Examensarbete utfört i Bildkodning
av
Lars-Åke Waldemarsson

LiTH-ISY-EX--06/3907--SE
Linköping 2006
Holografisk Video

Examensarbete utfört i Bildkodning
vid Linköpings tekniska högskola
av

Lars-Åke Waldemarsson

LiTH-ISY-EX--06/3907--SE

Examinator: Robert Forchheimer

Linköping, 19 Juni, 2006
Abstract

This thesis is based upon an article that describes a method for creating holographic video. The intention is to recreate this method.

First a study in the field of holography is presented. This includes how the eye can see depth and the different displays for creating 3D images. There are two types of holography, optical and computer generated, this thesis exploits the latter.

The foundation for creating holographic images is diffraction (bending of rays of light) and interference between rays of light. In optical holography rays of light from an object and a reference beam interfere with each other. The interference pattern is caught on photographic film. Later a hologram of the object can be created by illuminating the film with the same reference beam. Computer generated holography simulates optical holography through a Fourier transform. With a numerical description of the object as input the transform has an output of an interference pattern. When this interference pattern is used as an input in the DLP it will control the rays of light in a way that a hologram is created.

A Spatial Light Modulator (SLM) is needed to create 3D holographic images. The one studied have Texas Instruments DLP. Its main component is the Digital Micromirror Device (DMD) which is a small chip with micromirrors in a grid pattern; each micromirror can tilt off or on. When directing a light source on the DMD each micromirror can choose whether or not to reflect its part of the light.

The next part is recreating the method. Matlab is used to compute the transform. The input is two 2D images slightly apart from each other in the z-direction, this is the object. The output is a 2D image representing the interference pattern of the object.

The result is an interference pattern for a given object. Although several optimizations are made it still takes a lot of time to compute an interference pattern even for rather small objects. Only interference patterns for nonmoving objects are presented, but calculating for moving objects is possible. To get holographic video around 24 images per second is required. This is feasible but very time consuming at least with the program presented here.

Keywords: computer generated holography, Fourier transform, diffraction, DMD
Sammanfattning

Detta examensarbete utgår ifrån en artikel i vilken en metod för att skapa holografisk video beskrivs. Syftet med arbetet är att återskapa denna metod. Metoden bygger på projicering av hologram med hjälp av delar från en projektor, en laser och några linser.


 För att återge tredimensionella holografiska bilder så behövs en SLM ("Spatial Light Modulator"). Den SLM som används här är Texas Instruments DLP ("Digital Light Processing"). Denna återfinns i DLP-projektörer i vilken huvudkomponenten är en DMD ("Digital Micromirror Device"). En DMD är ett datorchip bestående av mikroskopiska små speglar i ett rutmönster. DMD:n belyses i projektorn av en lampa och därigenom fördelas mikrospegel kan vinklas mot resp. från ljuskällan och därigenom förhindra sitt lilla ljusknippe vidare eller inte.


Stor vikt har lagts vid optimering av detta program genom att utnyttja Matlabs styrka i matrisoperationer och att förenkla beräkningen för de punkter som i hologrammet är genomskinliga, dvs. de punkter som inte hör till objektet.


Nyckelord: datorgenererade hologram, fouriertransform, diffraction, DMD
Förord

Detta arbete har till stor del varit en studie inom holografisk video. Arbetet utgår ifrån en artikel om holografisk video. För förståelse inom ämnet har ett antal personer hjälpt mig och jag skulle här vilja framföra min tacksamhet för det.

Professor Robert Forchheimer, för uppslaget till examensarbetet och god hjälp på vägen. Bala Munjuluri och Michael Huebschman för stor hjälp av förståelse för transformen i arbetet, bakomliggande optiken och hela processen. Emil Hällstig för handledning och hjälp med lämplig litteratur. Tack alla!

Linköping i april 2006

Lars-Åke Waldemarsson
# Innehållsförteckning

<table>
<thead>
<tr>
<th>Kapitel</th>
<th>Titel</th>
<th>Sidnummer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INLEDNING</td>
<td>7</td>
</tr>
<tr>
<td>1.1</td>
<td>Bakgrund</td>
<td>7</td>
</tr>
<tr>
<td>1.2</td>
<td>Syfte</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>PROBLEMBESKRIVNING</td>
<td>9</td>
</tr>
<tr>
<td>2.1</td>
<td>Utrustning</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Mål</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Begreppningar av målen</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>TEORETISK BAKGRUND</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Djupseende</td>
<td>11</td>
</tr>
<tr>
<td>3.2</td>
<td>3D-Displayer</td>
<td>11</td>
</tr>
<tr>
<td>3.3</td>
<td>Diffakteterat ljus</td>
<td>11</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Fresneldiffraktion</td>
<td>14</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Fraunhoferdiffraktion</td>
<td>15</td>
</tr>
<tr>
<td>3.4</td>
<td>Holografi</td>
<td>15</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Optisk holografi</td>
<td>16</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Datorgenererad holografi</td>
<td>16</td>
</tr>
<tr>
<td>3.5</td>
<td>SLM</td>
<td>17</td>
</tr>
<tr>
<td>3.6</td>
<td>Texas Instrument DLP</td>
<td>17</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Verklig och Virtuell bild</td>
<td>18</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Fasändring av ljusväv</td>
<td>20</td>
</tr>
<tr>
<td>3.7</td>
<td>Huygen-Fresnels Princip</td>
<td>20</td>
</tr>
<tr>
<td>3.8</td>
<td>Fresnellapproximationen</td>
<td>21</td>
</tr>
<tr>
<td>3.9</td>
<td>Helmholtz ekvation</td>
<td>22</td>
</tr>
<tr>
<td>3.10</td>
<td>Transform för datorgenerade hologram</td>
<td>22</td>
</tr>
<tr>
<td>3.10.1</td>
<td>Härledning av Fresneltransformen</td>
<td>24</td>
</tr>
<tr>
<td>3.10.2</td>
<td>Transformen</td>
<td>25</td>
</tr>
<tr>
<td>3.10.3</td>
<td>Härledning av transformen</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>GENOMFÖRANDE</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Programmeringsdel</td>
<td>27</td>
</tr>
<tr>
<td>4.2</td>
<td>Matlabkoden</td>
<td>27</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Inläsning och behandling av bilderna</td>
<td>27</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Skapande av $U(x',y',z')$, $U_s(x,y)$, samt konstanter</td>
<td>28</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Tidigare transformberäkning</td>
<td>28</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Optimering 1</td>
<td>29</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Optimerad transformberäkning 1</td>
<td>30</td>
</tr>
<tr>
<td>4.2.6</td>
<td>Skapande av transformbild</td>
<td>30</td>
</tr>
<tr>
<td>4.2.7</td>
<td>Optimering 2</td>
<td>30</td>
</tr>
<tr>
<td>4.2.8</td>
<td>Optimerad transformberäkning 2</td>
<td>32</td>
</tr>
<tr>
<td>4.2.9</td>
<td>Optimering 3</td>
<td>33</td>
</tr>
<tr>
<td>4.2.10</td>
<td>Optimerad transformberäkning 3</td>
<td>34</td>
</tr>
<tr>
<td>4.2.11</td>
<td>Optimering 4</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>RESULTAT</td>
<td>37</td>
</tr>
<tr>
<td>5.1</td>
<td>Bilder till $V'$</td>
<td>37</td>
</tr>
</tbody>
</table>
Figurer och formler

Figurer
Figur 1. Objektet ......................................................................................................................... 9
Figur 2. Utrustning för att skapa verklig bild ........................................................................... 10
Figur 3. Konstruktiv interferens ............................................................................................... 12
Figur 4. Destructiv interferens ................................................................................................. 13
Figur 5. Diffraktionsmönster från laser .................................................................................... 13
Figur 6. Diffraktionsmönster för vågor .................................................................................... 13
Figur 7. Diffraktion enkel spalt ............................................................................................... 14
Figur 8. Diffraktion hinder ....................................................................................................... 15
Figur 9. Skapning av interferensmönster ................................................................................. 16
Figur 10. DMD ........................................................................................................................... 18
Figur 11. Utrustning för att skapa verklig bild ......................................................................... 19
Figur 12. Utrustning för att skapa virtuell bild ......................................................................... 19
Figur 13. Huygen-Fresnel diffraction ...................................................................................... 21
Figur 14. Konstruktionsprocess för datorgenererad holografi ................................................. 23
Figur 15. Rekonstruktionsprocess för datorgenerade hologram ............................................. 24
Figur 16. Uppdelning av DMD:n ............................................................................................. 31
Figur 17. Bild till V’ vit cirkel .................................................................................................. 37
Figur 18. Bild till V’ svart kvadrat ........................................................................................... 37
Figur 20. Bild till V’ vit kvadrat ............................................................................................... 37
Figur 21. Transformbild för optimering 1 ................................................................................ 38
Figur 22. Matlabprofil för optimering 1 ................................................................................... 39
Figur 23. Matlabprofil för optimering 3 ................................................................................... 40
Figur 24. Matlabprofil för optimering 4 ................................................................................... 40

Formler
Formel 1. Huygen-Fresnels princip .......................................................................................... 20
Formel 2. Huygen-Fresnels princip omskrivning ..................................................................... 21
Formel 3. Binomialutveckling av kvadratrot ........................................................................... 21
Formel 4. Omskrivning av $r_{01}$ ............................................................................................ 21
Formel 5. Approximation av $r_{01}$ ......................................................................................... 22
Formel 6. Fresnelapproximationen av Huygen-Fresnels princip ............................................. 22
Formel 7. Helmholtzekvationen ............................................................................................... 22
Formel 8. Fresneltransform ....................................................................................................... 23
Formel 9. Transform för datorgenererade hologram 1 ............................................................. 23
Formel 10. Huygen-Fresnels princip för en punktkälla ............................................................ 24
Formel 11. Huygen-Fresnels princip för en punktkälla med Fresnelapproximation 1 .......... 24
Formel 12. Huygen-Fresnels princip för en punktkälla med Fresnelapproximation 2 .......... 25
Formel 13. Transform för datorgenerade hologram 2 ............................................................. 25
Formel 14. Huygen-Fresnels princip med Fresnelapproximation 1 ......................................... 25
Formel 15. Huygen-Fresnels princip med Fresnelapproximation 2 ......................................... 26
Formel 16. Fasändringen pga. den konvexa linsen .................................................................. 26
Formel 17. Huygen-Fresnels princip med Fresnelapproximation 3 ......................................... 26
Formel 18. xArray och yArray ................................................................................................. 32
Formel 19. zprimArray, xprimArray och yprimArray ............................................................. 32
1 Inledning

Mitt intresse för bildkodning trädde fram mer tydligt då jag studerade ämnet i kurserna: TSBK30 Källkodning samt fortsättningskursen TSBK01 Bildkodning och datakompression. Detta gjorde mitt val av examensarbete betydligt lättare. Jag tog kontakt med Robert Forchheimer på ISY här på Linköpings Universitetet och han gav mig ett intressant förslag på ett examensarbete. Det rörde en avhandling med namnet ”Dynamic holographic 3-D image projection”\(^1\), gjord på ”the University of Texas Southwestern Medical Center” i Dallas. I korthet handlar denna avhandling om en metod att skapa holografisk video.

1.1 Bakgrund


1965 så kom Brown och Lohmann fram till att processen i optisk holografie kan återskapas i datorn. Ett datagenererat hologram är en numerisk representation av ett interferensmönster.\(^3\) Teorin för holografisk video har alltså funnits ganska länge. Det är först nu på senare år, då datorkapaciteten har växt till en sådan grad att det är möjligt att göra de komplexa beräkningar som krävs för att skapa datagenererade hologram, och inte minst holografisk video.

Det har sedan en tid tillbaka existerat flera metoder som skapar en illusion av en tredimensionell (rörlig) bild. Hit kan man räkna t.ex. de grön-röda glasögon som på vissa TV-program gav denna illusion. Den form av holografisk video som studeras i detta arbete är ”verklig” 3-D bildåtergivning.

Framtida tillämpningar finns bland annat inom:
- Sjukvården: tredimensionella röntgenbilder
- Militären: projicera upp en tredimensionell bild av terrängen framför sig i cockpiten
- Underhållning: projicera upp tredimensionella TV-sändningar och filmer hemma i vardagsrummet

1.2 Syfte

Efter att studerat avhandlingen (som nämns i inledningen) och ämnet holografisk video närmare, beslutades att återskapa scenariot i avhandlingen. Syftet med detta examensarbete är alltså att verifiera den metod för holografisk video som presenteras i avhandlingen, genom att återskapa deras resultat här på universitetet i Linköping.

\(^1\) Michael L. Huebschman, Bala Munjuluri och Harold R. Garner, ”Dynamic holographic 3-D image projection”, Optics Express, Vol. 11, No. 5 (2003) s. 437-445
\(^3\) B. R. Brown och A. W. Lohmann, ”Complex Spatial Filtering with Binary Masks”, Applied Optics, Vol. 5, No. 6 (1966) s. 967-969
2 Problembeskrivning

Detta examensarbete behandlar en programmeringsdel och en praktisk del. Programmeringen består i att utifrån två eller flera tvådimensionella bilder beräkna ett interferogram. Objektet, utifrån vilket vi skall skapa ett hologram, består av två eller flera bilder på olika avstånd från varandra (i z-led), dessa bygger således upp en ”tredimensionell rymd” enligt figur 1. Interferogrammet beräknas enkelt utryckt genom att beräkna skillnaden mellan de olika bildernas (som bygger upp hologrammet) fas.

Interferogrammet används sedan i en s.k. DLP, där ”beräknas” hur hologrammet ska visas. DLP står för "Digital Light Processing", denna enhet utgörs i stort sett av en DMD ("Digital Micro-mirror Device"). Detta är ett litet mikrochip bestående av en mängd mikroskopiska speglar. Styrenheten i DLP:n läser av informationen i interferogrammet och ställer därefter in DMD:n utifrån denna information. Hologrammet kan slutligen visas genom att belysa DMD:n med en laser och med hjälp av ett par linser.

Således består den praktiska delen i att sätta upp och konfigurera DLP:n med en laser och två linser, samt en yta där hologrammet kan fångas upp.

2.1 Utrustning

Den praktiska delen i detta examensarbete kräver följande utrustning:
- en dator med ett standardgrafikkort
- Texas Instrument 1024x768 (XGA) DLP
- 15mW HeNe laser (633nm)
- ”spatial filter” (för att filtrera i rymdkoordinater)
- konvex lins med fokus på 10cm
- konvex lins med fokus på 40cm
- frostad glasplatta för att fånga upp olika plan i hologrammet
- ett genomskinligt block av Agarose gelé för att fånga upp hela hologrammet

De två sistnämnda punkterna behandlar föremål varpå man kan fånga upp hologrammet. Målet var först och främst att lyckas fånga upp olika plan av hologrammet och om tid ges kan möjligheterna med den sistnämnda kontrolleras.
Figur 2 visar utrustningen, det som där kallas för bildrekonstruktion är där man placerar den frostade glasplattan eller Agarose gelen.

2.2 Mål
Detta examensarbete har två huvudmål. Först ska ett datorprogram skapas för att beräkna fram det numeriska interferensmönstret (interferogrammet) för två bilder enligt figur 1. Därefter ska holografiska bilder skapas utifrån det numeriska interferensmönstret och med den utrustning som används i avhandlingen som tidigare refereras till. Om tid ges skall även holografisk video skapas.

2.3 Begränsningar av målen
Det visade sig mycket svårt att införskafta den utrustning som krävdes för att genomföra den praktiska delen. Framförallt DLP:n vilken måste plockas från en DMD-projektor, vilket i sin tur innebär kassering av en sådan. Därför genomfördes inte den praktiska delen utan mer vikt tas vid teorin och optimering av beräkningen av interferogrammet.
3 Teoretisk bakgrund


3.1 Djupseende

Om vi tittar på de signaler som ögat sänder vidare till hjärnan, så kan man dela upp dem i anpassning (fokusering) och sammanstrålning (triangulering mellan ögonen). En annan faktor som spelar in är vårt stereoseende och avståndet mellan ögonen som gör att det ena ögat ser ett föremål från en liten annan vinkel gentemot det andra ögat. En tredje faktor är när vi fysiskt förflyttar oss så ser vi föremålet från en annan observationspunkt och får då en uppfattning av djup. Ögat kan även uppfatta djup i tvådimensionella bilder genom överlappningstekniker där en del av bilden är dold av en annan överlappande del.

Efter denna kortfattade studie om ögats djupseende så studeras de apparater som utnyttjar detta.

3.2 3D-displayer

Vi tänker oss 3D-displayer som elektroniska visuella apparater. Dessa kan få oss att uppfatta djup genom att utnyttja egenskaperna för sammanstrålningen mellan ögonen (triangulering) och att ögonen ser föremål från olika vinklar. I vissa fall kan man även vid ändring av observationspunkt och fokusering uppfatta djup med dessa 3D-displayer.


3.3 Diffraktoriat ljus

Ljustrål som träffar ett föremål reflekteras från det och ett vågfält sänds ut från föremålet. Holografi utnyttjar att ljustrål kan böjas och interferera med varandra för att skapa kopior av olika föremåls vågfält.

Om vi tänker oss att ljus passar genom ett hål, antingen en spalt med viss spaltbredd eller en cirkulär öppning med en viss diameter, så sprids ljuset. Desto mindre öppning desto större blir spridningen av ljuset. Kravet för diffraction är att öppningen är i samma storleksordning som våglängden. Förhållandet mellan ljusets våglängd ($\lambda$) och spaltbredden ($b$) ger vinkeln ($\theta$) där diffraction inträffar.


radianer) vilket ljusets sprids med: \( \theta = \frac{\lambda}{b} \). På liknande sätt fås spridningsvinkeln för en cirkulär öppning: \( \theta = 1.2\frac{\lambda}{d_c} \), där \( d_c \) är cirkelns diameter.\(^6\)

Den cirkulära öppningen i föregående formel kan ersättas med en lins, där \( d_c \) då är linsens diameter. Om vi tänker oss en ljuskälla i punktform som avbildas i en lins så fås radien på den avbildade ljuskällan (\( r \)) av följande formel: \( r = 1.2\frac{\lambda f}{d_c} \), där \( f \) är linsens fokallängd och \( d_c \) är linsens diameter.

Hur en partikel ändrar sin riktning då en annan partikel kolliderar med den beskrivs av uttrycket ”scattering” (spridning). Diffraktion beskriver ”scattering” av strålning (t.ex. ljus) från en mängd olika källor.\(^7\) Diffraktionsmönster fås genom att lagra de diffrakterade strålar som produceras. Diffraktion är en typ av våginterferens.\(^8\)

När en våg (från någon form av strålning) träffar en partikel så sprider partikeln denna våg likformigt i alla riktningar. I en kristall är atomerna formade i ett regelbundet mönster. Träffas en kristall av en våg så kan vissa av de vågor som sprids i kristallen adderas, medan andra tar ut varandra. De vågor som inte tar ut varandra kommer att reflekteras ut från kristallen och skapa diffrakterade strålar.

\[ \text{Figur 3. Konstruktiv interferens} \]

Detta fenomen bygger på superposition. Ta t.ex. två vågor med samma amplitud och frekvens men olika fas (mellan vågorna). Konstruktiv interferens fås då de båda vågorna är i fas med varandra, då förstärker de varandra till en våg med dubbla amplituden. Detta illustreras i figur 3 ovan, där ser man även de båda vågorna (röd resp. blå) och den resulterande vågen (gul). För att fenomenet ska illustreras tydligare är vågorna inte riktigt i fas i figur 3. På motsvarande sätt fås destruktiv interferens när de båda vågorna är ur fas, följd bland att de tar ut varandra. Detta illustreras i figur 4, där är vågorna inte riktigt ur fas.

När vi behandlar ljusvågor så uppkommer diffraktion bara från koherent ljus. Två ljusvågor från två källor med konstant fasskillnad och samma frekvens (samma färg, monokromatiskt ljus) är koherenta med varandra.\(^9\) I detta arbete används koherent ljus, bestående av den 15mW HeNe laser (633nm, röd) som nämnts tidigare.

---


Figur 4. Destruktiv interferens


Figur 5. Diffraktionsmönster från laser

Figur 6. Diffraktionsmönster för vågor

Diffraktion kan illustreras genom att t.ex. släppa igenom en våg genom två smala öppningar. Som tidigare nämnts så uppkommer konstruktiv respektive destruktiv interferens om varje öppning har lägre bredd än vågens våglängd. I figur 6 kan man föreställa sig vattenvågor som kommer in genom två smala spalter. Desto fler spalter (med ovan krav) desto smalare (linjeformad) blir centralmaximum och desto mindre intensitet får de olika maxima.

I figur 5 används också två smala spalter, benämna avståndet mellan centra på de två spalterna (eller cirkulära öppningarna) till $d$ enligt figur 6. Formlerna för att beräkna spridningsvinkeln som togs upp i föregående stycke behandlade en öppning i form av en spalt eller cirkulär öppning. För att generalisera dessa till flera spalter (cirkulära öppningar) så används följande formler: $\sin \theta = \frac{m \lambda}{d}$ respektive $\sin \theta = \frac{m.22\lambda}{d}$, där $m=0, \pm 1, \pm 2, \ldots$
När \( m=0 \) fås de strålar som är parallella mot den inkommande vågen, likt sträckan \( c \) i figur 8. Här ligger alltså centralmaxima och detta kallas ”nollte” diffraktionsordningen. Där första maxima ligger \((m=\pm1)\) kallas för första diffraktionsordningen osv.\(^{10}\)

Figur 7 föreställer en vågfront (blå) som når en öppning i form av en spalt med bredden \( b \). Avståndet \( e \) fås som skillnaden mellan \( d \) och \( c \). Alltså \( f \) och \( c \) är lika långa \((d=e+f)\).

![Figu7](image)

**Figur 7. Diffraktion enkel spalt**

Vi tittar på det ljus som kommer från \( A \) och \( B \) i spaltens ändpunkter och hur det fångas upp på en skärm i punkt \( C \). Enligt Huygens princip\(^{11}\) så kan varje punkt på en våg ses som ett eget vågsystem från vilket strålar går ut i alla riktningar. Enligt superpositionsprincipen så tar vågorna från \( A \) och \( B \) ut varandra i \( C \) om \( e=n\lambda/2 \), där \( n \) är ett ojämnt heltal. Alltså destruktiv interferens uppstår, konstruktiv interferens fås på liknande sätt då \( e=n\lambda \), där \( n \) är ett heltal. När det skiljer en halv våglängd mellan vågorna från \( A \) och \( B \) i punkten \( C \) så är de helt ur fas. När det skiljer en hel våglängd mellan vågorna från \( A \) och \( B \) i punkten \( C \) är de helt i fas.

Det innebär tunga matematiska beräkningar att beräkna fram diffraktionsmönster. För att förkorta beräkningsstiden och simplifiera de matematiska beskrivningarna för diffraktion så har Fresnel- och Fraunhoferapproximationer införts.\(^{12}\)

### 3.3.1 Fresnel-diffraktion

Figur 8 är ett litet enklare scenario än i figur 7, här finns ingen spalt. Huygens princip gäller givetvis fortfarande. Beroende på vilket \( b \) vi väljer så kan punkter på vågfronten i \( B \) ligga i fas eller ur fas med punkter på vågfronten i \( A \), sett från \( C \). Kalla det \( b \) för vilket \( A \) och \( B \) är i fas för första gången för \( x \). Säg för enkelhet att \( x=b \). Detta \( x \) avgränsar vågfronten i den så kallade första fresnelzonen. Symmetri ger att vi också får en första fresnelzon i underkant, nedanför sträckan \( c \).

Vi tänker oss nu att vi för in en avgränsare i form av t.ex. en liten plåtbit i överkant (se figur 8). När avgränsaren når över punkten \( B \) så börjar ljuset böjas i punkt \( C \), jämför med hur vattenvågorna i figur 6 böjs. Desto längre den förs mot \( A \) desto mer böjs ljuset och desto lägre

---


\(^{12}\) Joseph W. Goodman, Introduction to Fourier Optics, s. 63
blir intensiteten i $C$. Intensiteten når inte noll i $C$ förrän plåtbiten passerat både den ”övre” första fresnelzonen samt den ”nedre” första fresnelzonen. För att förtydliga så täcker alltså plåtbiten både den övre respektive nedre första fresnelzonen i detta skede. Diffraktion av denna typ kallas Fresneldiffraktion. Eftersom intensiteten avtar relativt snabbt så är Fresneldiffraktion oväsentlig på stora avstånd (många gånger större än spaltbredden).

![Diagram](image.png)

Figur 8. Diffraktion hinder

### 3.3.2 Fraunhoferdiffraktion


De formler för att räkna ut spridningsvinkeln som visades i början av detta stycke är approximationer av de som används inom Fraunhoferdiffraktion. Här ges formlerna för spalt respektive cirkulär öppning enligt: $\sin \theta = \frac{\lambda}{a}$, där $a$ är spaltbredden, respektive $\sin \theta = 1.22 \frac{\lambda}{d_c}$, där $d_c$ är cirkelns diameter. Dessa formler gäller för vinklar till första minimum, dvs. när vågor slackar ut varandra. Föregående formler behandlar också bara en spalt respektive en cirkulär öppning. Formlerna säger att ju kortare våglängden är desto mindre blir den resulterande spridningsvinkeln.

### 3.4 Holografi

Med diffraktion i bakhuvudet presenteras nu holografi. Holografi kan definieras som en metod för att lagra och rekonstruera optiska vågfält. Skillnaden mot fotografi är den att vid...
holografi lagras både intensiteten och fasen hos ljusvågorna och inte bara dess intensitet som vid fotografi.\textsuperscript{15} Vi kan dela upp holografi i optisk respektive datorgenererad holografi. De två skiljer i konstruktionen av vågfältet. Inom optisk holografi förlitar vi oss på optik för att lagra vågfältet medan inom datorgenererad holografi så låter vi datom simulerar vågfältet.

3.4.1 Optisk holografi


![Figur 9. Skapning av interferensmönster](image)

Man rekonstruerar föremålet genom att belysa det resulterande interferensmönstret med samma referensstråle som användes vid konstruktionsprocessen. Interferensmönstret modulerar ljuset genom diffraktion (ljuset böjs och fokuseras) och resultatet blir en kopia av den våg som spreds från föremålet vid lagringen. 3D-bilden har samma djupegenskaper som det verkliga föremålet, både i horisontell och vertikal led.\textsuperscript{16}

3.4.2 Datorgenererad holografi

Med datorns hjälp kan man räkna fram olika interferensmönster, vilket är en mer praktisk metod för framtagning av föremål i 3D. På något sätt måste föremålet numeriskt beskrivas i 3D och ljuset måste numeriskt spridas från föremålet. Man imiterar optisk interferens genom att beräkna vågfronten från föremålet och addera den till vågfronten från referensstrålen. Interferensmönstret används sedan i en holografisk display, vilken modulerar en infallande


ljusstråle i x-, y- och z-led med hjälp av interferensmönstret. Detta imiterar rekonstrueringen av föremålet i den optiska holografen.

Att beräkna fram hologram på detta sätt med datorns hjälp är en mycket tidskrävande process. Det krävs detaljerad data av föremålet för att inte förlora upplösning och att på detta sätt simulera spridning av ljus och interferensens kräver mycket komplexa beräkningar. Tekniker för att minska beräkningstiden involverar bl.a. att ta bort den vertikala delen av vyn och bara möjliggöra att man kan förflytta sig horisontellt gentemot hologrammet. Man kan även minska storleken på föremålet och den vinkel i vilket man kan se föremålet. Därutöver kan man ignorera svarta regioner i föremålet då inget ljus sprids därifrån. 

Med teorin för diffraction och holografi genomgår så kan de apparater som används för att visa upp hologram beskrivas lite närmare.

### 3.5 SLM


Vidare studeras den DMD-teknik som används mer i detalj i nästa avsnitt.

### 3.6 Texas Instrument DLP

Grundstenen i Texas Instruments DLP är DMD:n, en optisk halvledare. Denna möjliggör digital modulering av ljus och återger hög upplösning i den projicerade bilden. DMD:n består av små aluminiumspeglar i mikroformat, i detta fall 16x16 mikrometer stora. Mikrospeglarna ligger i ett rutan i form av en rektangel, avståndet mellan varje mikrospigel är 1 mikrometer. De är direkt monterade på ett SRAM-chip (”Static Random Access Memory”).

---

Figur 10 visar en DMD till vänster och en uppförstoring av mikrospeglarna till höger. Som tidigare nämnts så belysas detta rutnät av mikrospeglar av en lampa (i en traditionell projektor) eller som i detta arbete av en laser.

Figur 10. DMD


3.6.1 Verklig och Virtuell bild


När vi använder en DMD som SLM så får vi en verklig och virtuell bild av det objekt som realiseras i hologrammet. Den verkliga bilden fångas upp av ett bildrekonstruktionsmedium, enligt figur 11 nedan.

---

Figur 11. Utrustning för att skapa verklig bild

Den virtuella bilden kan beskådas genom att ta bort den andra konvexa linsen och lägga till ett neutralt densitetsfilter enligt figur 12 nedan.

Figur 12. Utrustning för att skapa virtuell bild

I detta fall formas hologrammet på näthinnan i ögat. Det neutrala densitetsfiltret har som uppgift att minska intensiteten i laserljuset så att det inte blir skadligt för ögonen.


---

Om vi istället använder en lins som ger ifrån sig en konvergen våg så får vi precis som i föregående exempel fläckar utspridda på jämn a mellanrum. Men nu har de inte alls lika hög intensitet och är koncentrerade i mycket mindre fläckar. Dessutom ligger både den verkliga och virtuella bilden nu i fokus i skärmens plan. För varje "fläckkopia" rekonstrueras nu en verklig bild och en virtuellt bild.23

I detta avsnitt så presenterades hur DMD:n fasmodulerar ljuset, i nästa avsnitt visas på vilka sätt fasen hos en ljusväg kan ändras.

3.6.2 Fasändring av ljusväg


Hittills har det praktiska beskrivits i hur ljus böjs och interfererar och hur interferensmönster fås i den optiska holografien. Men detta arbete är främst inriktat på datorgenererad holografi och hur interferensmönster fås därigenom. De resterande avsnitten i denna teoretiska bakgrund beskriver teorin bakom, och härledningen av, de transformer som används för att generera dessa interferensmönster.

3.7 Huygen-Fresnels princip


Vi tänker oss en öppning med arean $V'$, denna är så liten att den interfererar infallande ljusstrålar. $V'$ ligger i $(x',y')$-planet och den belyses i positiv $z$-riktning, enligt figur 13.

Vi tittar på vågfältet som skapas i $(x,y)$-planet, vilket ligger parallellt med $(x',y')$-planet på avståndet $d$. Således går $z$-axeln genom origo på både $(x',y')$- och $(x,y)$-planet.

Huygen-Fresnels princip kan skrivas som:

**Formel 1. Huygen-Fresnels princip**

$$U(P_0) = \frac{1}{i\lambda} \int \int U(P_1) \exp[ikr_{01}] \cos \theta dV', \cos \theta = \frac{d}{r_{01}}, \quad r_{01} = \sqrt{d^2 + (x-x')^2 + (y-y')^2}$$

$U(P_0)$ och $U(P_1)$ beskriver vågfältet i punkten $P_0$ respektive $P_1$, $\lambda$ är det infallande ljusets våglängd och $\theta$ är vinkeln mellan normalen $z$ och vektorn $r_{01}$ som pekar från $P_0$ till $P_1$.

Mer generellt kan Huygen-Fresnels princip skrivas:

Formel 2. Huygen-Fresnels princip omskrivning

\[ U(x, y) = \frac{d}{i\lambda} \int \frac{U(x', y')}{r_{01}^2} \exp[ikr_{01}] \, dx' \, dy' \]

Förutsättningen för denna princip är att observationsavståndet (mellan \( P_0 \) och \( P_1 \)) är många våglängder från öppningen, alltså \( r_{01} \gg \lambda \).

3.8 Fresnelapproximationen

Fresnelapproximationen kan t.ex. användas för att förenkla Huygen-Fresnels princip. Denna approximerar avståndet \( r_{01} \) genom att göra en binomialutveckling av den. Binomialutvecklingen av en kvadratrot \( \sqrt{1+b} \), där \( b<1 \) är:

Formel 3. Binomialutveckling av kvadratrot

\[ \sqrt{1+b} = 1 + \frac{1}{2} b - \frac{1}{8} b^2 + \ldots \]

Vi förbereder \( r_{01} \) för binomialutvecklingen genom att faktorera ut \( d \):

Formel 4. Omskrivning av \( r_{01} \)

\[ r_{01} = d \sqrt{1 + \left( \frac{x-x'}{d} \right)^2 + \left( \frac{y-y'}{d} \right)^2} \]

Vi låter den andra och tredje termen i kvadratroten i formel 4 utgöra \( b \) i formel 3. Vidare så använder vi oss bara av den första (1) och andra (\( \frac{1}{2}b \)) termen i formel 3, vi får då approximationen:

---

25 Joseph W. Goodman, Introduction to Fourier Optics, s. 65-66
Nu när vi landat i en approximation av \( r_{01} \) så kan vi titta på hur denna kan förenkla Huygen-Fresnels princip. Vi kan se i formel 2 att Huygen-Fresnels princip har en \( r_{01}^2 \) term i nämnaren och en \( r_{01} \) term i exponenten. För termen i nämnaren så räcker det med att bara använda sig av den första termen i formel 5, dvs. \( r_{01}^2 \) kan approximeras till enbart \( d^2 \). Däremot för termen i exponenten så blir felet alldeles för stort om man bara använder sig av den första termen i formel 5. I exponenten så multiplikeras \( r_{01} \) med \( k \) vilket oftast är mycket stort, dessutom så kan fasändringar av en bråkdels radian förändra exponentens värde betydligt. Därför används både första och andra termen i binomialutvecklingen för att approximera \( r_{01} \) i exponenten. Huygen-Fresnels princip blir med dessa approximationer:

Formel 6. Fresnelapproximationen av Huygen-Fresnels princip

\[
U(x, y) = \frac{e^{ikd}}{i2d} \int \int U(x', y') \exp \left( i \frac{k}{2d} \left( x-x' \right)^2 + \left( y-y' \right)^2 \right) \, dx' \, dy'
\]

I formel 6 har \( d \) brutits ut från så väl exponenten som nämnaren.

### 3.9 Helmholtzekvationen

Innan transformerna beskrivs presenteras helmholtzekvationen. Denna beskriver hur ljus från ett objekt sprids i rummet. Den används för att ”simulera” reflekterat ljus från det föremål vars interferensmönster räknas fram med transformen. Den inhomogena Helmholtzekvationen ges av följande ekvation:

Formel 7. Helmholtzekvationen

\[
(\Delta + k^2) \phi(x, y) = f(x', y', z')
\]

\[
\Delta \phi(x, y) = \left( \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \right)
\]

\( \phi(x, y) \) beskriver i vårt fall vågfältet i hologramplanet, närmare bestämt amplituden på intensiteten hos en punkt i hologramplanet. \( f(x', y', z') \) beskriver i vårt fall vågfältet från objektet (aktiv källa), amplituden på intensiteten hos en punkt i volymen \( V' \). Helmholtzekvationen beskriver alltså hur t.ex. ljusvågor (i varje punkt) från ett objekt sprids i rummet. I den homogena Helmholtzekvationen så är \( f(x', y', z') = 0 \).

### 3.10 Transform för datorgenererade hologram

1968 presenterades en metod för att skapa datorgenererade hologram av tredimensionella objekt. Denna metod genererade ett Fresneltransformhologram bestående av både genomskinliga som ogenomskinliga delar. Man beskriver objektet utifrån en samling punktkällor. Amplituddistributionen av intensiteten för Fresneltransformhologrammet beskrivs av följande formel:

---

26 Joseph W. Goodman, Introduction to Fourier Optics, s. 66-67
Formel 8. Fresneltransform

\[ U(x, y) = \left( \frac{ik}{z'2\pi} \right) U(x', y', z') \exp \left[ i k \frac{(x-x')^2 + (y-y')^2}{2z'} \right] \]

\( x' \) och \( y' \) beskriver koordinaterna för en punktkälla på avståndet \( z' \) från hologramplanet.


En ny metod skapades som bygger på Fourierholografin utökad till att gälla även tredimensionella objekt. Figur 14 nedan visar konstruktionsprocessen för denna metod.

Figur 14. Konstruktionsprocess för datorgenererad holografi

Figuren visar från vänster objektet i form av två bilder (samma bilder som i figur 1), den konvexa linsen \( L \) och hologramplanet \( H \) (med koordinater \( x \) och \( y \)). Objektet som definieras av volymen \( V' \) (med koordinater \( x', y' \) och \( z' \)) ligger i fokalplanet för linsen, på avståndet \( f \) från linsen.

Vi tänker oss att från varje punktkälla i objektet så utbreder sig sfäriska vågor (i form av det ljus som sprids från varje punktkälla), deras amplituddistribution i hologramplanet beskrivs av \( U_s(x,y) \) i nedan formel:

Formel 9. Transform för datorgenererade hologram 1

\[ U_s(x, y) = \left( \frac{ik}{f'2\pi} \right) U(x', y', z') \exp \left[ -\frac{ik}{2} \left( \frac{z'^2}{f'} (x^2 + y^2) + \frac{2}{f} (xx' + yy') \right) \right] dV' \]

Integrationen sker över volymen \( V' \) (objektet) där \( U(x', y', z') \) är amplituddistributionen i \( V' \). \( k \) är ”vågnumret” för referensstrålen som används. Man kan se Fouriertransformen ovan som ett sätt att transformera tredimensionella objekt till ett tvådimensionellt plan (hologramplanet \( H \)).

Rekonstruktionsprocessen är som konstruktionsprocessen fast den omvänta kan man säga. Man belyser hologramplanet \( H \) med samma referensstråle som användes vid beräkningen i konstruktionsprocessen. Ljuset går vidare genom linsen \( L \) och hologrammet formas på

---

28 James P. Waters, ”Three-Dimensional Fourier-Transform Method for Synthesizing Binary Holograms”, Journal of the Optical Society of America, Vo. 58, No. 9, (1968) s. 1284
avståndet \( f \), i fokalplanet för linsen. I figur 15 nedan så indikerar \( P \) där linsens fokalpunkt ligger.

![Diagram](image)

**Figur 15. Rekonstruktionsprocess för datorgenerade hologram**

Vid rekonstruktionen så fås två uppsättningar av de två bilderna som formade \( V' \) i konstruktionsprocessen. Den högra utgör den ”korrekt” \( V' \) (objektet vi utgick ifrån i konstruktionsprocessen) medan den vänstra är \( V' \) speglad i horisontell och vertikal led.  

3.10.1 Härledning av Fresneltransformaten

I detta avsnitt härleds formel 8 utifrån Huygen-Fresnels princip. 

Som tidigare nämnts så beskriver formel 8 amplituddistributionen för en punktkälla. Vi tänker oss på samma sätt att Huygen-Fresnels princip i formel 2 beskriver amplituddistributionen för en punktkälla. Detta betyder för det första att vi tar bort integraltecknet i formel 2, precis som formel 8 saknar integraltecken. För det andra så ser vi nu den lilla öppningen \( V' \), som antogs i Huygen-Fresnel principen, som ett objekt i stil med de två bilderna i figur 14. Detta är möjligt då Huygens princip gäller för både den lilla öppningen och för bilderna. Enligt Huygens princip så kan varje punkt på en våg ses som ett eget vågsystem från vilket strålar går ut i alla riktningar, detta beskrevs tidigare. Varje punktkälla i bilderna (figur 14) kan ses som ett eget vågsystem, precis som varje punkt i öppningen (figur 13) kan ses som ett eget vågsystem. Då vi i Huygen-Fresnels princip inför en volym \( V' \) bestående av bilderna i figur 14 så blir avståndet \( d \) vi använt istället en variabel \( z' \), då det inte är samma avstånd från endera av bilderna i figur 14 till hologramplanet. Huygen-Fresnels princip med alla ovan förändringar blir:

Formel 10. Huygen-Fresnels princip för en punktkälla

\[
U(x, y) = \frac{z'}{i\lambda} \frac{U(x', y', z')}{r_{0i}} \exp[ikr_{0i}], \quad r_{0i} = \sqrt{z'^2 + (x-x')^2 + (y-y')^2}.
\]

Vi inför nu Fresnelapproximation på formel 10. Termen i nämnaren förkortar vi till bara första termen i approximationen och termen i exponenten förkortar vi till första och andra termen i approximationen. Vi får då följande formel:

Formel 11. Huygen-Fresnels princip för en punktkälla med Fresnelapproximation 1

\[
U(x, y) = \frac{1}{i\lambda} \frac{U(x', y', z')}{z'} \exp\left[ik\left(z' + \frac{(x-x')^2 + (y-y')^2}{2z'}\right)\right], \quad k = \frac{2\pi}{\lambda} \Rightarrow \frac{1}{\lambda} = \frac{k}{2\pi}.
\]

---

Bryter vi ut $z'$ från exponenten så får vi:

Formel 12. Huygen-Fresnels princip för en punktkälla med Fresnelapproximation 2

$$U(x, y) = -\exp[ikz'] \frac{ik}{z'2\pi} U(x', y', z') \exp\left[ -\frac{ik}{2z'} \left( (x-x')^2 + (y-y')^2 \right) \right], \quad -\exp[ikz'] = \exp\left[ \frac{1}{ikz'} \right] \approx 1$$

3.10.2 Transformen

Vi tittar återigen på figur 2 som beskriver den utrustning som används för att skapa hologram i detta arbete. Hologrammet skapas från en koherent laserstråle som i viss mån reflekteras av DLP:n och formar utifrån ett interferensmönster ett hologram i den konvexa lensens bakre plan.

Transformen för att beräkna fram hologram med datorns hjälp som används i detta arbete är följande:

Formel 13. Transform för datorgenererade hologram 2

$$U_s(x, y) = \frac{1}{V'} \int \int U(x', y', z') \exp\left[ -\frac{ik}{2f} \left( \sqrt{z'^2 + (x-x')^2 + (y-y')^2} - \frac{x^2 + y^2}{2f} \right) \right] dV'$$

$U_s(x, y)$ beskriver amplituden på intensiteten i en punkt i hologramplanet. $V'$ är den volym som hologrammet utgör, bestående av bild1 och bild2 i figur 1. $U(x', y', z')$ beskriver amplituden på intensiteten i en punkt i volymen $V'$. $z'$-axeln utgör normal till centrum i hologramplanet och går genom centrum av den rekonstruerade 3D-bilden. $k$ är "vågnumret" av laserljuset och $f$ är den konvexa lensens fokallängd.

Transformen innefattar fysiken hos de monokromatiska ljusvågorna som utstrålar från varje punkt på objektet, deras passage genom det optiska systemet och när de genomgår superposition i varje punkt i hologramplanet. Transformen representerar integrering över objektet med sfäriska våglösningar av vågekvationen på Helmholtz form. Det tillkommer även ändring av fasen pga. den sfäriskt konvexa lensen i ljusets väg. Då den konvexa lensen har varierande djup beroende på var strålen träffar så passera vissa delar av strålen i djupare delar än andra, detta tillför en fasskillnad dem emellan. Detta är den fördröjning av ljusvägen som sker då den passera genom ett icke ledande medium, som beskrevs i avsnitt 3.6.2 Fasåndering av ljusväg. Den andra termen i exponenten beskriver denna fasändring. Den första termen i exponenten beskriver fasen i förhållande till djupet genom avståndet från en punkt på objektet till en punkt på hologramplanet. 30

3.10.3 Härelädning av transformen

Precis som vid härledning av Fresneltransformen så utgår vi även i detta fall ifrån Huygen-Fresnels princip. Vi använder oss bara av första termen i approximationen i nämnaren, men behåller $r_0$ som det är i exponenten i Huygen-Fresnels princip, vi får då följande formel:

Formel 14. Huygen-Fresnels princip med Fresnelapproximation 1

$$U(x, y) = \frac{1}{i\lambda} \int \int \frac{U(x', y')}{d} \exp[ikr_{01}] \, dx' \, dy'$$

Nu låter vi $V'$ definieras av bilderna i figur 1, $z'$ införs och för att öka precisionen så återinför vi $r_{01}$ i nämnaren. Då vi använder oss av en konvex lins, för att skapa hologram med hjälp av transformen i föregående avsnitt, så införs ett minus framför termen i exponenten.\(^{31}\)
Vi får nu följande formel:

**Formel 15. Huygen-Fresnels princip med Fresnelapproximation 2**

$$U(x, y) = \frac{1}{i\lambda} \frac{U(x', y', z')}{r_{01}} \exp[-ikr_{01}] \, dV', \quad r_{01} = \sqrt{z'^2 + (x-x')^2 + (y-y')^2}.$$ 

För att beskriva fasändringen pga. den konvexa linsen i ljusets väg så införs följande uttryck:

**Formel 16. Fasändringen pga. den konvexa linsen**

$$t(x, y) \approx h_0 \exp \left[ ik \frac{x^2 + y^2}{2f} \right], \quad h_0 = \exp[-ikd_0], \text{ där } d_0 \text{ är linsens tjocklek.}\(^{32}\)$$

Inför vi detta uttryck i formel 15 så får vi följande formel:

**Formel 17. Huygen-Fresnels princip med Fresnelapproximation 3**

$$U(x, y) = \frac{h_0}{i\lambda} \frac{U(x', y', z')}{r_{01}} \exp \left[ -ik \left( r_{01} - \frac{x^2 + y^2}{2f} \right) \right] \, dV'.$$

Skillnaden mellan denna formel och transformen i formel 13 är den första termen $h_0 / i\lambda$. Om vi tittar på de som ligger bakom transformen i formel 13 så kan det förklaras av att transformen är framräknad utifrån den fysiska utrustning som används i figur 2. De beskrivar med transformen vad de vill skall visas och inte visas. Transformen får de genom att använda sig av metoden "ray tracing"\(^{33}\) på svenska. "Strålspårning" är en metod för att modulera ljusets väg genom att följa ljustrålar då de interagerar med optiska ytor, främst linser. Detta förklarar den första termen i formel 17. Med den fysiska utrustningen så kan hologrammet bara projiceras i fria rummen så länge vi inte har något medium för bildrekonstruktion. Detta medium behövs för att hologrammet ska bli synligt. Detta betyder att punktkällorna i det objekt som beskrivs av hologrammet inte är aktiva, då de behöver ett medium för att synas. Därför är vågfältet som strålar från lasern lösningar till den homogena Helmholtzekvationen.

\(^{31}\) Joseph W. Goodman, Introduction to Fourier Optics, s. 68

\(^{32}\) Bahaa EA Saleh och Malvin Carl Teich, Fundamentals of Photonics, s. 55-60

4 Genomförande

I detta avsnitt beskrivs tillvägagångssätt. Då den praktiska delen avskaffades så kommer här bara programmeringsdelen att beskrivas.

4.1 Programmeringsdel


I den diskreta domänen så är integrering inget annat än en summering över det intervall integreringen sker. I detta fall över volymen $V'$, vilket innebär att ett en punkt i hologramplanet $U_s(x,y)$ fås genom att summera alla bidrag från volymen $V'$. Detta ger att för en punkt $(x,y)$ så måste transformen ovan beräknas lika många gånger som det finns kombinationer av $x'$, $y'$ och $z'$. Som ett exempel kan vi tänka oss att volymen $V'$ har upplösningen 50x60x10 detta ger 30000 möjliga kombinationer av $x'$, $y'$ och $z'$. Om vi sedan antar att DMD:n har en upplösning på 1024x768 så ger det att hela $U_s$ matrisen fås genom att göra 23,59296 miljarder (1024x768x30000) beräkningar av ovan transform. Man ser tydligt att det krävs enorma beräkningar för att få ett enda hologram. För att sedan få rörliga hologram så måste det i princip göras 23,5 miljarder beräkningar 24 gånger per sekund. Sedan bör det även tillåggas att i detta räkneexempel användes bara 10st bilder av storlek 50x60, vilket ger ett mycket litet hologram. Vidare beskrivs programkoden för det program som beräknar transformen och genererar interferogram utifrån den.

4.2 Matlabkoden

I detta avsnitt beskrivs den programkod som skrivits i Matlab. Den är här indelad i mindre avsnitt som beskriver delar av programmet för lättare förståelse.

4.2.1 Inläsning och behandling av bilderna

Matlab har redan en trevlig funktion `imread` för att läsa in bilder från en fil och returnera bildinformationen i en matris. Som den används här returnerar `imread` gråskalevärden för varje bildpunkt i bilden, dvs. en matris av samma dimension som bilden fås. Två bilder läses in, nedan är koden för detta:

```matlab
image1Array = double(imread('bild1.png'));
image2Array = double(imread('bild2.png'));
```

Det man får tänka på är att bilder läses in som HxB (höjd x bredd). Efter inläsningen av bilderna så normaliseras gråskalevärdena mellan 0 och 1. För att få bättre precision på de normaliserade gråskalevärdena används funktionen `double` vid inläsningen av bilderna. Normaliseringen kan ses nedan:

```matlab
image1ArrayNorm = image1Array/max(max(image1Array));
image2ArrayNorm = image2Array/max(max(image2Array));
```

Då de två bilderna ska skapa en volym $V'$ så måste det kontrolleras att storleken på matriserna för de två bilderna är lika stora. Detta görs med nedan `if`-sats:

```matlab
if (image1Size(1,1)*image1Size(1,2))>(image2Size(1,1)*image2Size(1,2))
image2A = zeros(image1Size(1,1),image1Size(1,2));
image2A(1:(image2Size(1,1)),1:(image2Size(1,2)))=image2ArrayNorm;
```
Det är här förutsatt att bild1.gif alltid har större (eller lika stor) dimension som bild2.gif.

4.2.2 Skapande av \( U(x',y',z') \), \( U_s(x,y) \), samt konstanter

Nu kan \( U(x',y',z') \) skapas, den innehåller bildinformationen från bilderna i \( V' \). Då bilderna ligger som HxB så skapas \( U \) som HxBxD (höjd x bredd x djup), dvs. \( U(y',x',z') \).

\[
\begin{align*}
U &= \text{zeros}(\text{image1Size}(1,1),\text{image1Size}(1,2),2); \\
U(:,:,1) &= \text{image1ArrayNorm}; \\
U(:,:,2) &= \text{image2ArrayNorm};
\end{align*}
\]

Som visas ovan så har volymen \( V' \) två ”lager” med en bild i vardera ”lager”, \( z' \) kan alltså bara få värdena 1 eller 2.

Ljusets vågnummer \( k \), den konvexa lensens fokallängd \( f \), DMD:n, \( U \), där hologrammet kommer sparas samt dimensionen på \( V' \) definieras med följande kod:

\[
\begin{align*}
\text{wavelength} &= 633e^{-9}; \\
k &= (2*\pi)/\text{wavelength}; \\
f &= 0.4; \\
\text{xMDSize} &= 768; \\
\text{yMDSize} &= 1024; \\
\text{Us} &= \text{zeros}(\text{xMDSize},\text{yMDSize}); \\
[ySize \timesSize zSize] &= \text{size}(U); \\
VSize &= \timesSize*ySize*zSize;
\end{align*}
\]

4.2.3 Tidigare transformberäkning

Nu kommer vi till den tjänere delen där huvuddelen av arbetet kommer ske. I tidigare versioner av programmet så var det en for-loop för \( x, y, x', y' \) och \( z' \), alltså fem stycken nästlade for-loopar. Detta visas nedan:

\[
\begin{align*}
\text{Hpoint} &= \text{zeros}(1,\text{VSize}); \\
&\text{for } x=1:\text{xDMDsize} \\
&\quad \text{xD} = 16e^{-6}*x; \\
&\text{for } y=1:\text{yMDSize} \\
&\quad \text{yD} = 16e^{-6}*y; \\
&\quad \text{temp}=1; \\
&\text{for } xprim=1:\timesSize \\
&\quad \text{xV} = 51e^{-6}*xprim; \\
&\text{for } yprim=1:\timesSize \\
&\quad \text{yV} = 51e^{-6}*yprim; \\
&\text{for } zprim=1:zSize \\
&\quad \text{if } zprim == 1 \\
&\quad \text{zV} = f+51e^{-6}*zprim; \\
&\quad \text{else} \\
&\quad \text{zV} = (f+zprim*0.025)+51e^{-6}*zprim; \\
&\quad \text{end} \\
&\quad \text{xyzSqrt} = \sqrt{(zV^2+(\text{xD}-xV)^2+(\text{yD}-yV)^2)}; \\
&\quad \text{Hpoint}(\text{temp}) = (\text{Us}(yprim,xprim,zprim)/\text{xyzSqrt})*
\end{align*}
\]

\[
\begin{align*}
&\quad \text{exp}(-i*k*(\text{xyzSqrt}-(\text{xD}^2+\text{yD}^2)/(2*f)))); \\
&\quad \text{temp}=\text{temp}+1;
\end{align*}
\]

28
Us(x,y) = sum(Hpoint);
end

4.2.4 Optimering 1
Matlab fungerar dåligt med for-loopar, det finns ett sätt att undvika dessa. Matriser går att använda till mycket i Matlab och matrisoperationer kan ersätta for-loopar om man ordnar matriserna smart. Dessutom tar matrisoperationer mycket kortare tid än for-loopar. Därför skapades tre vektorer; en för x’- y’ och z’. Koden för detta visas nedan:

```
zprimArray=floor(1:(1/(xSize*ySize)):(zSize+1));
zprimArray=zprimArray(1:VSize);
xprimArray=floor(1:(1/ySize):(xSize+1));
xprimArray=xprimArray(1:(VSize/zSize));
xprimArray=repmat(xprimArray,1,(VSize/(xSize*ySize)));
yprimArray=1:1:ySize;
yprimArray=repmat(yprimArray,1,(VSize/ySize));
```

Matlabfunktionen floor avrundar till närmaste heltal neråt, t.ex. floor(1.2)=1 och floor(2.1)=2. Som exempel kan tas om xSize=4, ySize=3, zSize=2 (VSize=24)så är:

```
zprimArray=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2];
xprimArray=[1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 4 4 4 4 4 4 4 4];
yprimArray=[1 1 1 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3];
```

På så sätt fås alla kombinationer av x’, y’ och z’. I ovan exempel hålls z’-värdena konstanta under alla kombinationer av x’ och y’ (xSize*ySize), först z’=1 och sedan z’=2. På liknande sätt vill hålls x’-värdena konstanta under alla kombinationer av y’ (ySize), först x’=1, x’=2, x’=3 och sist x’=4. Detta ger att y’-värdena ”loopas” från 1 till 3.

För att få detta beteende i kodan så skapas zprimArray genom att först skapa en serie tal mellan 1 till zSize+1. För att få rätt antal värden i serien så ökar den med 1/(xSize*ySize) varje iteration. Detta ger i ovan exempel en serie som har värden <2 mellan 1 och VSize/2 (24/2=12) samt värden ≥2 mellan VSize/2+1 (13) till VSize+1 (25). När man sedan köt floor på denna serie så fås de rätta heltalsvärdena. Tills ”klipps” sista värdet från zprimArray av för att få rätt dimension på matrisen.

Talserien som ligger till grund för xprimArray går från 1 till xSize+1 med en ökning av 1/ySize varje iteration. Efter att floor har körts och det sista värdet ”klippits” bort så återstår en serie innehållande VSize/2 värden, därför måste denna repeteras en gång till vilket görs med funktionen repmat.

Talserien för yprimArray är den enklaste då den går från 1 till ySize med en ökning av 1 varje iteration, sedan används repmat för att få denna serie rätt antal gånger. För att koppla x’-, y’- och z’-värdena till verkligheten så måste de tas fram i meter, vilket görs nedan:

```
xprimArray=51e-6.*xprimArray;
yprimArray=51e-6.*yprimArray;
zprimArray=f+51e-6.*zprimArray(1:VSize);
zprimArray(((VSize/2)+1):VSize)=0.05+zprimArray(((VSize/2)+1):VSize);
```

Med detta gjort så måste U som skapades tidigare omordnas för att passa med zprimArray, xprimArray och yprimArray. Detta görs med den inbyggda matlabfunktionen reshape.
4.2.5 Optimerad transformberäkning 1

Nu kan beräkningen av transformen börja. Med de nya matriserna för $x'$, $y'$ och $z'$ så har vi uteslutet tre av de tidigare fem nästlade for-looparna. Resultatet visas nedan:

```matlab
Ureshaped = reshape(U, 1, VSize);

xyzSqrt = zeros(1, VSize);
for x = 1:xDMDsize
    xD = 16e-6 * x;
    for y = 1:yDMDsize
        yD = 16e-6 * y;
        xyzSqrt = sqrt(zprimArray.^2 + (xD-xprimArray).^2 +
                      (yD-yprimArray).^2);
        Us(x, y) = sum((Ureshaped./xyzSqrt).*exp(-i*k.*
                        (xyzSqrt-((xD^2+yD^2)/(2*f)))));
    end
    progress = 100*(x/xDMDsize)
end
```

Först skapas en matris $xyzSqrt$ för mellanlagring av resultatet från $\sqrt{x'^2+(x-x')^2+(y-y')^2}$ i transformen. Den yttre for-loopen loopar igenom $x$-värdena medan den inre loopar igenom $y$-värdena. För att få $x$ och $y$ i meter så beräknas $xD$ resp. $yD$. $Us$ är direkt översatt från transformen, det är här omkring 90 % av tiden för beräkningen spendedes. Variabeln progress är enbart för att kunna få feedback hur långt beräkningen kommit.

4.2.6 Skapande av transformbild

Med $Us$ beräknad så kan transformbilden skapas. $Us$ innehåller komplexa tal, för att skapa interferogrammet så räcker det att ”plocka ut” den imaginära delen av $Us$ och sedan normalisera värdena för den. Därefter används matlabfunktionen imwrite för att från värdena i den normaliserade matrisen skapa en bild. Förloppet visas nedan:

```matlab
UsImag = imag(Us);
UsImagNorm = UsImag/max(max(UsImag));
imwrite(UsImagNorm, 'trans.bmp');
```

Koden för den första optimeringen kan i sin helhet beskådas i appendix 1.

4.2.7 Optimering 2

I ett försök att undvika ännu fler for-loopar så skapades en version med enbart en for-loop. För att uppnå detta så måste även $x$- och $y$-värdena placeras i matriser inför transformberäkningen. I detta avsnitt presenteras bara delen av koden som skiljer mot tidigare.

Detta tankesätt med $x$ och $y$ i matriser ger att delar av DMD:n, inte som tidigare enskilda värden, kommer att beräknas under ett varv i for-loopen. Först måste det bestämmas hur stora delar DMD:n ska delas upp i:

```matlab
xDMDsize = 768;
yDMDsize = 1024;
DMDsize = xDMDsize*yDMDsize;
divFactor = 16;
DMDslices = divFactor^2;
DMDpsize = DMDsize/DMDslices;
xDMDpsize = (xDMDsize/divFactor)*(yDMDsize/divFactor)/yDMDsize;
yDMDpsize = yDMDsize;
```
Om vi först tittar på sista raden i koden ovan så ser vi att DMD:n delas i y-led upp i hela "y-strängar" (en y-sträng fås om vi följer y-värdena längs ett konstant x). Efter att definierat storleken på DMD:n så bestäms en divFactor utifrån vilken DMDslices sätts. DMDslices anger i hur många delar DMD:n delas upp i. I figur 16 nedan så demonstreras "y-strängar" i form av de streckade linjerna.

**Figur 16. Uppdelning av DMD:n**

DMDpsize anger storleken av de lika stora delarna. xDMDsize anger hur många x-värden som finns i varje del. Som exempel blir de olika värdena i ovan kod: DMDslices=256, DMDpsize=3072, xDMDpsize=3 och yDMDpsize=1024. Detta talar om att DMD:n delas upp i 256st lika stora delar vilka vardera innehåller 3072 värden, som fås från 3st "y-strängar" (i vilka x är t.ex. 1, 2 och 3). Man kan också se det som att xDMDsize/DMDslices=768/256=3 och 3st yDMDpsize ger oss 3x1024=3072 värden.

Framräkningen av DMD-uppdelningen ger oss grunden för att beräkna de matriser som ska användas i transformberäkningen.

Grunden i denna optimering är skapandet av matriser för x och y. Detta görs med följande kod:

```matlab
xArray=floor(1:(1/(yDMDpsize*VSize)):(xDMDpsize+1));
xArray=xArray(1:DMDpsize*VSize);
xArrayP=(reshape(xArray',VSize,DMDpsize))';
xArray=16e-6.*xArrayP;

yArray=1:1:yDMDsize;
yArray=16e-6.*yArray;
yArray=repmat(yArray',xDMDpsize,VSize);
```

Tekniken för att göra detta kan liknas vid hur zprimArray, xprimArray och yprimArray togs fram tidigare. För att förtydliga så kan som exempel tas xDMDsize=2, yDMDsize=3 och vi tänker oss att xDMDpsize=1, yDMDpsize=3. Vidare med xSize=4, ySize=3, zSize=2 (VSize=24) som användes i tidigare beräkningar så får vi:
Formel 18. xArray och yArray

\[
\begin{bmatrix}
1 & 1 & 1 & \ldots & 1 & 1 & 1 \\
1 & 1 & 1 & \ldots & 1 & 1 & 1 \\
1 & 1 & 1 & \ldots & 1 & 1 & 1
\end{bmatrix} \begin{bmatrix}
2 & 2 & 2 & \ldots & 2 & 2 & 2 \\
2 & 2 & 2 & \ldots & 2 & 2 & 2 \\
2 & 2 & 2 & \ldots & 2 & 2 & 2
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 1 & \ldots & 1 & 1 & 1 \\
2 & 2 & 2 & \ldots & 2 & 2 & 2 \\
3 & 3 & 3 & \ldots & 3 & 3 & 3
\end{bmatrix}
\]

I ovan matriser anges inte de riktiga värdena för enklare förståelse. xArray är en matris av storlek 3x24 fylld med 1:or första gången for-loopen körs och fylld med 2:or andra gången. yArray är av samma storlek som xArray och innehåller på första raden bara 1:or, på andra raden bara 2:or och på tredje raden bara 3:or. Att matriserna innehåller 24 kolumner kommer från VSize, eftersom V’ innehåller 24 värden. Matriserna har 3 rader eftersom yDMdpsize=3 i vårt exempel.

zprimArray, xprimArray och yprimArray fäst precis som innan med skillnaden att repmat används för att få rätt storlek på matriserna:

\[
\begin{align*}
zprimArray &= repmat(zprimArray,DMDpsize,1); \\
xprimArray &= repmat(xprimArray,DMDpsize,VSize/(xSize*ySize)); \\
yprimArray &= repmat(yprimArray,DMDpsize,VSize/ySize));
\end{align*}
\]

Detta ger i ovan exempel:

Formel 19. zprimArray, xprimArray och yprimArray

\[
\begin{bmatrix}
1 & 1 & 1 & \ldots & 1 & 1 & 1 & 2 & 2 & 2 & \ldots & 2 & 2 & 2 \\
1 & 1 & 1 & \ldots & 1 & 1 & 1 & 2 & 2 & 2 & \ldots & 2 & 2 & 2 \\
1 & 1 & 1 & \ldots & 1 & 1 & 1 & 2 & 2 & 2 & \ldots & 1 & 1 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 1 & \ldots & 4 & 4 & 4 & 1 & 1 & 1 & \ldots & 4 & 4 & 4 \\
1 & 1 & 1 & \ldots & 4 & 4 & 4 & 1 & 1 & 1 & \ldots & 4 & 4 & 4 \\
1 & 1 & 1 & \ldots & 4 & 4 & 4 & 1 & 1 & 1 & \ldots & 4 & 4 & 4
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 2 & 3 & \ldots & 1 & 2 & 3 & 1 & 2 & 3 & \ldots & 1 & 2 & 3 \\
1 & 2 & 3 & \ldots & 1 & 2 & 3 & 1 & 2 & 3 & \ldots & 1 & 2 & 3 \\
1 & 2 & 3 & \ldots & 1 & 2 & 3 & 1 & 2 & 3 & \ldots & 1 & 2 & 3
\end{bmatrix}
\]

Naturligtvis är dessa matriser också av storleken 3x24.

Efter att omordnat U för att passa med zprimArray, xprimArray och yprimArray som gjordes tidigare så måste även den anpassas med repmat:

\[
\text{Ureshaped} = repmat(\text{Ureshaped},\text{DMDpsize},1);
\]

4.2.8 Optimerad transformberäkning 2
Nu återstår bara transformberäkningen:

\[
\begin{align*}
\text{for n}=1:1:\text{DMD}\text{slices} \\
\text{xyzSqrt} &= \sqrt{(\text{zprimArray.}^2+(\text{xArray-xprimArray}.)^2+(\text{yArray-yprimArray}.)^2));} \\
\text{xStart} &= \text{xDMDpsize}*(\text{n-1})+1; \\
\text{xEnd} &= \text{xStart} + \text{xDMDpsize}-1; \\
\text{UsTemp} &= \text{sum}((\text{Ureshaped.}./\text{xyzSqrt}).*\exp(-i*k.*}
\end{align*}
\]

32
Man kan här se att det nu bara återstår en for-loop och att den bara behöver köras lika många ganger som DMD:n tidigare delades upp i.

Det som tillförs i denna transformberäkning är indexeringsvärdena xStart och xEnd för att placera de framräknade värdena rätt i Us. xArray måste också uppdateras för varje loop genom att öka xArray med xDMDsize så fås nästa serie av x-värden.

Koden för den andra optimeringen kan i sin helhet beskådas i appendix 2.

4.2.9 Optimering 3

Som beskrevs i avsnitt 3.4.2 Datorgenererad holografi så kan man ignorera svarta regioner i de bilder som ska avbildas som hologram. Detta ligger som grund för vad som görs i denna optimering.

Koden i optimering 1 används som utgångspunkt, anledningen till detta framgår av avsnitt 5.5 Beräkningstid för optimering 2.

Efter att bilderna lästs in och förts in i Us så skapas en matris Uunion vars syfte är att ta reda på bildernas gemensamma svarta områden. Detta görs enligt följande:

\[
\text{Uunion} = U(:,:,1) + U(:,:,2);
\]
\[
\text{UunionNorm} = \text{Uunion} / \text{max(max(Uunion))};
\]
\[
\text{UunionNormCeil} = \text{ceil(UunionNorm)};
\]

Det är ganska enkelt men effektivt. bild1 med bild2 summers helt enkelt, eftersom värdet "0" betyder svart så kommer Uunion att ha värdet "0" där de gemensamma svarta regionerna finns. Därefter normaliseras Uunion för att få värdena mellan noll och ett. Funktionen ceil används för att höja alla värden större än noll och mindre än ett. UunionNormCeil innehåller därför enbart värdena noll och ett. Anledningen till detta förklaras längre fram.

Efter att konstanter, DMD:n z', x' - och y'-array skapas enligt tidigare så skapas en matris UunionArray:

\[
\text{UunionArray} = \text{reshape(UunionNormCeil,1,\text{VSize}/\text{zprimSize})};
\]
\[
\text{UunionArray} = \text{repmat(UunionArray,1,\text{zprimSize})};
\]

Detta genom att utgå från UunionNormCeil och ordna om den så den får samma format som z', x' - och y'-array. Då UunionNormCeil fortfarande är summan av de två bilderna så utgör det bara i realiteten en bild, eller ett av de två "lagren" i V'. Genom att använda repmat så dubbleras UunionNormCeil så den får rätt storlek.

Nu kommer vi till den del där anledningen till att vi skapade UunionNormCeil förklaras. UunionArray (dubblerad UunionNormCeil) har som sagt värdet "0" för de gemensamma svarta regionerna, de som vi vill ignorera i beräkningen. Genom att multiplicera UunionArray med z', x' och y'-array så nollställs de z', x' och y'-värden som ska ignoreras:

\[
\text{zprimArray} = \text{zprimArray}.*\text{UunionArray};
\]
\[
\text{xprimArray} = \text{xprimArray}.*\text{UunionArray};
\]
\[
\text{yprimArray} = \text{yprimArray}.*\text{UunionArray};
\]

Tankesättet framgår tydligare vid transformberäkningen.

Innan transformberäkningen så skapas följande matriser:

\[
\text{xDzero} = \text{zeros(1,\text{VSize})+UunionArray};
\]
yDzero = zeros(1,VSize)+UunionArray;
xyzSqrtNonZero = ones(1,VSize);

Matriserna xDzero och yDzero kommer i transformberäkningen att användas för att se till att även x- och y-värdena är noll på ”rätt” ställen. Matrisen xyzSqrtNonZero används för att undvika division med noll i transformberäkningen.

4.2.10 Optimerad transformberäkning 3

Nu återstår bara transformberäkningen:

for x=1:xDMDsize
 xD = xDzero.*(16e-6*x);
for y=1:yDMDsize
 yD = yDzero.*(16e-6*y);
xyzSqrt = sqrt(zprimArray.^2+(xD-xprimArray).^2+(yD-yprimArray).^2);
xyzSqrtZeroIndex = find(xyzSqrt == 0);
xyzSqrtNonZero = xyzSqrt;
xyzSqrtNonZero(1,xyzSqrtZeroIndex) = 1;
UsPart = (Ureshaped./xyzSqrtNonZero);
Us(x,y)=sum(UsPart.*exp(-i*k.*(xyzSqrt-((xD.^2+yD.^2)./(2*f)))));
end
progress = 100*(x/xDMDsize)
end


Koden för den tredje optimeringen kan i sin helhet beskådas i appendix 3.

4.2.11 Optimering 4

I optimering 3 så tas bara hänsyn till de gemensamma svarta regionerna. Men eftersom de två bilderna inte ser likadana ut så finns det svarta regioner som inte är gemensamma mellan dem.

Skillnaden mot optimering 3 är på vilket sätt matrisen tas fram för de gemensamma och, i detta fallet också, de icke gemensamma svarta regionerna:

Uzprim1 = U(:,:,1);
Uzprim2 = U(:,:,2);
Uzprim1Norm = Uzprim1/max(max(Uzprim1));
Uzprim2Norm = Uzprim2/max(max(Uzprim2));
Uzprim1NormCeil = ceil(Uzprim1Norm);
Uzprim2NormCeil = ceil(Uzprim2Norm);
Två matriser skapas, $Uzprim1$ och $Uzprim2$, i dessa sparas bild1 respektive bild2. De båda
matriserna normaliserar separat och $ceil$ används för att få endast ettor och nollor, precis som
vid optimering 3. Sedan förenas två matriserna enligt:

```matlab
UunionArray = zeros(1,VSize);
U1temp = reshape(Uzprim1NormCell,1,VSize/zprimSize);
U2temp = reshape(Uzprim2NormCell,1,VSize/zprimSize);
UunionArray(1,1:VSize/zprimSize) = U1temp;
UunionArray(1,((VSize/zprimSize)+1):VSize) = U2temp;
```

De båda matriserna ordnas om för att få rätt format. Därefter så sammanfogas dem, här är
viktigt att matrisen för bild1 hamnar före matrisen för bild2. Resultatet är en matris
$UunionArray$ precis som vid optimering 3 med skillnaden att denna innehåller ytterligare
nollvärden från de icke gemensamma svart regionerna.

Transformberäkningen görs precis som i optimering 3.
Koden för den fjärde optimeringen kan i sin helhet beskådas i appendix 4.
5 Resultat

I detta avsnitt visas de uppnådda resultaten. Då den praktiska delen inte genomfördes så beskrivs bara resultaten från programmeringsdelen i detta avsnitt.

5.1 Bilder till V’

Som tidigare nämnts så byggs volymen $V'$ upp av "lager" av bilder. De bilder som hänvisas till i avsnitt 4.2.1 Inläsning och behandling av bilderna är följande:

![Figur 17. Bild till $V'$ vit cirkel](bild1.png)

![Figur 18. Bild till $V'$ svart kvadrat](bild2.png)

Figur 17 motsvarar bild1.gif och figur 18 bild2.gif. Det har valts mycket enkla bilder för att det skall vara enklare att urskilja dem i hologrammet. De är av exakt samma storlek, figur 18 har en vit ram runt sig som inte är synlig i detta dokument. En storlek på 80x80 pixlars valdes för att hålla nere beräkningstiden. Eftersom bara vit och svart används så blir kontrasten mellan bilderna lättare att se.

5.2 Bilder till V’ för optimering 3 och framåt

Från optimering 3 används den inverterade varianten av bild2.gif istället. Detta innebär alltså en vit kvadrat på svart bakgrund. De nya bilderna är:

![Figur 19. Bild till $V'$ vit cirkel](bild1.png)

![Figur 20. Bild till $V'$ vit kvadrat](bild2.png)

5.3 Transformbild för optimering 1

Det är här resultatet från avsnitt 4.2.6 Skapande av transformbild kan beskådas. Med bilderna i figur 17 och 18 samt med en DMD-storlek på 1024x768 (storleken på transformbilden) så genereras följande transformbild:

Figur 21. Transformbild för optimering 1

5.4 Beräkningstid för optimering 1

Matlab har en funktion som heter profile viewer vilken loggar bl.a. beräkningstid för en eller flera beräkningar. Loggarna från profile viewer ger mycket nyttig information om matlabbkoden, som bl.a. kan användas för att utföra optimering. I figur 22 kan man se hur lång tid matlabbprogrammet tog samt var i matlabbkoden tiden spenderades.

Matlabprogrammet helmholtz6a_ny tog 21267s på sig att slutföra, vilket är strax under 6 timmar. Datorn som användes för beräkningen är en AMD 64-bit 4000+ (4GHz) med 1024MB minne. Under programmets gång så är processorutnyttjandet 100 % och c:a 150MB av minnet utnyttjas. Med dessa uppgifter så får man en uppfattning om vilka enorma datorresurser som behövs för att skapa holografisk video.

Figur 22 visar också att 94 % av tiden spenderades på rad 78 vilken innehåller beräkningen:

\[ Us(x,y)=\text{sum}((\text{Ureshaped}./\text{xyzSqrt}).*\exp(-i*k.*
(xyzSqrt-((xD^2+yD^2)/(2*f)))); \]

Man kan också se att rad 78 har körts 786432 gånger vilket naturligtvis är storleken på DMD:n 1024x768. Mycket tid har spenderats för att försöka optimera denna beräkning. Komplexiteten i den ligger i exponentberäkningen med funktionen exp. Tyvärr har matlab ingen bättre funktion för detta, så det är svårt att optimera det vidare.
Figur 22. Matlabprofil för optimering 1

5.5 Beräkningstid för optimering 2

Meningen med matlabprogrammet *helmholtz6e* i appendix 2 är att förkorta beräkningstiden. Att använda sig av en uppdelad DMD vid beräkningarna gjorde att hela mitt internminne på 1024MB utnyttjades till max. Det behövdes t.o.m. mer minne så datorns växlingsfil utnyttjades också. Detta är inte bra eftersom då datorn läser från hårddisken så används processorn för detta också. Om man tittar på processorutnyttjandet under programmets gång så är den på 100 % större delen av tiden, men sjunker neråt 20 % då växlingsfilen används. Resultatet av allt detta är att programmet tar längre tid än det som används i optimering 1. Man kan självklart dela upp DMD:n i ännu mindre bitar, men ”overhead:n” för att sköta indexeringen och alla matriser för en allt för stor uppdelning gör att programmet går långsamt.

5.6 Beräkningstid för optimering 3

Matlabprogrammet *helmholtz7* i appendix 3 bör förkorta beräkningstiden ytterligare med *helmholtz6a* (appendix 1) som grund.

Figur 23 visar matlabprofilen för helmholtz7. Datorn som användes för beräkningen är en AMD 64-bit 3700+ (3.7GHz) med 1024MB minne. Beräkningen tog 12094s eller 3 timmar och 20 minuter. Det visade sig att för de bilder som användes (i avsnitt 5.2) så blev beräkningstiden ungefär 50 % av den mot i *helmholtz6a*. Då har hänsyn tagits till att en något långsammare processor användes vid detta tillfälle då 12094/21268 \( \approx \) 0.57 eller 57 %, alltså beräkningstiden för helmholtz7 dividerat med helmholtz6a. Detta är en klar förbättring, men man bör vara medveten om att ju mindre gemensamma svarta regioner som finns mellan bilderna desto närmare kommer vi beräkningstiden för *helmholtz6a*. Som nämnades i avsnittet om beräkningstiden för optimering 1 så ligger komplexiteten i exponentberäkningen exp. Med denna optimering så blir indatat för denna beräkning mycket enklare (noll) för de flesta av de punkter som beräknas. Det är alltså här större delen av tiden tjänas in.
5.7 Beräkningstid för optimering 4

Matlabprogrammet *helmholtz7a* i appendix 4 så bör förkorta beräkningstiden ytterligare med *helmholtz7* (appendix 3) som grund.

**helmholtz7a (1 call, 9674.016 sec)**

Generated 10-Mar-2006 17:40:03 using real time.
M-script in file D:\Mina_dokument\VExjobb\matlab\helmholtz7a.m

[Copy to new window for comparing multiple runs]

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Code</th>
<th>Calls</th>
<th>Total Time</th>
<th>% Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>122</td>
<td>US(x,y)=sum(Uspart.<em>exp(-i</em>K.*...</td>
<td>786432</td>
<td>8333.453 s</td>
<td>86.2%</td>
</tr>
<tr>
<td>113</td>
<td>xyzSqrt = sqrt(zprimArray.^2+(...</td>
<td>786432</td>
<td>534.922 s</td>
<td>5.5%</td>
</tr>
<tr>
<td>119</td>
<td>xyzSqrtNonZero(1,xyzSqrtZeroIn...</td>
<td>786432</td>
<td>301.594 s</td>
<td>3.9%</td>
</tr>
<tr>
<td>117</td>
<td>xyzSqrtZeroIndex = find(xyzSqr...</td>
<td>786432</td>
<td>252.906 s</td>
<td>2.6%</td>
</tr>
<tr>
<td>120</td>
<td>Uspart = (Ureshaped./xyzSqrtNo...</td>
<td>786432</td>
<td>91.781 s</td>
<td>0.9%</td>
</tr>
</tbody>
</table>

Other lines & overhead 73.353 s 0.8%

Totals 9674.016 s 100%

Figur 24. Matlabprofil för optimering 4

6 Slutdiskussion

7 Referenslista

7.1 Bok

7.2 Tidskriftsartikel


Tillgänglig på [http://www.opticsinfobase.org/abstract.cfm?id=53497](http://www.opticsinfobase.org/abstract.cfm?id=53497) (Acc. 06-03-24)

7.3 Referenser till Internet
Klaus Biedermann, ”Lippmann's and Gabor's Revolutionary Approach to Imaging”, Nobel Foundation (2005).


Wikipedia, the free encyklopedia, ”Diffraction”, (2005).


Appendix 1. Helmholtz6a

% Helmholtz6a, 2xfor-loops (nested) in transform-loop
% Have replaced 3xfor-loops (nested) with matrix operations for speed
% optimization
% Transform using Helmholtz wave equation

% Read the image into an array, the format is HeightxWide (y/x) (pixels)
image1Array = double(imread('circle.gif'));
image2Array = double(imread('rect.gif'));
% Values in imageArrays are normalized between 0 and 1
image1ArrayNorm = image1Array/max(max(image1Array));
image2ArrayNorm = image2Array/max(max(image2Array));
image1Size = size(image1Array)
image2Size = size(image2Array)

% resize image2Array if necessary, image1Array should always be larger or
% of equal size
if (image1Size(1,1)*image1Size(1,2))>(image2Size(1,1)*image2Size(1,2))
    image2A = zeros(image1Size(1,1),image1Size(1,2));
    image2A(1:(image2Size(1,1)),1:(image2Size(1,2)))=image2ArrayNorm;
    image2ArrayNorm=image2A;
end

% create array U, image1 put at zprim=1, image2 at zprim=2
% the format is HeightxWidexDepth (y/x/z) (pixels)
U = zeros(image1Size(1,1),image1Size(1,2),2);
U(:,:,1) = image1ArrayNorm;
U(:,:,2) = image2ArrayNorm;

% k is the wavenumber of the light used
% f is the focal length of the converging lens, 40cm
wavelength = 633e-9;
k = (2*pi)/wavelength;
f = 0.4;

% we use a 1024x768 DMD with 16x16 microns mirrors
xDMDsize = 768;
yDMDsize = 1024;
% create array Us where final hologram is stored
Us = zeros(xDMDsize,yDMDsize);

% calculate dimensions of V'
[ySize xSize zSize] = size(U);
VSize = xSize*ySize*zSize;

% all combinations of z',x',y' are stored in 3 arrays
% these 3 arrays "simulate" former "for-loops"
% it's reordered with z' as 1st, x' 2nd and y' 3rd (most inner loop)
% z' therefore is "1" for 1->VSize/2 and "2" in the remaining calculations
zprimArray=floor(1:(1/(xSize*ySize)):(zSize+1));
zprimArray=zprimArray(1:VSize);
% x' is "1" for 1->ySize, then "2" for 1->ySize,..."xSize" for 1->ySize
xprimArray=floor(1:(1/ySize):(xSize+1));
xprimArray=xprimArray(1:(VSize/zSize));
xprimArray=repmat(xprimArray,1,(VSize/(xSize*ySize)));
% y' is 1->ySize and then looped to 1->ySize again and so on...through
ySize
yprimArray=1:1:ySize;
yprimArray=repmat(yprimArray,1,(VSize/ySize));
% get x',y',z' in meter
xprimArray=51e-6.*xprimArray;
yprimArray=51e-6.*yprimArray;
% z' has offset f for 1->VSize/2 (when z'=1) and offset f+0.05 for the rest (z'=2)
% need to fix further when z'>2
zprimArray=f+51e-6.*zprimArray(1:VSize);
zprimArray(((VSize/2)+1):VSize)=0.05+zprimArray(((VSize/2)+1):VSize);

%U is reshaped, starting with z'=1 (image1) going through y=1->ySize for each x column in U
Ureshaped=reshape(U,1,VSize);

% transform-loop
xyzSqrt = zeros(1,VSize);
for x=1:xDMDsize
    % get x,y in meter
    xD = 16e-6*x;
    for y=1:yDMDsize
        yD = 16e-6*y;
        % calculate the contribution from a point in the volume V'
        xyzSqrt = sqrt(zprimArray.^2+(xD-xprimArray).^2+(yD-yprimArray).^2);
        Us(x,y)=sum((Ureshaped./xyzSqrt).*exp(-i*k.*(xyzSqrt-
                    ((xD^2+yD^2)/(2*f)))));
    end
    progress = 100*(x/xDMDsize)
end
UsReal=real(Us);
UsImag=imag(Us);
UsImagNorm=UsImag/max(max(UsImag));
% write transformed image
imwrite(UsImagNorm,'transfImg6a_ny.bmp');
Appendix 2. Helmholtz6e

% Helmholtz6e, 1xfor-loop!
% Have replaced for-loops with matrix-operations (vectorizing) for speed optimization
% for example "repmat" is now frequently used
% One drawback is that matrices becomes very big when larger images/DMD is used
% I've solved this by dividing DMD-hologram into typical 16 parts
% Transform using Helmholtz wave equation

% k is the wavenumber of the light used
% f is the focal length of the converging lens, 40cm
wavelength = 633e-9;
k = (2*pi)/wavelength;
f = 0.4;

% ----------
% *** DMD and Us ***
% we use a 1024x768 DMD with 16x16 microns mirrors
% ----------
xDMDsize = 768;
yDMDsize = 1024;
DMDsize=xDMDsize*yDMDsize;
% divide DMD in xDMDpsize x yDMDsize slices for later transform calculations
%divFactor = 16;
%DMDslices = divFactor^2;
%DMDsize = DMDsize/DMDslices;
%xDMDpsize = (xDMDsize/divFactor)*(yDMDsize/divFactor)/yDMDsize;
yDMDpsize = yDMDsize;

% Not using above divFactor we use xDMDsize parts
xDMDpsize = 1;
DMDslices = xDMDsize;
DMDsize = yDMDsize;

% create array Us where final hologram is stored
Us = zeros(xDMDsize,yDMDsize);
% ----------
% Read the image into an array, the format is HeightxWide (y/x) (pixels)
image1Array = double(imread('circle.gif'));
image2Array = double(imread('rect.gif'));
% Values in imageArrays are normalized between 0 and 1
image1ArrayNorm = image1Array/max(max(image1Array));
image2ArrayNorm = image2Array/max(max(image2Array));
image1Size = size(image1Array)
image2Size = size(image2Array)

% resize image2Array if necessary, image1Array should always be larger or of equal size
if (image1Size(1,1)*image1Size(1,2))>(image2Size(1,1)*image2Size(1,2))
    image2A = zeros(image1Size(1,1),image1Size(1,2));
    image2A(1:(image2Size(1,1)),1:(image2Size(1,2)))=image2ArrayNorm;
end

% ----------
% *** U(x',y',z') ***
% create array U, imag1 put at zprim=1, image2 at zprim=2
% the format is HeightxWidexDepth (y/x/z) (pixels)
U = zeros(image1Size(1,1),image1Size(1,2),2);
U(:,:,1) = image1ArrayNorm;
U(:,:,2) = image2ArrayNorm;
% calculate dimensions of V'
[ySize xSize zSize] = size(U);
VSize = xSize*ySize*zSize;
% U is reshaped, starting with z'=1 (image1) going through y=1->ySize for
each x column in U
Ureshaped=reshape(U,1,VSize);
Ureshaped=repmat(Ureshaped,DMDpsize,1);   % for transform calculation
%----------
% *** x',y',z' ***
% all combinations of z',x',y' are stored in 3 arrays
% these 3 arrays "simulate" former "for-loops"
% it's reordered with z' as 1st, x' 2nd and y' 3rd (most inner loop)
%----------
% z' therefore is "1" for 1->VSize/2 and "2" in the remaining calculations
zprimArray=floor(1:(1/(xSize*ySize)):(zSize+1));
zprimArray=zprimArray(1:VSize);
% z' has offset f for 1->VSize/2 (when z'=1) and offset f+0.05 for the rest
% need to fix further when z'>2
zprimArray=f+51e-6.*zprimArray(1:VSize);
% get z' in meter for z'=1
zprimArray(((VSize/2)+1):VSize)=0.05+zprimArray(((VSize/2)+1):VSize); % get
% z' in meter for z'>1
zprimArray=repmat(zprimArray,DMDpsize,1);    % replicate for transform
calculation
% x' is "1" for 1->ySize, then "2" for 1->ySize,...,"xSize" for 1->ySize
xprimArray=floor(1:(1/ySize):(xSize+1));
xprimArray=xprimArray(1:(VSize/zSize));
xprimArray=51e-6.*xprimArray;    % get x' in meter
xprimArray=repmat(xprimArray,DMDpsize,(VSize/(xSize*ySize)));    % for
% transform calculation
% y' is 1->ySize and then looped to 1->ySize again and so on...through
yprimArray=1:1:ySize;
yprimArray=51e-6.*yprimArray;    % get y' in meter
yprimArray=repmat(yprimArray,DMDpsize,(VSize/ySize));    % for transform
calculation
%----------
% *** x,y ***
% create x,y arrays
% x is 1 for 1->VSize*yDMDsize, then 2...xDMDsize
% y is 1->yDMDsize and then looped to 1->yDMDsize again and so on...through
xArray=floor(1:(1/(yDMDsize*VSize)):(xDMDpsize+1));
xArray=xArray(1:DMDpsize*VSize);
xArrayP=(reshape(xArray',VSize,DMDpsize))';
xArray=16e-6.*xArrayP;    % get x in meter
yArray=1:1:yDMDsize;
yArray=16e-6.*yArray;    % get y in meter
yArray=repmat(yArray',xDMDpsize,VSize);    %----------
% --------
% *** TRANSFORM ***
% calculate the contribution from a point in the volume V'
for n=1:1:DMDslices
    xyzSqrt = sqrt(zprimArray.^2+(xArray-xprimArray).^2+(yArray-
yprimArray).^2);
    xStart = xDMDpsize*(n-1)+1;
    xEnd = xStart + xDMDpsize-1;
    UsTemp = sum(((Ureshaped./xyzSqrt).*exp(-i*k.*(xyzSqrt-
    ((xArray.^2+yArray.^2)/(2*f))))')';
    Us(xStart:xEnd,1:yDMDsize) = (reshape(UsTemp',yDMDsize,xDMDpsize))';
    xArrayP = xDMDpsize+xArrayP; % get new set of x values
    xArray = 16e-6.*xArrayP; % and get them in meter
    progress = 100*(n/DMDslices)
end
%
% --------

%UsReal=real(Us);
UsImag=imag(Us);
UsImagNorm=UsImag/max(max(UsImag));
% write transformed image
imwrite(UsImagNorm,'transfImg6e.bmp');
Appendix 3. Helmholtz7

% Helmholtz7, 2xfor-loops (nested) in transform-loop
% Have replaced 3xfor-loops (nested) with matrix operations for speed
% optimization
% Further speed optimization by manually seeing that black parts in the
% input images will be calculated faster as they won't do anything for the
% final result
% Transform using Helmholtz wave equation

% Read the image into an array, the format is HeightxWide (y/x) (pixels)
image1Array = double(imread('circle.gif'));
image2Array = double(imread('rect2.gif'));
% Values in imageArrays are normalized between 0 and 1, 0 being black
image1ArrayNorm = image1Array/max(max(image1Array));
image2ArrayNorm = image2Array/max(max(image2Array));
imageSize = size(image1Array)
image2Size = size(image2Array)
clear image1Array image2Array;

% resize image2Array if necessary, image1Array should always be larger or
% of equal size
if (image1Size(1,1)*image1Size(1,2))>(image2Size(1,1)*image2Size(1,2))
   image2A = zeros(image1Size(1,1),image1Size(1,2));
   image2A(1:(image2Size(1,1)),1:(image2Size(1,2)))=image2ArrayNorm;
   image2ArrayNorm=image2A;
end

% create array U, image1 put at zprim=1, image2 at zprim=2
% the format is HeightxWidthxDepth (y/x/z) (pixels)
U = zeros(image1Size(1,1),image1Size(1,2),2);
U(:,:,1) = image1ArrayNorm;
U(:,:,2) = image2ArrayNorm;
clear image1ArrayNorm image2ArrayNorm;

% create array Uunion that lets me see where both images are black (0)
% UNonzero holds yprim,xprim coordinates of nonzero values in Uunion
Uunion = U(:,:,1) + U(:,:,2);
UunionNorm = Uunion/max(max(Uunion)); % normalize values between 0 and 1
UunionNormCeil = ceil(UunionNorm); % raises all values >0 to 1
UunionNormCeil = UunionNormCeil*UunionNorm;
clear Uunion UunionNorm yprimNonzero xprimNonzero yprimNZsize;

% k is the wavenumber of the light used
% f is the focal length of the converging lens, 40cm
wavelength = 633e-9;
k = (2*pi)/wavelength;
f = 0.4;

% we use a 1024x768 DMD with 16x16 microns mirrors
xDMDsize = 768;
yDMDsize = 1024;
% create array Us where final hologram is stored
Us = zeros(xDMDsize,yDMDsize);

% calculate dimensions of V'
[yprimSize xprimSize zprimSize] = size(U);
VSize = xprimSize*yprimSize*zprimSize;
% all combinations of z',x',y' are stored in 3 arrays
% these 3 arrays "simulate" former "for-loops"
% it's reordered with z' as 1st, x' 2nd and y' 3rd (most inner loop)
% z' therefore is "1" for 1->VSize/2 and "2" in the remaining calculations
% x' is "1" for 1->yprimSize, then "2" for 1->yprimSize,...,"xprimSize" for 1->yprimSize
% y' is 1->yprimSize and then looped to 1->yprimSize again through VSize

zprimArray = floor(1:(1/(xprimSize*yprimSize)):(zprimSize+1));
zprimArray = zprimArray(1:VSize);
xprimArray = floor(1:(1/yprimSize):(xprimSize+1));
xprimArray = xprimArray(1:(VSize/zprimSize));
xprimArray = repmat(xprimArray,1,(VSize/(xprimSize*yprimSize)));

yprimArray = 1:1:yprimSize;
yprimArray = repmat(yprimArray,1,(VSize/yprimSize));

% get x',y',z' in meter
xprimArray = 51e-6.*xprimArray;
yprimArray = 51e-6.*yprimArray;
zprimArray = zprimArray.*UunionArray;
zprimArray(((VSize/2)+1):VSize)=0.05+zprimArray(((VSize/2)+1):VSize);

UunionArray = reshape(UunionNormCeil,1,VSize/zprimSize);
UunionArray = repmat(UunionArray,1,zprimSize);

% set x',y',z' to zero when UunionArray is zero to speed calculation up
zprimArray = zprimArray.*UunionArray;
xprimArray = xprimArray.*UunionArray;
yprimArray = yprimArray.*UunionArray;

%U is reshaped, starting with z'=1 (image1) going through y=1->yprimSize
% for each x column in U
Ureshaped=reshape(U,1,VSize);

xyzSqrt = zeros(1,VSize);

% set x,y,v to zero only when UunionArray is zero
xDzero = zeros(1,VSize)+UunionArray;
yDzero = zeros(1,VSize)+UunionArray;
xyzSqrtNonZero = ones(1,VSize);

for x=1:xDMDsize
    % get x,y in meter
    xD = xDzero.*(16e-6*x);
    for y=1:yDMDsize
        yD = yDzero.*(16e-6*y);
        xyzSqrt = sqrt(zprimArray.^2+(xD-xprimArray).^2+(yD-yprimArray).^2);
        % find where xyzSqrt is zero and set it to one, to avoid division by zero.
        % This won't make errors in Us cause Ureshaped is zero for every time xyzSqrt is zero.
        % so it will result in zero anyway
        xyzSqrtZeroIndex = find(xyzSqrt == 0);
        xyzSqrtNonZero = zeros(1,VSize);
        xyzSqrtNonZero(xyzSqrtZeroIndex) = 1;
        UsPart = (Ureshaped./xyzSqrtNonZero);
        % calculate the contribution from a point in the volume V'
        Us(x,y)=sum(UsPart.*exp(-i*k.*((xD.^2+yD.^2)./(2*f))));
    end
    progress = 100*(x/xDMDsize)
end

clear zprimArray xprimArray yprimArray xyzSqrt xDzero yDzero;
clear xyzSqrtNonZero xD yD xyzSqrt xyzSqrtZeroIndex UsPart;
UsReal=real(Us);
UsImag=imag(Us);
UsImagNorm=UsImag/max(max(UsImag));

% write transformed image
imwrite(UsImagNorm,'transfImg7.bmp');
Appendix 4. Helmholtz7a

% Helmholtz7a, 2xfor-loops (nested) in transform-loop
% Have replaced 3xfor-loops (nested) with matrix operations for speed
% optimization
% Further speed optimization by manually seeing that black parts in the
% input images will be calculated faster as they won't do anything for the
% final result
% Transform using Helmholtz wave equation

% Read the image into an array, the format is HeightxWide (y/x) (pixels)
image1Array = double(imread('circle.gif'));
image2Array = double(imread('rect2.gif'));
% Values in imageArrays are normalized between 0 and 1, 0 being black
image1ArrayNorm = image1Array/max(max(image1Array));
image2ArrayNorm = image2Array/max(max(image2Array));
image1Size = size(image1Array)
image2Size = size(image2Array)
clear image1Array image2Array;

% resize image2Array if necessary, image1Array should always be larger or
% of equal size
if (image1Size(1,1)*image1Size(1,2))>(image2Size(1,1)*image2Size(1,2))
    image2A = zeros(image1Size(1,1),image1Size(1,2));
    image2A(1:(image2Size(1,1)),1:(image2Size(1,2)))=image2ArrayNorm;
    image2ArrayNorm=image2A;
end
clear image2A;

% create array U, image1 put at zprim=1, image2 at zprim=2
% the format is HeightxWidexDepth (y/x/z) (pixels)
U = zeros(image1Size(1,1),image1Size(1,2),2);
U(:,:,1) = image1ArrayNorm;
U(:,:,2) = image2ArrayNorm;
clear image1ArrayNorm image2ArrayNorm;

% create array Uzprim1 and Uzprim2 that lets me see where image1 and image
% 2 are black (0)
Uzprim1 = U(:,:,1);
Uzprim2 = U(:,:,2);
Uzprim1Norm = Uzprim1/max(max(Uzprim1)); % normalize values between 0 and 1
Uzprim2Norm = Uzprim2/max(max(Uzprim2));
Uzprim1NormCeil = ceil(Uzprim1Norm); % raises all values >0 to 1
Uzprim2NormCeil = ceil(Uzprim2Norm);
clear Uzprim1 Uzprim2 Uzprim1Norm Uzprim2Norm;

% k is the wavenumber of the light used
% f is the focal length of the converging lens, 40cm
wavelength = 633e-9;
k = (2*pi)/wavelength;
f = 0.4;

% we use a 1024x768 DMD with 16x16 microns mirrors
xDMDsize = 768;
yDMDsize = 1024;
% create array Us where final hologram is stored
Us = zeros(xDMDsize,yDMDsize);

% calculate dimensions of V'
yprimSize xprimSize zprimSize = size(U);
VSize = xprimSize*yprimSize*zprimSize;

% all combinations of z',x',y' are stored in 3 arrays
% it's reordered with z' as 1st, x' 2nd and y' 3rd (most inner loop)
% z' therefore is "1" for 1->VSize/2 and "2" in the remaining calculations
zprimArray = floor(1:(1/(xprimSize*yprimSize)):(zprimSize+1));
zprimArray = zprimArray(1:VSize);
% x' is "1" for 1->yprimSize, then "2" for 1->yprimSize,...,"xprimSize" for 1->xprimSize
xprimArray = floor(1:(1/yprimSize):(xprimSize+1));
xprimArray = xprimArray(1:(VSize/zprimSize));
xprimArray = repmat(xprimArray,1,(VSize/(xprimSize*yprimSize)));
% y' is 1->yprimSize and then looped to 1->yprimSize again through VSize
yprimArray = 1:1:yprimSize;
yprimArray = repmat(yprimArray,1,(VSize/yprimSize));
% get x',y',z' in meter
xprimArray = 51e-6.*xprimArray;
yprimArray = 51e-6.*yprimArray;
% z' has offset f for 1->VSize/2 (when z'=1)and offset f+0.05 for the rest
% (z'='2), need to fix further when z'>2
zprimArray = f+51e-6.*zprimArray(1:VSize);
zprimArray(((VSize/2)+1):VSize)=0.05+zprimArray(((VSize/2)+1):VSize);

% create UunionArray from the two parts Uzprim1 and Uzprim2, reshape it to % get same format as zprimArray, xprimArray and yprimArray
UnionArray = zeros(1,VSize);
U1temp = reshape(Uzprim1NormCeil,1,VSize/zprimSize);
U2temp = reshape(Uzprim2NormCeil,1,VSize/zprimSize);
UnionArray(1,1:VSize/zprimSize) = U1temp;
UnionArray(1,((VSize/zprimSize)+1):VSize) = U2temp;
clear U1temp U2temp;

% set x',y',z' to zero when UnionArray is zero to speed calculation up
zprimArray = zprimArray.*UnionArray;
xprimArray = xprimArray.*UnionArray;
yprimArray = yprimArray.*UnionArray;

% U is reshaped, starting with z'=1 (image1) going through y=1->yprimSize % for each x column in U
Ureshaped=reshape(U,1,VSize);

% transform-loop
xyzSqrt = zeros(1,VSize);
% set x,y values to zero only when UnionArray is zero
xDzero = zeros(1,VSize)+UnionArray;
yDzero = zeros(1,VSize)+UnionArray;
xyzSqrtNonZero = ones(1,VSize);
for x=1:xDMDsize
  % get x,y in meter
  xD = xDzero.*(16e-6*x);
  for y=1:yDMDsize
    yD = yDzero.*(16e-6*y);
    xyzSqrt = sqrt(zprimArray.^2+(xD-xprimArray).^2+(yD-yprimArray).^2);
    % find where xyzSqrt is zero and set it to one, to avoid division by % zero. This won't make errors in Us cause Ureshaped is zero for every % time xyzSqrt is, so it will result in zero anyway
    xyzSqrtZeroIndex = find(xyzSqrt == 0);
    xyzSqrtNonZero = xyzSqrt;
    xyzSqrtNonZero(1,xyzSqrtZeroIndex) = 1;
UsPart = (Ureshaped./xyzSqrtNonZero);
% calculate the contribution from a point in the volume V'
Us(x,y)=sum(UsPart.*exp(-i*k.*(xyzSqrt-((xD.^2+yD.^2)./(2*f)))));
end
progress = 100*(x/xDMsize)
end
clear zprimArray xprimArray yprimArray xyzSqrt xDzero yDzero;
clear xyzSqrtNonZero xD yD xyzSqrt xyzSqrtZeroIndex UsPart;
%UsReal=real(Us);
UsImag=imag(Us);
UsImagNorm=UsImag/max(max(UsImag));
% write transformed image
imwrite(UsImagNorm,'transfImg7a.bmp');
Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/.

Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Lars-Åke Waldemarsson
Sammanfattning
Detta examensarbete utgår ifrån en artikel i vilken en metod för att skapa holografisk video beskrivs. Syftet med arbetet är att återskapa denna metod. Metoden bygger på projicering av holoagram med hjälp av delar från en projektor, en laser och några linser.


Nyckelord
- datagenererade holoagram
- fouriertransform
- diffraktion
- DMD