chapter our implementation is discussed, an application which is able to calibrate a screen and reconstruct mesostructure surfaces from specularities using a single camera and a raster display. Chosen techniques are motivated and encoutered difficulties are highlighted. First an overview of the specifications and the adopted camera-screen setup is presented.

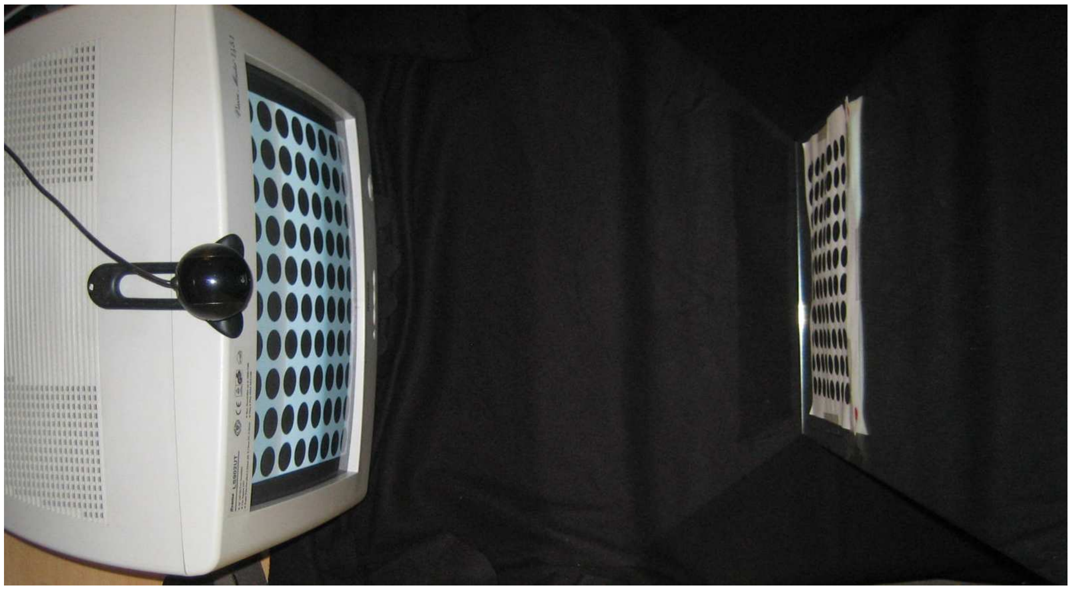
5.1 Specifications

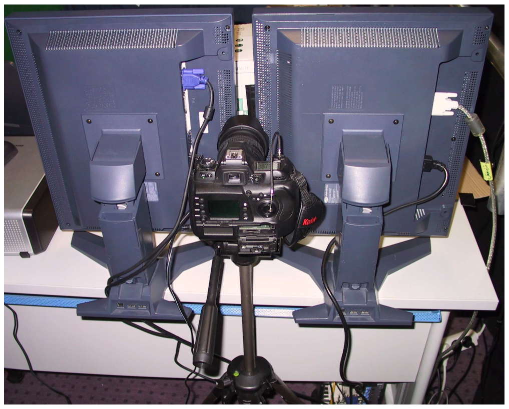
The implementation is written in the programming language C++ and makes use of the Qt [Tro], OpenGL [Gra] and DirectX [Mic] library.

5.2 Setup

During the implementation two different camera-screen setups have been adopted. The first one is a camera-screen setup where a single camera is placed on top of a single screen, as shown in figure 5.1(a). Most desktop computers and laptops use this setup and so it is probably the most familiar one. The charm of this setup is that it is obtainable at low prices and can be found in most homes. The object to be measured is placed in front of the screen.

Because calibration can be a difficult and tedious process, a second, simplified camera-screen setup is adopted. This setup makes use of two screens and a single camera. The camera is placed between the two screens, as shown in figure 5.1(b) and 5.1(c). The object to be measured is placed in front of and parallel to the two screens. The advantage of this setup is that camera, screens and object are placed parallel to each other. This makes it


(a)


(b)

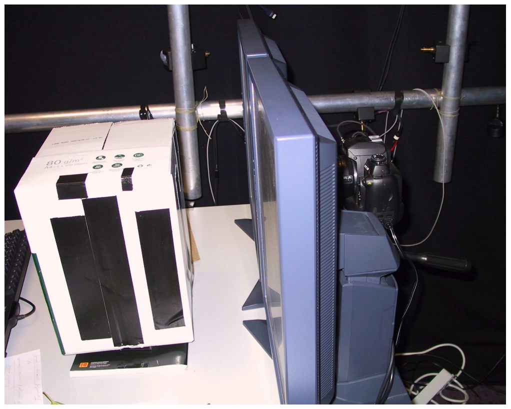

(c)

Figure 5.1: Camera-screen setups: (a) a single camera is placed on top of a single screen, (b-c) a single camera is placed between two screens.

possible to make a good estimate of the positions of camera, screens and object without a calibration step. This setup is particulary helpful when conducting fast experiments.

5.3 Multiplexed Illumination Models

Multiplexed illumination models are used throughout the whole implementation. They are utilized to generate the illumination patterns, for decoding and for the computation of individual light source positions. These models are based on the general multiplexed illumination principle, recall equation (4.1)

$$\mathbf{a}(x, y) = W \mathbf{i}(x, y)$$

and equation (4.2)

$$\widehat{\mathbf{i}}(x, y) = W^{-1} \mathbf{a}(x, y).$$

We have chosen to represent these models with an XML-file, an example is shown in figure 5.2. All the necessary information is provided by this file, using the following tags:

`<name></name>` contains a simple name describing the model.

`<type></type>` represents the type of multiplexed illumination model, currently supported are:

- `HADAMARD` model based on the work of Schechner et al. [SNB03],
- `CODEDILLUMINATION` our proposed model,
- `CODEDILLUMINATION_COMPLEMENTS` our proposed model extended with complements of the standard patterns.

`<horizontallights></horizontallights>` contains the number of horizontal lights (on the screen).

`<verticallights></verticallights>` contains the number of vertical lights (on the screen).

`<numpatterns></numpatterns>` contains the number of patterns, for our proposed model this will be equal to $O(\log_2 N)$ with N the number of light sources (horizontal lights $\times$ vertical lights). For the Hadamard-based model the number of patterns will be equal to N.

`<multiplex></multiplex>` contains a reference to the file containing the multiplex matrix (S).

`<demultiplex></demultiplex>` contains a reference to the file containing the demultiplex matrix (T). This last option is not obligatory.

```
<?xml version="1.0"?>

<multiplexedilluminaton>
    <name></name>
    <type></type>
    <horizontallights></horizontallights>
    <verticallights></verticallights>
    <numpatterns></numpatterns>
    <multiplex></multiplex>
    <demultiplex></demultiplex>
</multiplexedilluminaton>
```

Figure 5.2: XML-file for a multiplexed illumination model.

The fact that we defined these models based on the *general* multiplexed illumination principle allows us to support multiple types of multiplexed illumination models and thus makes the implementation easily expandable.

A separate application has been implemented for the creation of multiplexed illumination models and currently supports models of type `CODEDILLUMINATION` and the one extended with complements `CODEDILLUMINATION_COMPLEMENTS`.

5.4 Recording

This part of the implementation generates patterns from a given multiplexed illumination model. Each row of the multiplex matrix, which is given by the model, consists of *zeros* and *ones*. The first column corresponds to the first light source, the second to the second light source, etc. Each pattern is defined by a single row where a "0" corresponds to a disabled light source and a "1" corresponds to an enabled light source.

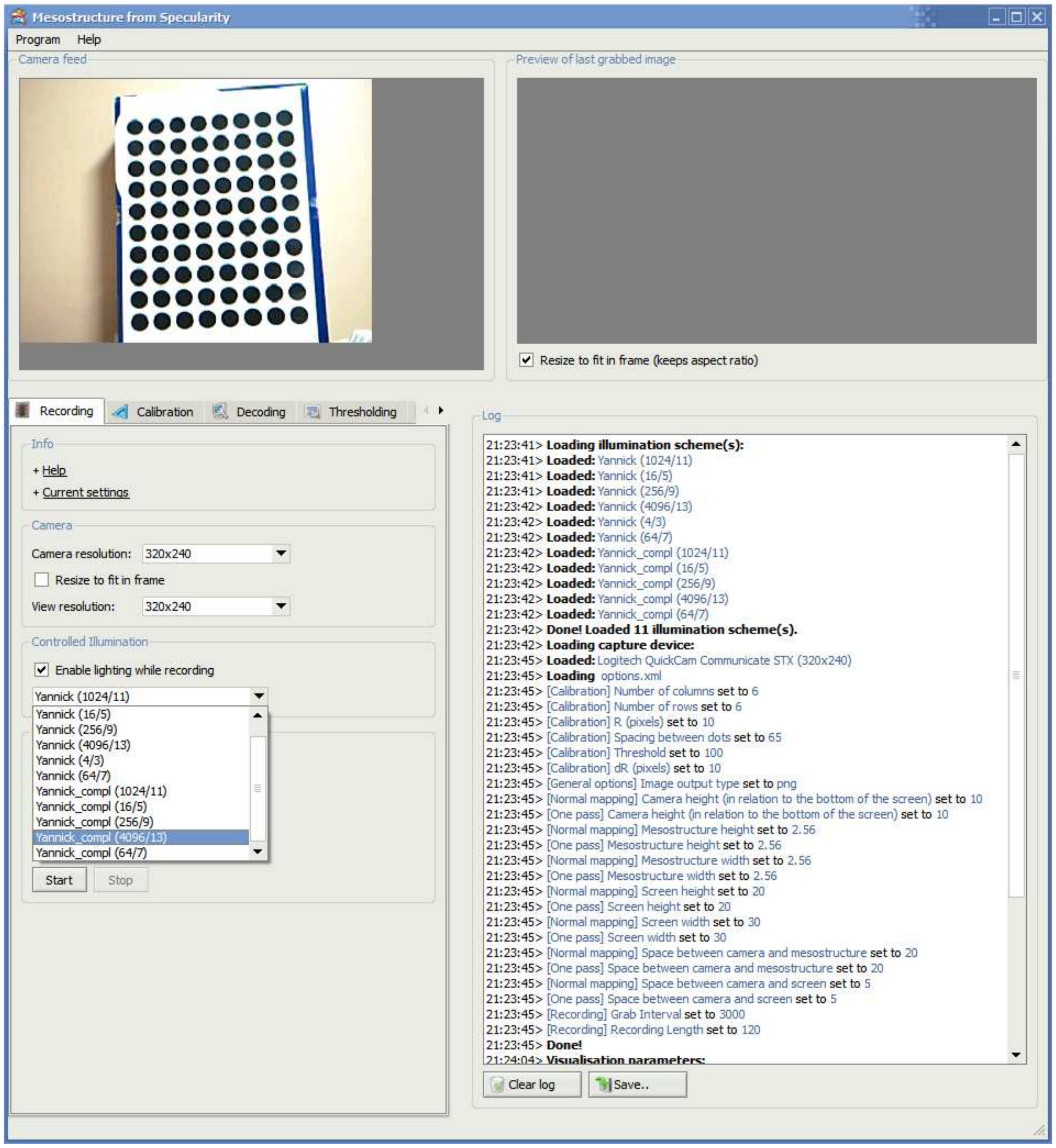
Subsequently these patterns are shown on the screen and used to illuminate the object to be measured. The recording interface and available options are shown in figure 5.3.

5.5 Calibration

When reconstructing a mesostructure surface from an image, the position and orientation of the light sources and the mesostructure surface with respect to the camera need to be computed. This is done through a calibration step.

5.5.1 Camera Calibration

To facilitate the calibration of the camera, *tclcalib* is used, a tool for camera calibration [Lab], which returns both internal and external parameters, as shown in figure 5.4. These parameters are used to construct the camera matrix. The internal parameters are given by:



[✓] Resize to fit in frame (keeps aspect ratio)

Camera resolution: 320x240

[ ] Resize to fit in frame

View resolution: 320x240

[✓] Enable lighting while recording

Yannick (1024/11)

Yannick (16/5)
Yannick (256/9)
Yannick (4096/13)
Yannick (4/3)
Yannick (64/7)
Yannick_compl (1024/11)
Yannick_compl (16/5)
Yannick_compl (256/9)
Yannick_compl (4096/13)
Yannick_compl (64/7)

```
21:23:41> Loading illumination scheme(s):
21:23:41> Loaded: Yannick (1024/11)
21:23:41> Loaded: Yannick (16/5)
21:23:41> Loaded: Yannick (256/9)
21:23:42> Loaded: Yannick (4096/13)
21:23:42> Loaded: Yannick (4/3)
21:23:42> Loaded: Yannick (64/7)
21:23:42> Loaded: Yannick_compl (1024/11)
21:23:42> Loaded: Yannick_compl (16/5)
21:23:42> Loaded: Yannick_compl (256/9)
21:23:42> Loaded: Yannick_compl (4096/13)
21:23:42> Loaded: Yannick_compl (64/7)
21:23:42> Done! Loaded 11 illumination scheme(s).
21:23:42> Loading capture device:
21:23:45> Loaded: Logitech QuickCam Communicate STX (320x240)
21:23:45> Loading options.xml
21:23:45> [Calibration] Number of columns set to 6
21:23:45> [Calibration] Number of rows set to 6
21:23:45> [Calibration] R (pixels) set to 10
21:23:45> [Calibration] Spacing between dots set to 65
21:23:45> [Calibration] Threshold set to 100
21:23:45> [Calibration] dR (pixels) set to 10
21:23:45> [General options] Image output type set to png
21:23:45> [Normal mapping] Camera height (in relation to the bottom of the screen) set to 10
21:23:45> [One pass] Camera height (in relation to the bottom of the screen) set to 10
21:23:45> [Normal mapping] Mesostructure height set to 2.56
21:23:45> [One pass] Mesostructure height set to 2.56
21:23:45> [Normal mapping] Mesostructure width set to 2.56
21:23:45> [One pass] Mesostructure width set to 2.56
21:23:45> [Normal mapping] Screen height set to 20
21:23:45> [One pass] Screen height set to 20
21:23:45> [Normal mapping] Screen width set to 30
21:23:45> [One pass] Screen width set to 30
21:23:45> [Normal mapping] Space between camera and mesostructure set to 20
21:23:45> [One pass] Space between camera and mesostructure set to 20
21:23:45> [Normal mapping] Space between camera and screen set to 5
21:23:45> [One pass] Space between camera and screen set to 5
21:23:45> [Recording] Grab Interval set to 3000
21:23:45> [Recording] Recording Length set to 120
21:23:45> Done!
21:24:04> Visualisation parameters:
```

Figure 5.3: The recording interface and the available options.

```
640.000000  # NDX - [pix] number of pixels in x direction
480.000000  # NDY
6.384000    # Sx - effective CCD chip size in horizontal direction
4.804800    # Sy -
0.963191    # Asp - aspect ratio
5.939386    # Effective focal length
353.051566  # Cx - [pix] principal point
245.274074  # Cy
-0.006705   # Rad1 - lens distortion parameters
0.000157    # Rad2
0.000786    # Tan1
-0.005542   # Tan2
-0.032590   # Inverse lens distortion parameters
0.001763
-0.000759
0.005345
0.001896
-0.000100
0.000001
-0.039093
-505.7523   # Tx - [mm] Extrinsic parameters
109.213337  # Ty
995.917779  # Tz
-170.885852 # Rx [rad]
-2.951705   # Ry
-6.014927   # Rz
```

Figure 5.4: Example output of tclcalib [Lab]: both internal and external parameters are computed.

- focal length: *effective focal length*,
- principal point: (C_x, C_y),
- number of pixels per unit distance in image coordinates: $\left(\frac{S_x}{NDX}, \frac{S_y}{NDX}\right)$

and the external parameters:

- rotation: (R_x, R_y, R_z),
- translation: (T_x, T_y, T_z).

This tool has been integrated in the implementation.

Tclcalib uses a dotted calibration pattern for calibration (Figure 5.5) and requires a number of pictures of a dotted pattern taken with the (stationary) camera, changing the orientation of the pattern for each picture. The dots are detected using a Hough transformation [DH72] and subsequently analyzed to obtain the camera parameters. The accuracy of the calibration increases together with the number of input images.

For each input image, *tclcalib* will produce an output file. The parameters are automatically extracted from this file and are used to generate the camera matrix. The camera

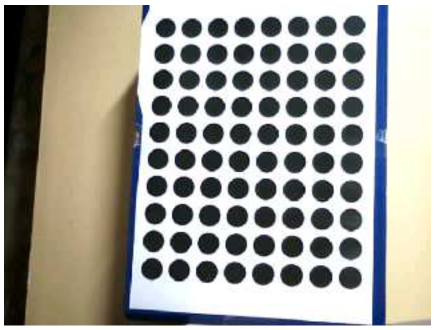


Figure 5.5: Example of calibration pattern.

calibration matrix K is stored as a 4×4 matrix:

$$K = \begin{bmatrix} \text{focal length} \cdot \frac{S_x}{NDX} & 0.0 & Cx & 0.0 \\ 0.0 & \text{focal length} \cdot \frac{S_y}{NDY} & Cy & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{5.1}$$

5.5.2 Screen Calibration

To retrieve the location of the screen with respect to the camera, a screen calibration step is necessary. In our implementation, this is done using a planar mirror. To retrieve the external parameters associated with the screen, a dotted pattern is displayed on the screen and a dotted pattern is attached to the mirror. The mirror is placed in such a way that a mirrored version of the screen pattern is visible for the camera. A single picture is taken which shows two patterns, the one attached to the mirror and the one displayed on the screen. To separate these two patterns, two red dots are placed between the patterns before recording. A thresholding step is used to detect these dots. Once these dots are detected they are connected and used to separate the patterns. This is illustrated in figure 5.6. These separated patterns can now be used to recover the external parameters. The external parameters are stored in a 4×4 rotation matrix R and a 4×4 translation matrix T:

$$\begin{aligned} c_x &= \cos R_x, & c_y &= \cos R_y, & c_z &= \cos R_z, \\ s_x &= \sin R_x, & s_y &= \sin R_y, & s_z &= \sin R_z, \end{aligned}$$

$$R = \begin{bmatrix} c_z c_y & s_z c_y & s_y & 0.0 \\ c_z s_y s_x + s_z c_x & c_z c_x - s_z s_y s_x & -c_y s_y & 0.0 \\ s_z s_x - c_z s_y c_x & s_z s_y c_x + s_x c_z & c_y c_x & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{5.2}$$

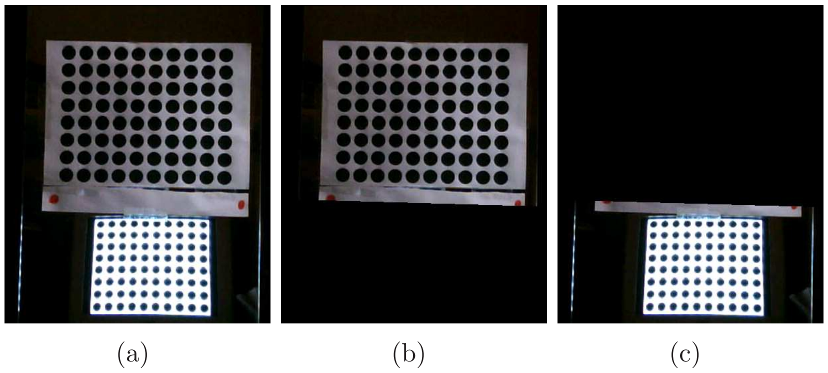


Figure 5.6: (a) The mirror with a calibration pattern and the mirrored calibration shown on the screen, (b) the mirror with a calibration pattern, (c) the mirrored calibration shown on the screen.

and

$$T = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.3}$$

The external parameters recovered for the screen pattern are associated with a *mirrored* version of the screen. To recover the actual screen-to-camera transformation instead of the mirrored-screen-to-camera transformation we will use the mirror matrix M, recall equation (2.13)

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and equation (2.14)

$$T_{MS} = MT_{MS'} = MT_{MC}T_{CS'} = MT_{CM}^{-1}T_{CS'}$$

where T_{CM}^{-1} is given by the external parameters associated with the mirror and $T_{CS'}$ is given by the external parameters associated with the *mirrored* screen.

5.6 Decoding

All the locations are now known and light patterns can be displayed on the screen while recording. The next step will be the recovery of single source images from images captured with multiplexed illumination. To do this the *demultiplex* matrix T, given by the chosen multiplexed illumination model, is used. Recall equation (4.11)

$$\mathbf{a} = S\mathbf{i}$$

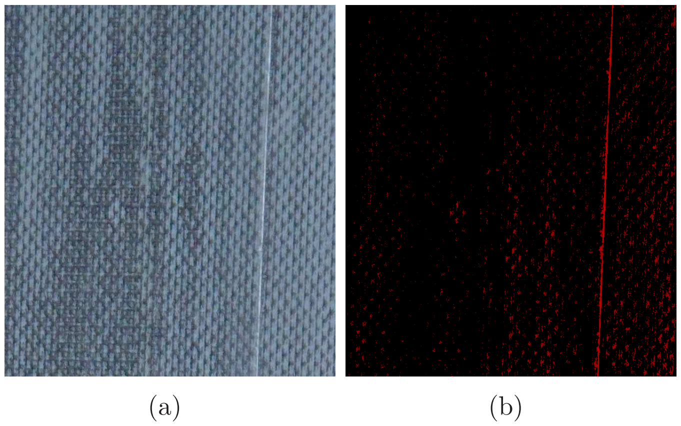


Figure 5.7: Thresholding example: (a) original image, (b) image thresholded, intensities greater than the threshold value are marked red.

where $\mathbf{a}$ represents the images captured with multiplexed illumination, S the multiplex matrix and $\mathbf{i}$ the single source images. The multiplex matrix relates the single source images to the images capture with multiplexed illumination. For decoding equation (4.26) is needed:

$$\widehat{\mathbf{i}} = T\mathbf{a}$$

where $\widehat{\mathbf{i}}$ refers to estimates of the original single source images ($\mathbf{i}$) and T represents the demultiplex matrix relating the images captured with multiplexed illumination to the single source images. This means that decoding is given by a matrix multiplication.

5.7 Thresholding

To recover specular highlights, thresholding is used. The thresholding is based on the intensity of each pixel, recall equation (3.3)

$$I = \frac{R + G + B}{3}$$

where the color of the pixel is expressed in the RGB color model. These intensities are thresholded against a given threshold value. This is illustrated by figure 5.7.

An additional *range* parameter can be entered. All thresholded intensities are compared to the highest intensity. If a thresholded intensity is smaller than the highest intensity minus the given range, this intensity will be discarded. This parameter is useful when thresholding multiple images at once. The threshold value can be used to set a minimum value. Consequently the range parameter will only gather intensities in a given range of the highest intensity, this makes the thresholding more adaptive to each image.

The resulting (thresholded) intensities are written to a bitmap. Note that the intensity values are preserved, in our implementation we use a single color channel (the red channel) to save the values. This bitmap is also referred to as a *specularity field*.

5.8 Normal Map Recovery

To recover the normal map, we rely on the thresholded specularities. The input for this algorithm are n specularity fields (bitmaps) where n is the number of *single* light sources. Each specularity field is read and for each pixel is checked whether a specularity is present (intensity > 0). If a specularity is present the intensity value is compared with the current intensity value associated with the pixel. At initialisation each pixel is associated with intensity value zero. If the intensity value is greater than the current intensity value associated with the pixel, the latter value is replaced with the former one. When a new intensity value is found, a new normal is computed as wel. Since the light source associated with the current specularity field is known, the location can be recovered. Note that we assume a calibration step has been taken and thus all the needed transformations are available. This means all the information needed to compute the normal is available:

$$L = \frac{P_{\text{light}} - P_{\text{meso}}}{\|P_{\text{light}} - P_{\text{meso}}\|} \tag{5.4}$$

where L represents the light vector, P_{light} the 3D-coordinates of the light source, P_{meso} the 3D-coordinates of the mesostructure surface,

$$V = \frac{P_{\text{cam}} - P_{\text{meso}}}{\|P_{\text{cam}} - P_{\text{meso}}\|} \tag{5.5}$$

V the view vector, P_{cam} the 3D-coordinates of the camera and

$$N = \frac{L + V}{\|L + V\|} \tag{5.6}$$

the resulting surface normal. All coordinates are assumed to be in the camera reference frame.

The resulting normals are written to a bitmap, or *normal map*, since the elements in a normal vector range from -1.0 to 1.0, these value are converted to a $[0.0,\ 1.0]$ range to avoid loss of data. We use the following conversion to convert the normal to a RGB color value:

$$\begin{array}{rcl} R & = & (X_N + 1) / 2 \\ G & = & (Y_N + 1) / 2 \\ B & = & (Z_N + 1) / 2 \end{array} \tag{5.7}$$

with normal $N = (X_N, Y_N, Z_N)$.

5.9 One-Pass Normal Map Recovery

This is the implementation of the pixel codification method which allows for a faster recovery. Unlike the normal recovery method discussed above this method takes the images captured while using multiplexed illumination as input. The algorithm also recovers the specularities itself.

Each input image is read and for each pixel the intensity (see equation (3.3)) is computed. If the intensity value is greater than a given threshold, the value is added to an intensity list associated with the pixel, otherwise a zero is added to the list. The value is also compared to the highest intensity value met for that pixel. The initial value for the highest intensity associated with each pixel is zero. If the intensity value is greater than the current highest intensity value, the latter value is replaced with the former one. After all the images are processed, we will have a list of intensities for each pixel including a reference to the highest intensity in this list. Note that this list has a size m equal to the number of patterns / input images.

The next step will be the encoding of each pixel using the intensity lists. For each pixel we pass through the intensity list associated with that particular pixel. Each pixel is assigned m bits, if a intensity value is greater than zero, we set the bit to 1, otherwise 0. The first pattern and the first intensity value in the list corresponds with the MSB (most significant bit), the last pattern and the last intensity value in the list with the LSB (least significant bit). We could have done the encoding in the previous step, but for the same reason we kept a reference to the highest intensity value in the list, we allow for an extra thresholding constraint, namely the *range* constraint. Using this constraint, an extra check will be done while encoding the pixels. While passing through the list for each pixel, if an intensity value is greater than zero *and* is greater than the highest intensity in the list minus the range value, we set the bit to 1, otherwise 0. Each pixel is now encoded.

We have reached the final step of the algorithm. The binary code associated which each pixel will be used to recover the light source number. Recall equation (4.30)

$$\text{light source number} = 2^{\text{number of patterns}} - \text{decimal notation of the binary code}$$

This means that the light source can be recovered. Again we assume a calibration step has been taken and thus all the needed transformations are available. The normal vector can be computed using equation (5.6). The resulting normal vectors are written to a bitmap, or *normal map*, after a conversion (see equation (5.7)).

5.10 Normal Map Visualization

This part of the implementation is used to create synthetic images visualizing a given normal map and is embedded in the main application. A normal map is given as input and a multiplexed illumination model is chosen. Subsequently the images lit by the multiplexed illumination are computed. All the single source images can be computed too. The output images are grayscale images where the color of each pixel corresponds to the computed intensity at that point. The intensity is computed using the Phong model for specular reflection [Pho73] which is given by

$$I = k_a I_a + \left(I_p / \left(d\right)\right) \left[k_d \left(L \cdot N\right) + k_s \left(V \cdot R\right)^n\right] \tag{5.8}$$

where

$$\begin{array}{lcl} k_s & : & \text{specular reflection,} \\ k_d & : & \text{diffuse reflection,} \\ k_a & : & \text{ambient reflection.} \end{array}$$

Since we only need to visualize the specular highlights we can simplify this equation to:

$$I = (V \cdot R)^n \,. \tag{5.9}$$

The view vector (V) can be computed from the mesostructure position and the camera position. These positions are known, assuming a calibration step has been taken. The reflection vector is given by

$$R = 2(N^T L)N - L. \tag{5.10}$$

Both the normal vector N and the light vector L are known. N is given by the input normal map. Recall equation (5.7)

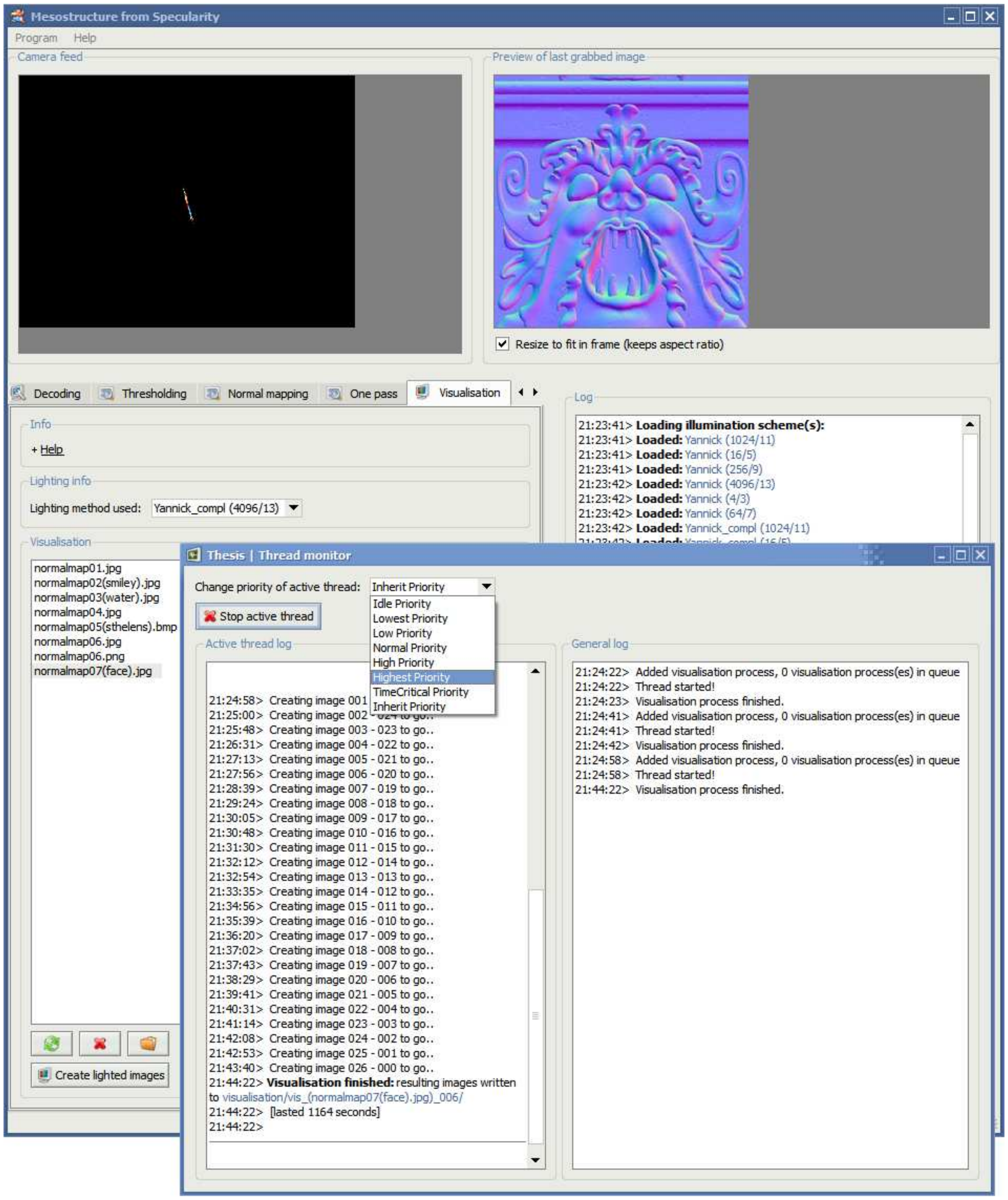
$$\begin{array}{lcl} R & = & (X_N + 1) \;/\; 2 \\ G & = & (Y_N + 1) \;/\; 2 \\ B & = & (Z_N + 1) \;/\; 2 \end{array}$$

with normal $N = (X_N, Y_N, Z_N)$. To recover the normal vector from a RGB color value we reverse this conversion:

$$\begin{array}{lcl} X_N & = & (R \times 2) - 1 \\ Y_N & = & (G \times 2) - 1 \\ Z_N & = & (B \times 2) - 1. \end{array} \tag{5.11}$$

The light vector L is computed from the selected multiplexed illumination model.

Since the visualization of these images is computational intensive, this has been implemented in a separate thread. Figure 5.8 shows the visualization interface.



✓ Resize to fit in frame (keeps aspect ratio)

Lighting method used: Yannick_compl (4096/13)

```
normalmap01.jpg
normalmap02(smiley).jpg
normalmap03(water).jpg
normalmap04.jpg
normalmap05(sthelens).bmp
normalmap06.jpg
normalmap06.png
normalmap07(face).jpg
```

```
21:23:41> Loading illumination scheme(s):
21:23:41> Loaded: Yannick (1024/11)
21:23:41> Loaded: Yannick (16/5)
21:23:41> Loaded: Yannick (256/9)
21:23:42> Loaded: Yannick (4096/13)
21:23:42> Loaded: Yannick (4/3)
21:23:42> Loaded: Yannick (64/7)
21:23:42> Loaded: Yannick_compl (1024/11)
```

Change priority of active thread: Inherit Priority

```
Idle Priority
Lowest Priority
Low Priority
Normal Priority
High Priority
Highest Priority
TimeCritical Priority
Inherit Priority
```

```
21:24:58> Creating image 001
21:25:00> Creating image 002 - 024 to go..
21:25:48> Creating image 003 - 023 to go..
21:26:31> Creating image 004 - 022 to go..
21:27:13> Creating image 005 - 021 to go..
21:27:56> Creating image 006 - 020 to go..
21:28:39> Creating image 007 - 019 to go..
21:29:24> Creating image 008 - 018 to go..
21:30:05> Creating image 009 - 017 to go..
21:30:48> Creating image 010 - 016 to go..
21:31:30> Creating image 011 - 015 to go..
21:32:12> Creating image 012 - 014 to go..
21:32:54> Creating image 013 - 013 to go..
21:33:35> Creating image 014 - 012 to go..
21:34:56> Creating image 015 - 011 to go..
21:35:39> Creating image 016 - 010 to go..
21:36:20> Creating image 017 - 009 to go..
21:37:02> Creating image 018 - 008 to go..
21:37:43> Creating image 019 - 007 to go..
21:38:29> Creating image 020 - 006 to go..
21:39:41> Creating image 021 - 005 to go..
21:40:31> Creating image 022 - 004 to go..
21:41:14> Creating image 023 - 003 to go..
21:42:08> Creating image 024 - 002 to go..
21:42:53> Creating image 025 - 001 to go..
21:43:40> Creating image 026 - 000 to go..
21:44:22> Visualisation finished: resulting images written
to visualisation/vis_(normalmap07(face).jpg)_006/
21:44:22> [lasted 1164 seconds]
21:44:22>
```

```
21:24:22> Added visualisation process, 0 visualisation process(es) in queue
21:24:22> Thread started!
21:24:23> Visualisation process finished.
21:24:41> Added visualisation process, 0 visualisation process(es) in queue
21:24:41> Thread started!
21:24:42> Visualisation process finished.
21:24:58> Added visualisation process, 0 visualisation process(es) in queue
21:24:58> Thread started!
21:44:22> Visualisation process finished.
```

Figure 5.8: The visualization interface.

6 Results

In this chapter, the results obtained from our implementation are shown, interpreted and discussed. Both synthetic and real-world results are presented to demonstrate the effectiveness of our proposed shape recovery algorithm.

6.1 Technical Parameters

For each result a technical summary is given. The following parameters will be adopted:

- **Light sources** refers to the *number* of individual light sources used.
- **Patterns** refers to the *number* of patterns needed to simulate all of the light sources.
- **Original single source images** refers to the *original* single source images generated from a given normal map.
- **Decoded single source images** refers to the single source images *decoded* from the images obtained using light patterns.
- **Input images** refers to the *number* of input images needed.
- **Normal map** refers to the *figure* in which the computed normal map is shown.
- **Orig** refers to the computed normal map obtained by using the original single source images.
- **Coded** refers to the computed normal map from images obtained by using light patterns.

- **Black pixels** refers to the *percentage* of pixels where no normal was found, i.e. no specular highlight was detected, with respect to the total amount of pixels.
- **Average error** refers to the *average vector distance* between the original normals and those from the computed normal map. Only non-black pixels are considered. The vector distance d between two vectors $v_1 = (x_1, y_1, z_1)$ and $v_2 = (x_2, y_2, z_2)$ is given by
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{6.1}$$
- **Highest error** refers to the *largest vector distance* between the original normals and those from the computed normal map. Only non-black pixels are considered.
- **Capture device** refers to the capture device that is used to capture the mesostructure surface. For detailed specifications of the capture device we refer to the appendix.
- **Resolution** refers to the resolution used by the capture device.

6.2 Normal Map Recovery with Decoded Images

Decoding is the process of recovering the single source images from the images obtained using light patterns. These estimates of the single source images are used instead of the original single source images to compute the normal map. Recall equation (4.26)

$$\widehat{\mathbf{i}} = T\mathbf{a}$$

where $\widehat{\mathbf{i}}$ represents the estimates of the single source images, T represents the demultiplex matrix and $\mathbf{a}$ represents the images obtained using light patterns. Hence the demultiplex matrix is used for decoding and is given by the chosen multiplexed illumination model.

6.2.1 Synthetic Results

The effectiveness of normal map recovery with decoded images will be illustrated by example using synthetic images. An example is presented using 256 single light sources, the following images are generated:

- The 256 original single source images,
- The 9 $((\log_2 256) + 1)$ images obtained by using the light patterns,
- The 256 decoded single source images using the 9 images obtained by using the light patterns.

Both the single source images and the decoded single source images are used to compute the normal map. The results are shown in figure 6.1. The normal map computed from the decoded single source images (Figure 6.1(c)) shows a lot of *black pixels* while the normal map computed from the original single source images (Figure 6.1(b)) shows a good estimate of the original normal map (Figure 6.1(a)). The average and highest errors of the computed normal maps do not differ much (table 6.1). This means that information has gone lost in the decoding process. This loss of information is caused by the demultiplex matrix. Recall equation (4.12)

$$\widehat{I} = TS = \begin{bmatrix} 1 & \leq 0 & \ldots & \leq 0 \\ \leq 0 & 1 & \ldots & \leq 0 \\ \vdots & \vdots & \ddots & \vdots \\ \leq 0 & \leq 0 & \ldots & 1 \end{bmatrix}$$

the alternative identity matrix that was used to construct T, the demultiplex matrix. The assumption was made that the *specular highlights* in the single source images are unaffected when these images are subtracted from each other. This assumption was based on:

1. The law of reflection which states that every specular highlight on a surface is precisely defined by the incident light direction, the surface orientation and the viewing direction. Hence a specular highlight is uniquely defined by a single light source.

2. Recall equation (4.9)

$$a^{single}(x,y) = i_{\tilde{s}}^{diffuse}(x,y) + i_{\tilde{s}}^{specular}(x,y) \gg i_{\tilde{s}}^{diffuse}(x,y)$$

   showing that the specular component is far greater than the diffuse component.

Although theoretically correct, practice shows this assumption tends to fail. Glossiness will *spill* highlights to nearby pixels, which are included in the subtractions causing loss of information. To alleviate this problem the synthetic images are generated to have a mirror-like surface. Recall that images are generated using the Phong model for specular reflection (equation (5.8)), simplified to equation (5.9)

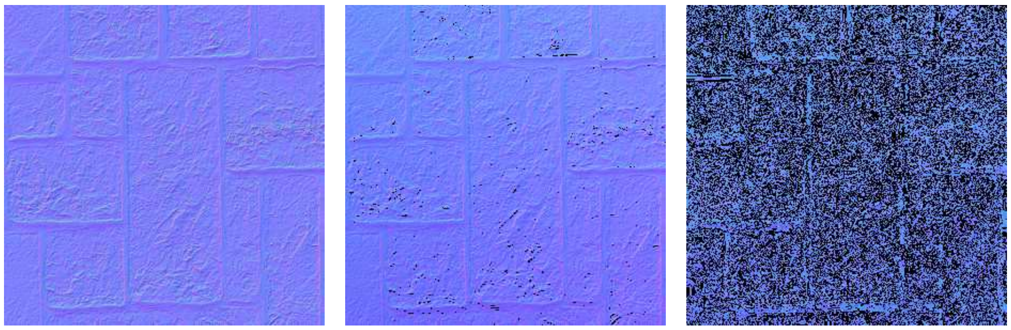
$$I = (V \cdot R)^n$$

where I represents the intensity, V the viewing direction, R the reflection and n is the Phong exponent. The Phong exponent determines the glossiness, a high value will reduce the *spilling* of highlights. Our examples are generated with a Phong exponent of 200.

Next to this glossiness effect another factor has to be taken in consideration. Although the specular component is far greater than the diffuse component, using a great amount of light sources will result in small diffuse values to become large values. In our current example 256 light sources are used, this means a smaller diffuse value can be subtracted 256 times and this will affect the specular highlights. This effect is illustrated by example in table 6.2 and graphically illustrated in figure 6.2.

	Original single source images	**Decoded single source images**
Input images	256	256
Normal map	Figure 6.1(b)	Figure 6.1(c)
Black pixels	0.65%	47.55%
Average error	0.33	0.30
Highest error	1.15	1.09

Table 6.1: Decoding: summary of figure 6.1.



(a) (b) (c)

Figure 6.1: Decoding: (a) original normal map, (b) normal map recovered from the original single source images, (c) normal recovered from the decoded single source images.

Light sources	4	15	64	256
Black pixels using original single source images	98.03%	33.33%	0.68%	0.65%
Black pixels using decoded single source images	98.03%	33.82%	14.67%	47.55%
Difference black pixels	0.00	0.49	13.99	46.90

Table 6.2: Decoding: loss of information increases when the amount of light sources increases.

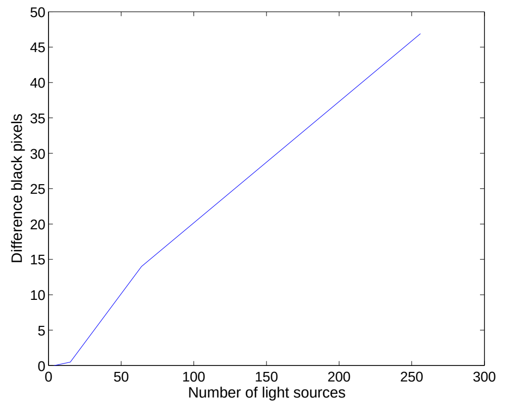


Figure 6.2: Decoding: plotting the difference of black pixels versus the number of light sources (table 6.2).

6.3 Normal Map Recovery with Pixel Codification

Pixel codification assigns a codeword to each pixel. If a specular highlight at a certain pixel is detected the codeword associated with that pixel will refer to the light source causing the highlight to appear. This technique eliminates the need for all the single source images. Consequently no decoding step is needed. Both synthetic and real-world results are presented.

6.3.1 Synthetic Results

To illustrate the effectiveness of this technique the results from the original single source images are compared to the results obtained with pixel codification. The following images are generated:

- The original single source images,
- The images obtained by using the light patterns,

for different amounts of light sources, namely 4, 16, 64 and 256 light sources. The results are shown in table 6.3. The average and highest errors stay in the same range because both techniques are affected by the same effect, namely the glossiness effect. Glossy reflections blur the unique mapping between image pixels and light source, resulting in ambiguous specularities. Therefore both techniques are compared on the basis of the amount of black pixels, i.e. the amount of not-detected highlights. A plot is created (Figure 6.3) of the amount of black pixels versus the amount of light sources. The blue line shows the results from the original single source images, the red line shows the results from the images obtained by using the light patterns. Both lines are almost inseparable, which means that our proposed technique successfully simulated N light sources with only $((\log_2 N) + 1)$

patterns. To illustrate this significantly decrease in acquisition time a plot is shown (Figure 6.4) of the amount of patterns versus the amount of light sources. An extra benefit of this significantly decrease in acquisition time is that the number of light sources can be increased to decrease the amount of black pixels as shown in figure 6.3.

Examples

Two examples are shown, both using 4096 light sources, simulated by only 13 patterns. The input images for the first example are shown in figure 6.5, the original normal map and the recovered normal map are shown in figure 6.7. The input images for the second example are shown in figure 6.6, the original normal map and the recovered normal map are shown in figure 6.8. A summary of the obtained results for the first example is shown in table 6.4, for the second example in table 6.5.

Improvements

An edge-preserving bilateral filter is used to de-noise the recovered normal maps [TM98]. Both the number of *black pixels* and the average error decreases. The highest error increases due to the fact that the *black pixels* are filled in using nearby values, without any information about the original normal.

Another way to improve the resulting normal maps is to add redundancy. This can be done by adding the complements of the patterns. The use of complements will double the Hamming distance between the indices of the individual light sources, which means the chance of picking the wrong individual light source for a particular specular highlight decreases. The amount of input images will double $2\left((\log_2 N) + 1\right)$ which is still a significantly decrease. The results obtained from using complements are shown in table 6.4 and table 6.5. The resulting normal maps are shown in figure 6.7 and 6.8, for the first and second example respectively. Both tables show the best results are obtained using the complements and applying the filter.

Light sources	4		15		64		256	
	Original	Coded	Original	Coded	Original	Coded	Original	Coded
Black pixels	98.03%	98.02%	33.82%	38.93%	0.68%	0.73%	0.66%	0.65%
Average error	0.74	0.11	0.31	0.11	0.16	0.24	0.11	0.29
Highest error	1.10	1.50	1.31	1.49	1.40	1.49	1.44	1.00

Table 6.3: The results from the original single source images are compared to the results obtained with pixel codification.

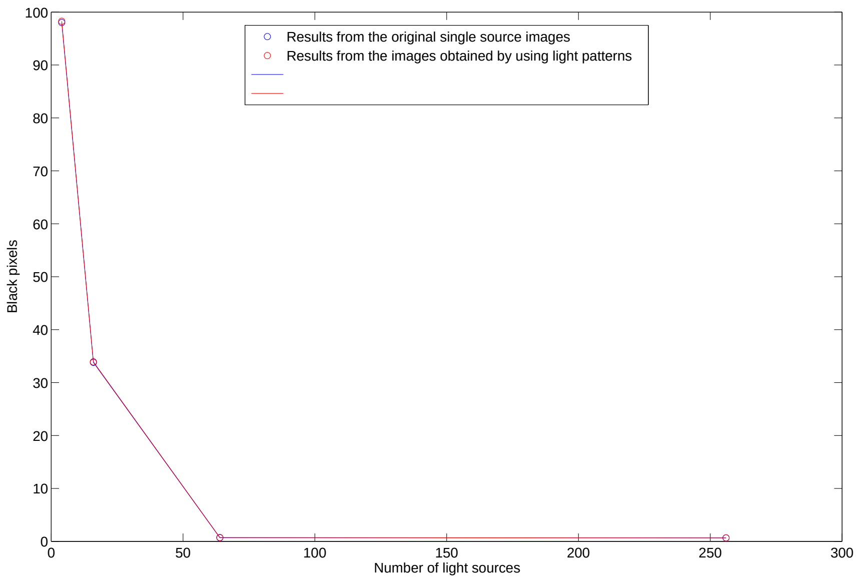


Figure 6.3: Single light sources versus patterns.

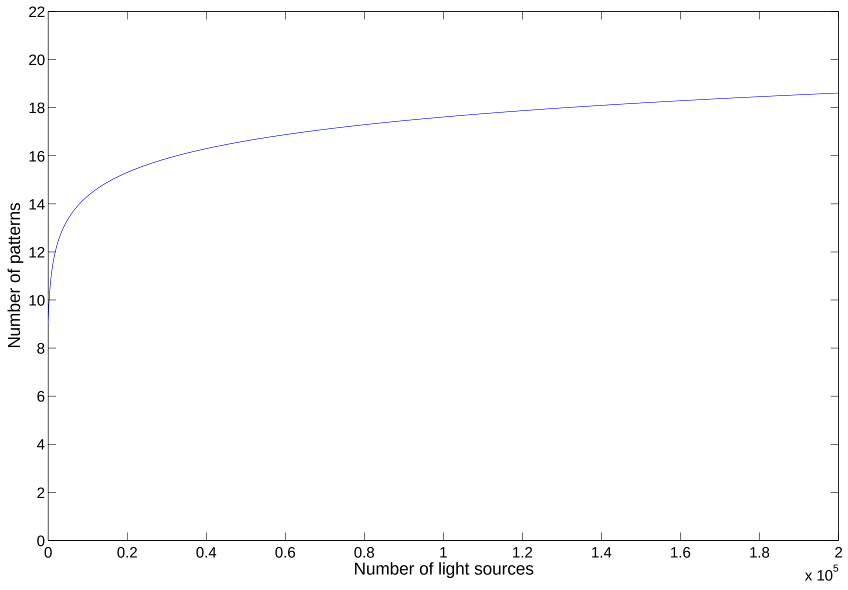


Figure 6.4: Simulating N light sources with $O\left(\log_2 N\right)$ patterns.

	Without complements		**With complements**	
	Not filtered	**Bilateral filtered**	**Not filtered**	**Bilateral filtered**
Light sources	4096		4096	
Patterns	13		26	
Normal map	Figure 6.7(b)	Figure 6.7(c)	Figure 6.7(d)	Figure 6.7(e)
Black pixels	0.53%	0.37%	0.53%	0.53%
Average error	0.32	0.27	0.26	0.19
Highest error	1.32	1.96	1.39	1.31

Table 6.4: Pixel codification synthetic results (wall).

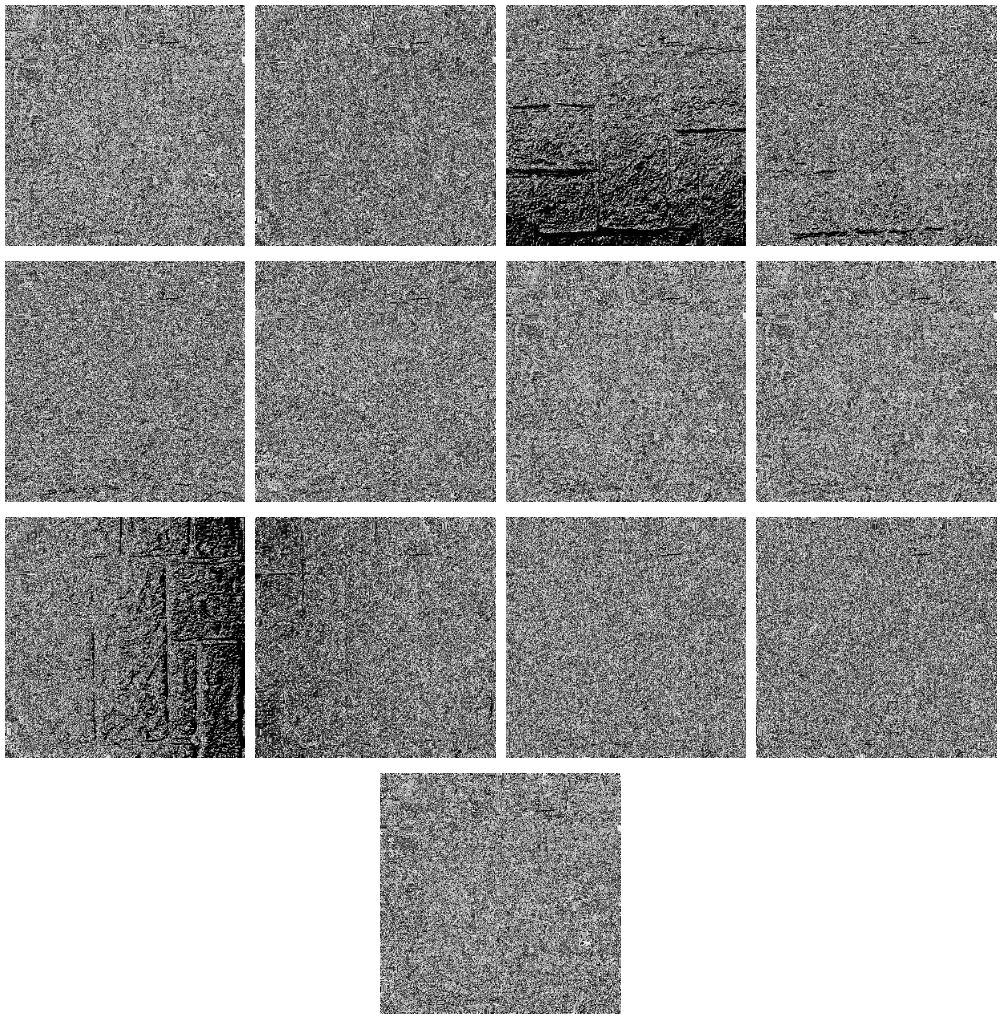


Figure 6.5: Pixel codification synthetic results (wall): simulating 4096 single light source images with 13 images lit by coded patterns.

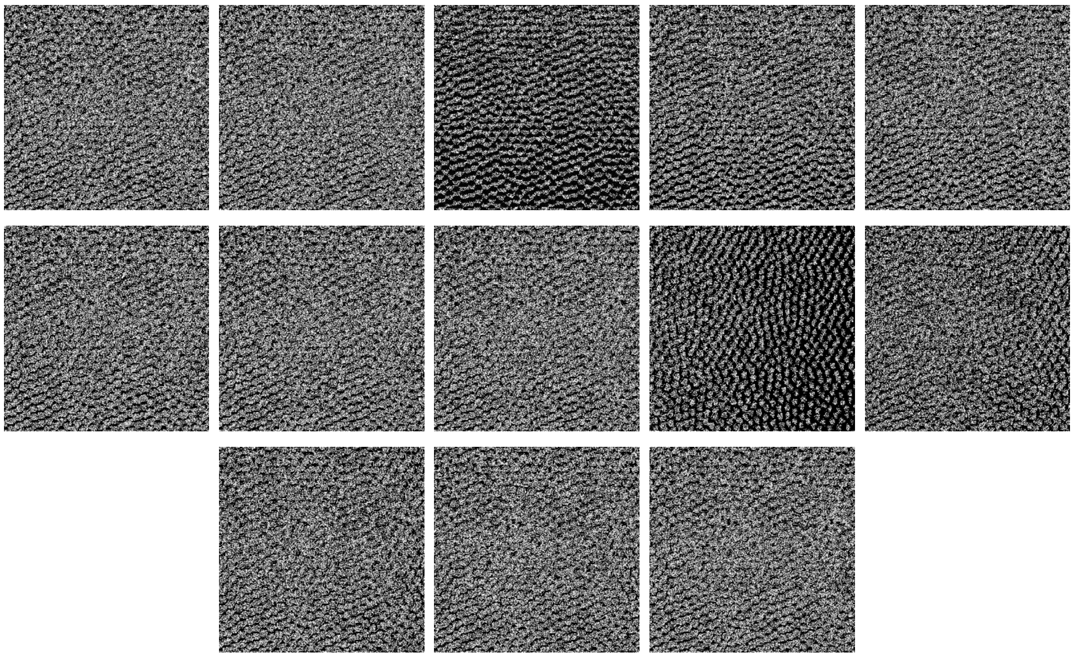
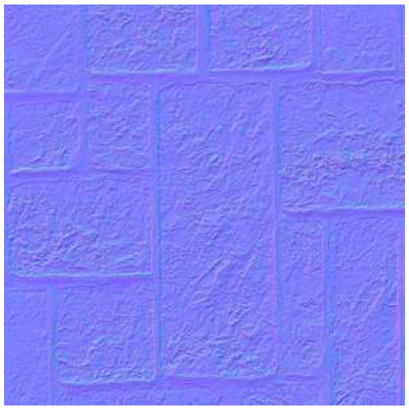
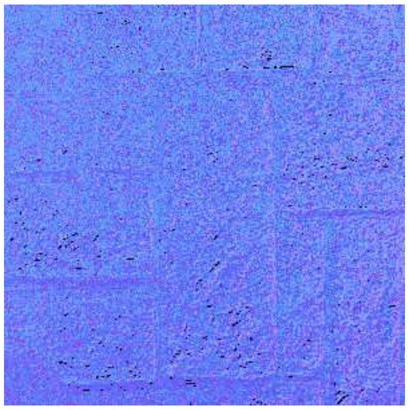


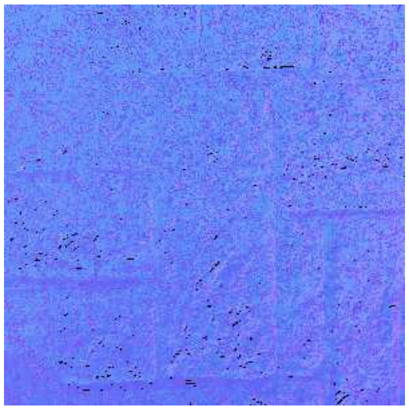
Figure 6.6: Pixel codification synthetic results (simple bumps): simulating 4096 single light source images with 13 images lit by coded patterns.

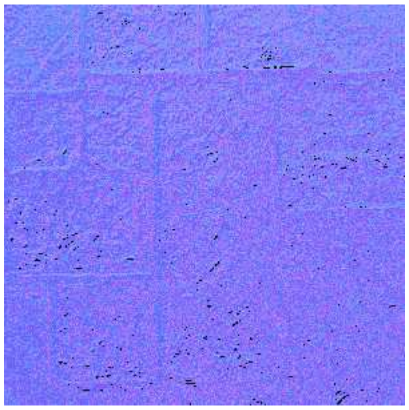
	Without complements		With complements	
	Not filtered	Bilateral filtered	Not filtered	Bilateral filtered
Light sources	4096		4096	
Patterns	13		26	
Normal map	Figure 6.8(b)	Figure 6.8(c)	Figure 6.8(d)	Figure 6.8(e)
Black pixels	13.44%	12.72%	13.44%	13.44%
Average error	0.29	0.27	0.25	0.22
Highest error	1.33	1.97	1.34	1.24

Table 6.5: Pixel codification synthetic results (simple bumps).


(a)


(b)


(c)


(d)

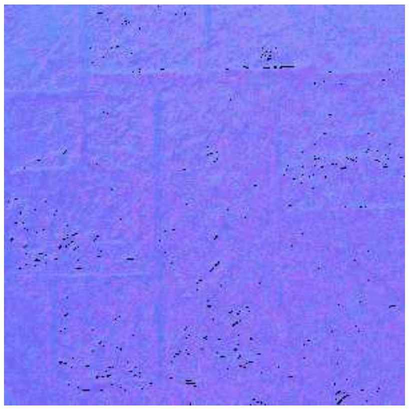

(e)

Figure 6.7: Pixel codification synthetic results (simple bumps): (a) original normal map, (b) recovered normal map, (c) recovered normal map filtered, (d) recovered normal map using complements, (e) recovered normal map using complements filtered.

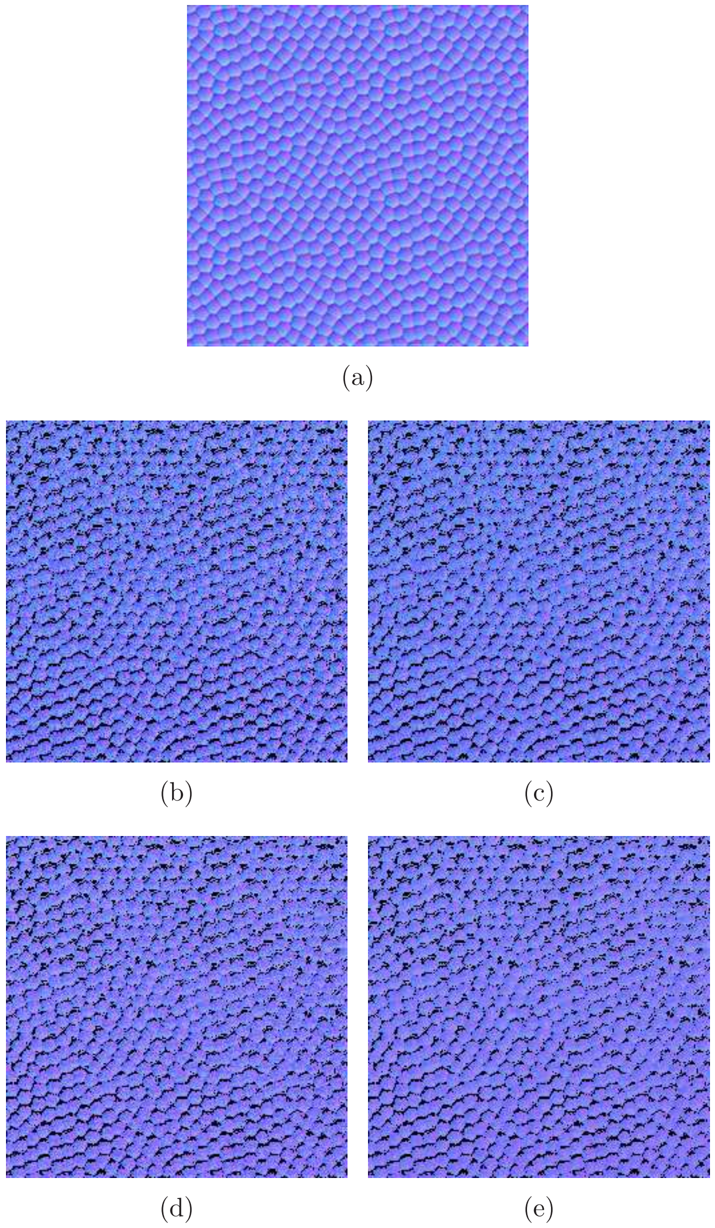


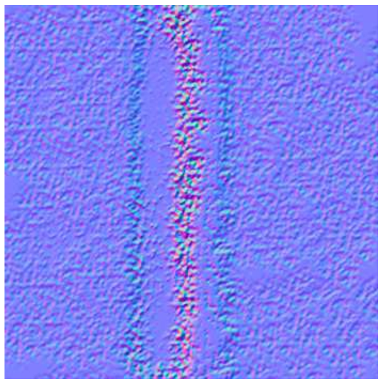
Figure 6.8: Pixel codification synthetic results (simple bumps): (a) original normal map, (b) recovered normal map, (c) recovered normal map filtered, (d) recovered normal map using complements, (e) recovered normal map using complements filtered.

6.3.2 Real-World Results

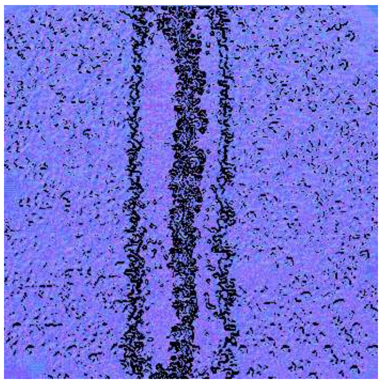
Since screen calibration using a planar mirror proved to be less robust than anticipated, a simplified setup was adopted. We use two juxtaposed LCD screens placing the camera in the center and parallel to the screens. The mesostructure surface is placed in front and parallel to the screens and camera. The use of two LCD screens also increases the range of possible normals that can be recovered. However, many *extreme* surface orientations are undetectable anyway due to self-shadowing, regardless of the extent of the screen. This is illustrated by figure 6.9: a normal map with extreme normals is shown (Figure 6.9(a)). The recovered normal map ((Figure 6.9(b))) shows a great amount of black pixels in the center, the place where most of the extreme normals are situated. An example of such an extreme surface orientation is given by figure 6.9(c). It shows a topview of the structure, illustrating the self-shadowing effect.

examples

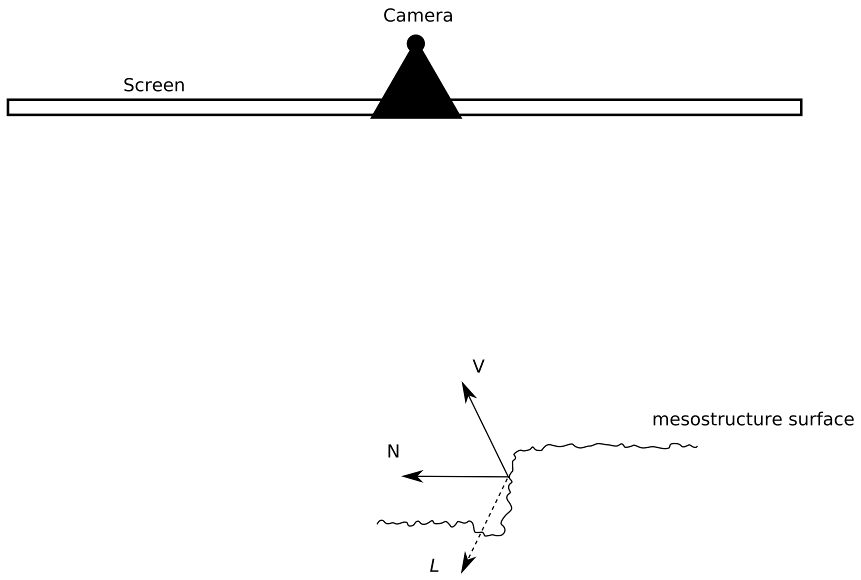
Two examples are presented, first a coin mesostructure surface is recovered, the input images are shown in figure 6.10 and the recovered normal map is shown in figure 6.11(a). A filter is applied to the recovered normal map yielding figure 6.11(b) which is used to bump map the first input image, as shown in figure 6.11(c). The results are summarised in table 6.6. The second example shows the recovery of a tape mesostructure surface, the input images are shown in figure 6.12. The recoverd normal map is shown in figure 6.13(a) and a filtered version in figure 6.13(b) which is used to bump map a teapot, this *tape-pot* is shown in figure 6.13(c). Again the results are summarised in table 6.13.



(a)



(b)



(c)

Figure 6.9: Extreme normals: (a) normal map with extreme normals, (b) recovered normal map, (c) top view of what goes wrong.

	Not filtered	**Filtered**
Capture device	Kodak DCS PRO 14n	
Resolution	4500×3000	
Light sources	4096	
Patterns	13	
Normal map	Figure 6.11(a)	Figure 6.11(b)
Black pixels	0.11%	0.00(3)%

Table 6.6: Pixel codification real-world results (coin).

	Not filtered	**Filtered**
Capture device	Kodak DCS PRO 14n	
Resolution	4500×3000	
Light sources	4096	
Patterns	13	
Normal map	Figure 6.13(a)	Figure 6.13(b)
Black pixels	0.61%	0.00%

Table 6.7: Pixel codification real-world results (tape).

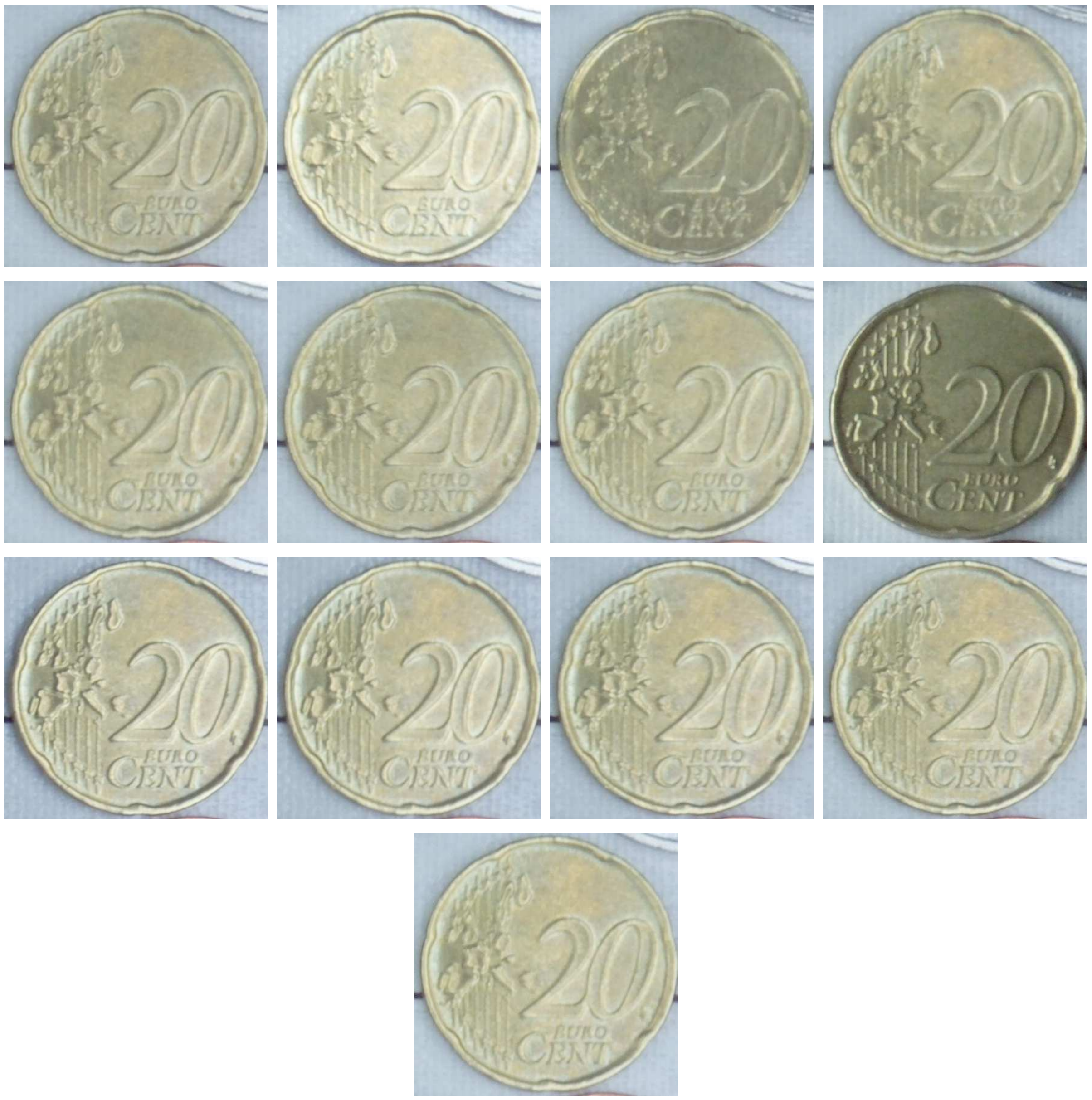


Figure 6.10: Pixel codification real-world results (coin): simulating 4096 single light source images with 13 images lit by coded patterns.

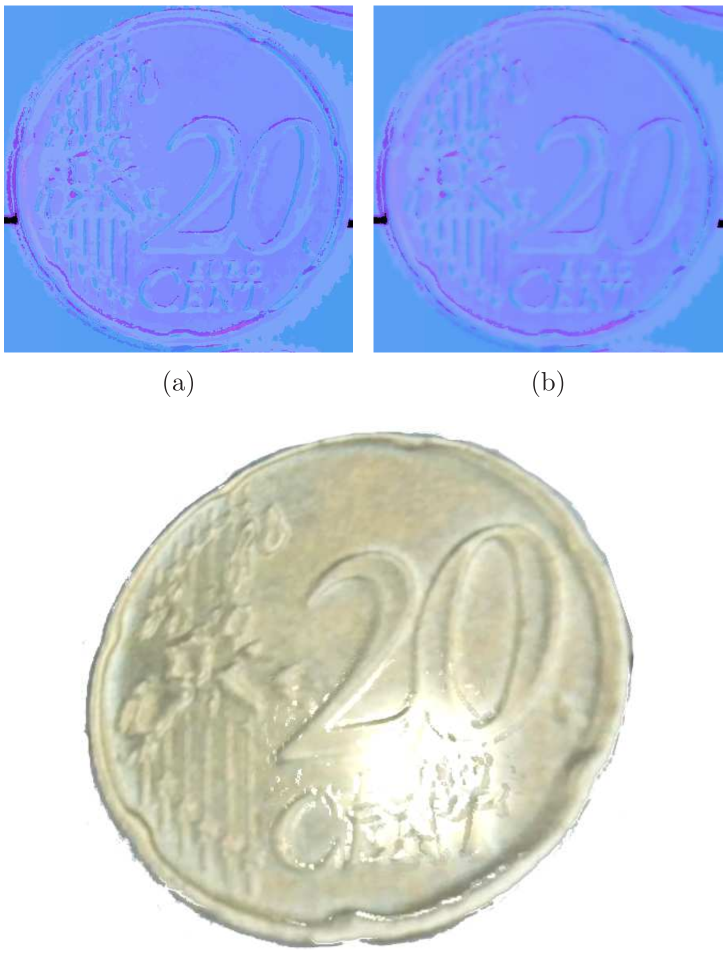


Figure 6.11: Pixel codification real-world results (coin): (a) recovered normal map, (b) recovered normal map filtered, (c) visualization of normal map.

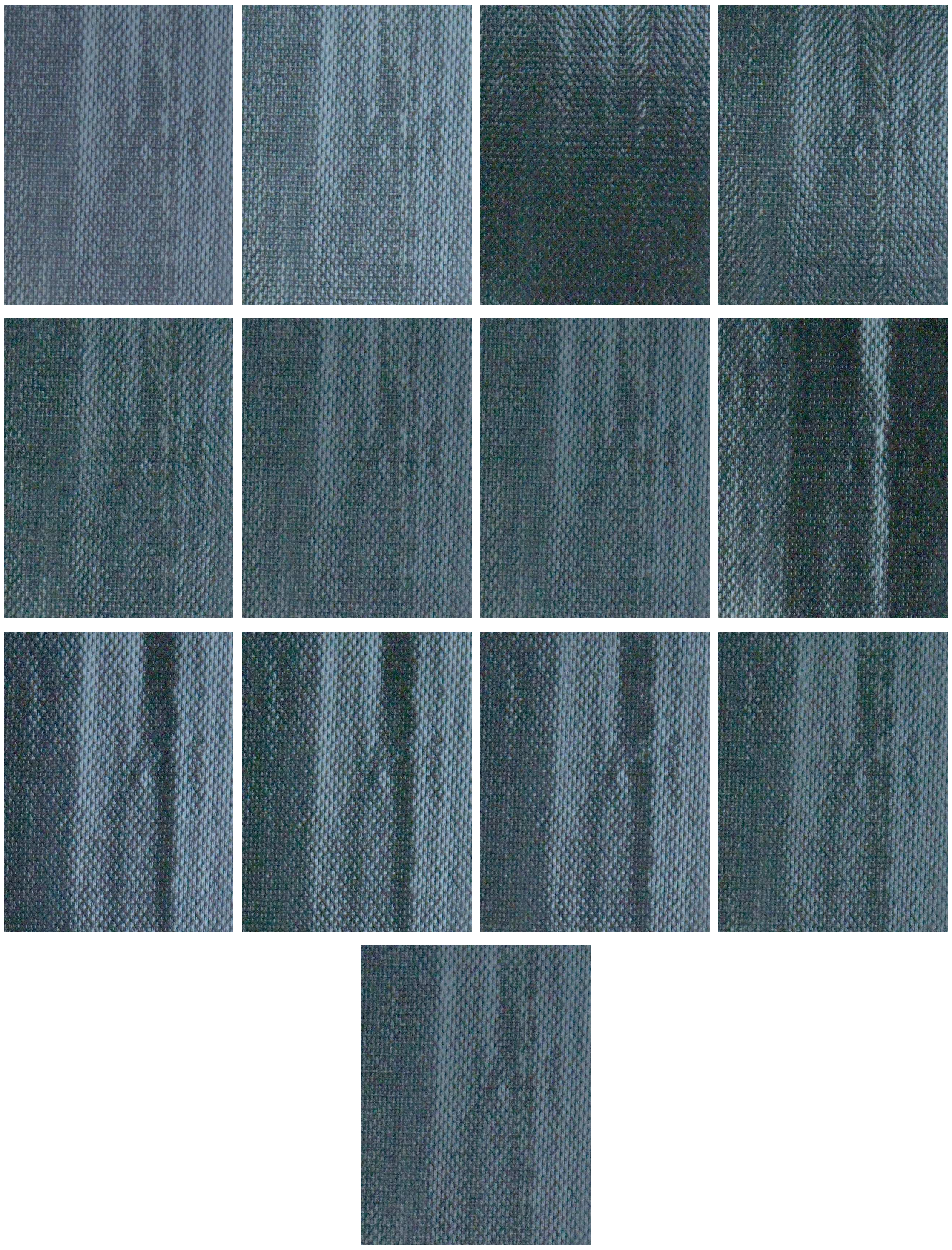
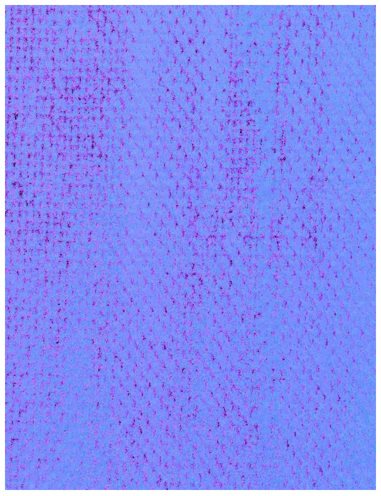
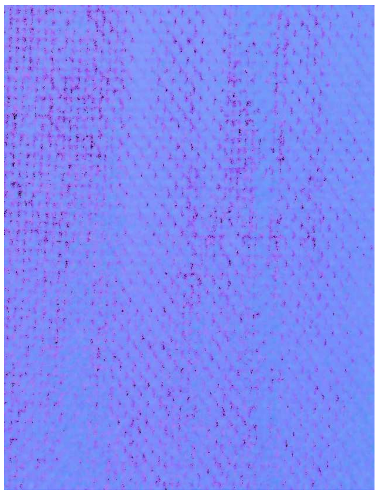


Figure 6.12: Pixel codification real-world results (tape): simulating 4096 single light source images with 13 images lit by coded patterns.


(a)


(b)

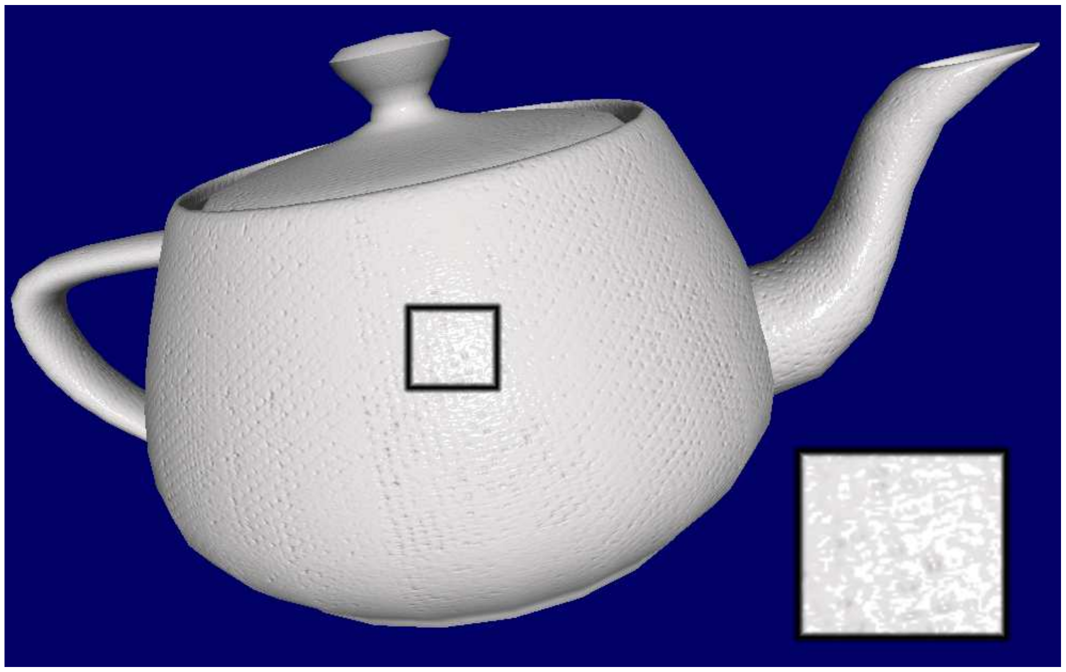

(c)

Figure 6.13: Pixel codification real-world results (tape): (a) recovered normal map, (b) recovered normal map filtered, (c) visualization of normal map.

7 Conclusions

This thesis has investigated the acquisition of fine-scale surface details, or mesostructure surfaces, from specularities using an inexpensive setup. When using a camera-screen setup where the camera is placed around the screen, usually on top of the screen, a calibration step is inevitable. Since the calibration step proved to be less robust than anticipated, a simplified setup was adopted. This setup uses two juxtaposed screens with the camera placed between them. Besides the geometrical benefit of this setup, the range of possible normals that can be recovered is increased. However, many extreme surface orientations are undetectable anyway due to self-shadowing, regardless of the range of the screen.

Further the effectiveness of our newly proposed multiplexed illumination scheme was demonstrated by example, including real-world objects. This technique is currently limited to surfaces with mirror-like reflections. Glossy reflections blur the unique mapping between image pixels and light source, resulting in ambiguous specularities. In other words, glossiness will *spill* highlights to nearby pixels, which might have a slightly different normal. Nevertheless desirable results were obtained from both synthetic and real-world mesostructure surfaces, while decreasing the acquisition time significantly. N light sources are simulated by $O(\log_2 N)$ patterns.

8 Future Work

The implementation of screen calibration using a planar mirror proved to be less robust than anticipated. It would be beneficial to perform experiments using a spherical mirror. This technique requires less user intervention and may provide a more robust solution.

To improve results the current shape recovery technique could be extended by adopting a smoothness constraint yielding more uniform results.

The use of a screen, being a *planar* illuminant, for illumination will always restrict the range of possible normals that can be recovered. A hemispherical illuminant could be adopted to increase this range. However, many extreme surface orientations are undetectable regardless of the range of the screen. A solution might be to use a combination of shape recovery techniques. Shape from specularity can be used for pixels where specular highlights are detected and shape from photometric stereo for pixels below the threshold for specular highlights.

A
Camera Specifications

A.1 Kodak DCS PRO 14n

- 13.89 MP CMOS Sensor (4560 x 3048)
  - 13.89 million total pixels (4560 x 3048)
  - 13.5 million recorded pixels (4500 x 3000)
- Bit depth: 36-bit color (12 bits per color) original capture
- 1:1 focal length (no magnification)
- Conforms to FCC Class B, CE Mark
- Class B Declaration, VCCI Class B Certified

Bibliography

[BJK07] R. Basri, D. Jacobs, and I. Kemelmacher. Photometric Stereo with General, Unknown Lighting. *International Journal of Computer Vision*, 72(3):239–257, 2007.

[BK87] KL Boyer and AC Kak. Color-encoded structured light for rapid active ranging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):14–28, 1987.

[BS03] T. Bonfort and P. Sturm. Voxel carving for specular surfaces. *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 591–596, 2003.

[CGS06] T. Chen, M. Goesele, and H.P. Seidel. Mesostructure from Specularity. *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 2*, pages 1825–1832, 2006.

[DH72] R.O. Duda and P.E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[Eye] Eyetronics. 3D Scanning Company. http://www.eyetronics.com/.

[FB81] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[FC88] R.T. Frankot and R. Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):439–451, 1988.

[FHB07] Y. Francken, C. Hermans, and P. Bekaert. Screen-Camera Calibration using a Spherical Mirror. *Canadian Conference on Computer and Robot Vision (CRV)*, 2007. (to appear).

[FTA04] RW Fleming, A. Torralba, and EH Adelson. Specular reflections and the perception of shape. *Journal of Vision*, 4(9):798–820, 2004.

[Fun05] N. Funk. Using a Raster Display Device for Controlled Illumination. Master's thesis, University of Alberta, 2005.

[FY07] N. Funk and Y.-H. Yang. Using a Raster Display for Photometric Stereo. *Canadian Conference on Computer and Robot Vision (CRV)*, 2007. (to appear).

[Gra] Silicon Graphics. OpenGL (Open Graphics Library). http://www.opengl.org/.

[GW02] R.C. Gonzalez and R.E. Woods. *Digital Image Processing.* Prentice Hall, second edition, 2002.

[Hor70] B. K.P. Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1970.

[Hor90] B.K.P. Horn. Height and gradient from shading. *International Journal of Computer Vision*, 5(1):37–75, 1990.

[HS05] A. Hertzmann and S.M. Seitz. Example-based photometric stereo: shape reconstruction with general, varying BRDFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1254–1264, 2005.

[HZ04] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, second edition, 2004.

[J$^+$01] S. Joaquin et al. Pattern codification strategies in structured light systems. *Instituto de Informática, Universidad de Girona, España*, 2001.

[JY] Bouguet J.-Y. Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[Lab] Stanford Computer Graphics Laboratory. tclcalib - A tool for camera calibration. http://www.soe.ucsc.edu/davis/projects/tclcalib/.

[LR85] C.H. Lee and A. Rosenfeld. Improved methods of estimating shape from shading using the light source coordinate system. *Artificial Intelligence*, 26(2):125–143, 1985.

[Mic] Microsoft. DirectX. http://msdn.microsoft.com/directx/.

[PEN82] A.P. PENTLAND. Finding the illuminant direction. *Optical Society of America, Journal*, 72:448–455, 1982.

[Pho73] Bui Tuong Phong. *Illumination for computer-generated images.* PhD thesis, 1973.

[SNB03] YY Schechner, SK Nayar, and PN Belhumeur. A theory of multiplexed illumination. *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 808–815, 2003.

[SP01] S. Savarese and P. Perona. Local Analysis for 3D Reconstruction of Specular Surfaces. *Proceedings of CVPR*, 2:738–745, 2001.

[SP02] S. Savarese and P. Perona. Local Analysis for 3D Reconstruction of Specular Surfaces–Part II. *European Conf. on Computer Vision*, 2002.

[SS02] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 47(1):7–42, 2002.

[SS03] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. *cvpr*, 01:195, 2003.

[Sto] D. Stoyanov. Camera Calibration Tools. http://www.doc.ic.ac.uk/˜dvs/calib/main.html.

[Td91] HD Tagare and RJP deFigueiredo. A theory of photometric stereo for a class of diffusenon-Lambertian surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(2):133–152, 1991.

[TLGS05] M. Tarini, H.P.A. Lensch, M. Goesele, and H.P. Seidel. 3D acquisition of mirroring objects using striped patterns. *Graphical Models*, 67(4):233–259, 2005.

[TM98] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *Proceedings of the Sixth International Conference on Computer Vision*, page 839, 1998.

[Tro] Trolltech. Qt – C++ class library and tools for cross-platform development and internationalization. http://www.trolltech.com/products/qt.

[Woo78] RJ Woodham. Photometric stereo: A reflectance map technique for determining surface orientation from image intensity. *Proc. SPIE*, 155:136–143, 1978.

[Woo89] R.J. Woodham. Photometric method for determining surface orientation from multiple images. *Mit Press Series Of Artificial Intelligence Series*, pages 513–531, 1989.

[WWC$^+$05] M. Waschbüsch, S. Würmlin, D. Cotting, F. Sadlo, and M. Gross. Scalable 3D video of dynamic scenes. *The Visual Computer*, 21(8):629–638, 2005.

[ZC91] Q. Zheng and R. Chellappa. Estimation of illuminant direction, albedo, and shape from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):680–702, 1991.

[ZCS03] Li Zhang, Brian Curless, and Steven M. Seitz. Spacetime stereo: Shape recovery for dynamic scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 367–374, June 2003.

[ZFA97] J.Y. Zheng, Y. Fukagawa, and N. Abe. 3D surface estimation and model construction from specular motion in image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):513–520, 1997.

[ZM00] J.Y. Zheng and A. Murata. Acquiring a complete 3D model from specular motion under the illumination of circular-shaped light sources. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):913–920, 2000.

[ZTCS99] R. Zhang, P.S. Tsai, J.E. Cryer, and M. Shah. Shape from Shading: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.

[ZWCS99] D.E. Zongker, D.M. Werner, B. Curless, and D.H. Salesin. Environment matting and compositing. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 205–214, 1999.