CHALMERS
UNIVERSITY OF TECHNOLOGY

Förbättrad skanning av kvitton med mobilkameran i mörka förhållanden

Kandidatarbete inom Data- och Informationsteknik

DANIEL SUNNERBERG
JACOB LUNDBERG
MAGNUS GUSTAVER
GUSTAV BERGSTRÖM
JOAKIM ELIASSON
VICTOR CHRISTOFFERSSON

Chalmers tekniska högskola
Göteborgs universitet
Institutionen för Data- och Informationsteknik
Göteborg, Sverige, 2017

Kandidatarbete

Förbättrad skanning av kvitton med mobilkameran i mörka förhållanden

Daniel Sunnerberg
Gustav Bergström
Jacob Lundberg
Joakim Eliasson
Magnus Gustaver
Victor Christoffersson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för Data- och Informationsteknik
Chalmers tekniska högskola
Göteborgs universitet
Göteborg, Sverige, 2017

Förbättrad scanning av kvitton med
mobilkameran i mörka förhållanden

Daniel Sunnerberg
Gustav Bergström
Jacob Lundberg
Joakim Eliasson
Magnus Gustaver
Victor Christoffersson

Handledare: Sven Knutsson, Institutionen för data- och informationteknik
Examinator: Arne Linde, Institutionen för data- och informationteknik

Kandidatarbete 2017:30
Institutionen för Data- och Informationsteknik
Chalmers tekniska högskola
Göteborgs universitet
SE-412 96 Göteborg
Telefon: +46 (0)31 772 1000

Omslag: Visualisering av en person som med hjälp av sin telefon använder sig av tekniken att skanna ett kvitto.

Institutionen för Data- och Informationsteknik
Göteborg, Sverige, 2017

Förbättrad scanning av kvitton med mobilkameran i mörka förhållanden
Daniel Sunnerberg
Gustav Bergström
Jacob Lundberg
Joakim Eliasson
Magnus Gustaver
Victor Christoffersson
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs universitet

Sammandrag

Mobiltelefonen har kommit att bli ett användbart verktyg för att manuellt skanna in dokument såsom kvitton. Detta tack vare dess ständigt förbättrade kameraprestanda samt dess lättillgänglighet. Det kan dock förekomma problem i mörka förhållanden, då störningar uppstår i bilden och kamerablixten reflekteras i kvittots blanka yta.

Rapporten beskriver en undersökning av hur olika bildbehandlingsmetoder kan användas för att underlätta skanning av kvitton i mörka förhållanden. Den primära lösningen som utvecklats går ut på att ta en video med mobilkameran för att sedan slå samman de bildrutor som har högst kvalitet till en så kallad superbild. Denna lösning jämförs med kameratekniken HDR, som sammanfogar ett antal bilder tagna med olika exponeringstid, för att ge ett stort ljusomfång. Jämförelsen görs i syfte att hitta den teknik som bäst kan förbättra kvittoskanningen.

Texten på det kvitto som skannats läses sedan av med hjälp av en maskinläsare, och bör därmed vara så tydlig som möjligt. Därför har även bildbehandlingsmetoder för att försöka öka kontrasten mellan ljusa och mörka delar på kvittot, och på så vis framhäva texten, undersökts.

Då blixtreflektioner ej uppstår har uppmätta resultat visat att kamerans inbyggda HDR-funktion presterar bättre än videobehandling och egenutvecklade HDR-lösningar. Vidare har det påvisats att videolösningen har potential att eliminera reflektioner om dessa uppkommer. Däremot har sådana reflektioner ej visat sig vara vanligt förekommande.

För att demonstrera resultaten som uppnåddes, utvecklades en mobilapplikation som kan skanna in kvitton i realtid. Rapporten beskriver även utvecklingen av denna applikation, samt en tillhörande serverapplikation som sköter de prestandakrävande bildbehandlingsoperationerna.

Nyckelord: Maskinläsning, Kvitto, Text, HDR, Superbild, OpenCV, Bildbehandling

Abstract

The mobile phone has become a useful tool for manually scanning documents such as receipts. This is because of its constantly improved camera performance as well as its accessibility. However, problems may arise in dark conditions, as disturbances occur in the image and the camera flash is reflected on the shiny surface of the receipt.

This report describes an investigation of how different image processing methods can be used to facilitate the scanning of receipts in dark conditions. The primary solution that was developed is to film the receipt with the mobile camera and then merge the highest quality frames into a so-called super image. This solution is compared to the camera technique HDR, which combines a number of images taken with different exposure times, to provide a high dynamic range. The comparison is made in order to find the technique that improves the scanning of the receipt in the best way possible.

The text on the receipt that has been scanned is then read using optical character recognition and should thus be as clear as possible. Therefore, image processing methods that try to increase the contrast between light and dark parts of the receipt, thus highlighting the text, has also been investigated.

When flash reflections do not occur, measured results show that the camera's built-in HDR function performs better than video processing and self developed HDR solutions. Although, the video solution shows potential to eliminate reflections if they occur. However, such reflections have not been found to be common.

In order to demonstrate the achieved results, a mobile application that can scan receipts in real-time was developed. The report also describes the development of this application, as well as an associated server application that handles the performance demanding image processing operations.

Keywords: OCR, Receipt, Text, HDR, Super image, OpenCV, Image Processing

Ordlista

API	Application Programming Interface. Specifikation av hur en specifik programvara kan användas.
Bildruta	Enskild bild i en video.
CRF	Camera Response Function, sv. *kamerasvarsfunktion*.
HDR	High Dynamic Range. Kamerateknik som producerar bilder med stort intensitetsomfång.
JSON	JavaScript Object Notation. Textbaserat format som används vid utbyte av data.
Klient	Program som kommunicerar med en server.
Levenshteinavstånd	Mått på hur mycket som skiljer två textsträngar åt.
Maskinläsning	Teknik för att omvandla bilder av tryckt text till bokstäver och siffror i digitalt format.
OpenCV	Programbibliotek för bildbehandling.
RGB	Färgsystem som består av tre kanaler: röd, grön och blå. Dessa kanaler kan blandas för att skapa olika färger.
Skarphetsvärde	Numeriskt värde som indikerar hur skarp en bild är.
Superbild	Bild som skapats med de bästa delarna från olika bildrutor från en video.
Xamarin	Verktyg för att utveckla plattformsoberoende mobilapplikationer.

Förord

Denna rapport beskriver ett kandidatarbete som genomfördes vid institutionen för data- och informationsteknik vid Chalmers tekniska högskola år 2017.

Vi vill tacka Sven Knutsson för handledning genom arbetet och Claes Ohlsson på fackspråk för hjälp med rapportskrivandet. Vi vill även tacka Bokio för den vägledning de har bidragit med, samt för att vi har fått möjligheten att genomföra detta kandidatarbete hos dem.

Innehåll

1
Inledning

Bokio[1] är ett företag som vill underlätta bokföringsprocessen för småföretagare genom att låta deras digitala system göra en så stor del av jobbet som möjligt. En viktig del av bokföringsprocessen är att överföra information om inköp från kvitton till bokföringssystemet. För detta vill Bokio ge användaren möjlighet att skanna kvitton, för att sedan låta en maskinläsare läsa av dem och omvandla informationen till digitalt format. Systemet kan då ge ett förslag på vad som ska bokföras, och hur. För att maskinläsaren ska kunna läsa av den information som är nödvändig från kvittot ställs krav på att bilden är av tillräckligt hög kvalitet.

I takt med att mobiltelefonen har blivit en naturlig del i vardagen så har dess användningsområde ständigt utökats. Det är numera vanligt att mobiltelefonen har en inbyggd kamera av god kvalitet samt hårdvara kraftig nog för att klassas som en dator. Tack vare detta, tillsammans med att mobiltelefonen ofta är tillgänglig, så har den blivit ett användbart verktyg för att skanna dokument, såsom kvitton.

Vid skanning i mörka förhållanden kan det dock lätt uppstå problem. Mobilkamerans autofokus fungerar ofta sämre i mörker och det kan dessutom uppkomma brus i bilden. En naturlig lösning på detta är att använda kamerans blixt vid skanning av dokumentet. Om kvittots yta är glansig kan däremot en blixtreflektion uppstå, vilket gör det området oläsligt.

1.1 Syfte

Projektet är av en undersökande natur och det primära syftet är att hitta en förbättringsmetod som lyckas förbättra kvaliteten på kvitton som är skannade i mörka förhållanden. Både existerande metoder och egenutvecklade lösningar analyseras.

Vidare ska den eller de lösningar som ger bäst resultat implementeras i en mobilapplikation. I denna applikation ska kvaliteten på kvittobilden förbättras i realtid.

1.2 Uppgift

För att förbättra skanning av kvitton i mörka förhållanden har tre förslag till förbättringsmetoder tagits fram. Metoderna använder olika tekniker för att få fram en bättre slutbild på ett kvitto. Projektets uppgift är att utveckla och undersöka dessa metoder.

Videobehandling Spela in en video och sammanfoga delar av de bästa bildrutorna till en bild i syfte att eliminera blixtreflektioner.

HDR	Använda HDR-algoritmer för att förbättra kvittots kontrast mellan text och bakgrund. Mobilkamerans inbyggda HDR jämförs med andra existerande HDR-algoritmer.
Bildförbättring	Applicera olika filter och tröskeloperationer för att förtydliga kvittots text.

En förbättringsmetod anses vara lyckad då resultatet som metoden producerar förbättrar maskinläsning av kvittobilden som behandlats. Detta innebär att kvittots avlästa text från den förbättrade bilden skiljer sig mindre från den faktiska texten på det analoga kvittot än vad den avlästa texten från ett vanligt foto gör.

1.3 Avgränsningar

För att säkerställa god kvalitet på de metoder som utvecklas så kommer testning att ske mot en extern färdigutvecklad maskinläsare. Projektet kommer alltså inte att innefatta utveckling av en egen maskinläsare.

Vidare kommer projektet i första hand inte heller att fokusera på användarvänlighet när det gäller för användaren att själv skanna kvitton med mobilkameran. Detta på grund av att det primära målet är att förbättra bilderna på ett algoritmiskt vis.

Slutligen har det i samråd med Bokio beslutats att de utvecklade algoritmernas prestanda inte ska vara i fokus. Det är viktigare att uppnå goda resultat än att algoritmerna går snabbt, då Bokio själva kan optimera detta i efterhand.

2

Teori

I syfte att utveckla och undersöka lösningsförslagen i avsnitt 1.2 kommer ett flertal metoder för bildhantering att användas. För att ge vidare förståelse om hur dessa fungerar, beskrivs de mer i detalj i detta kapitel.

2.1 Bildförbättring

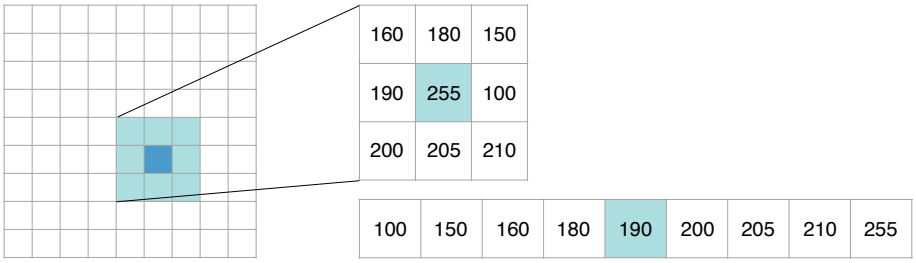
För att förbättra en redan existerande bild och göra den lättare att avläsa med maskinläsning finns det olika metoder som kan appliceras på bilden. Om en bild skulle vara tagen i mörka förhållanden är det vanligt att den innehåller störningar i form av brus. Sådana störningar kan orsaka att texten i bilden läses av felaktigt. För att reducera störningar i bilder kan därför en förbättringsmetod i form av ett utjämnande filter användas. Dessa filter ersätter avvikande bildpunkter med något som påminner om dess närliggande bildpunkter.

Vidare består ett kvitto ofta av mörk text på en ljus bakgrund. Genom att använda tröskeloperationer kan kontrasten mellan det mörka och det ljusa i en bild ökas för att tydligare framhäva texten och göra den lättare att läsa av.

2.1.1 Utjämnande filter

Det finns ett flertal sätt att gå tillväga för att minska påverkan av brus. Bland annat finns det filter som bygger på Gaussisk oskärpa men även de som utgår från medianen hos ett närområde. Båda typerna av filter jämnar ut en bild genom att ersätta varje bildpunkt med en uppskattning av vilken färg den bör ha med avseende på dess omgivning.

För ett medianfilter[2] studeras ett kvadratiskt område av en specificerad storlek runt varje bildpunkt. En färgkanal av intresse väljs ut, vilket oftast är den för gråskala. Därefter studeras bildpunktsvärdet för kanalen för varje bildpunkt i området. Medianen bland dessa värden ersätter slutligen den bildpunkt som var i centrum av det studerade området. Metoden tar därmed bort brus av impulskaraktär. I Figur 2.1 nedan ersätts en bildpunkt med medianbildpunkten i dess omringande kvadrat med sidlängden 3. Den utvalda bildpunkten har ursprungligen bildpunktsvärdet 255 och ersätts av en bildpunkt med värdet 190.

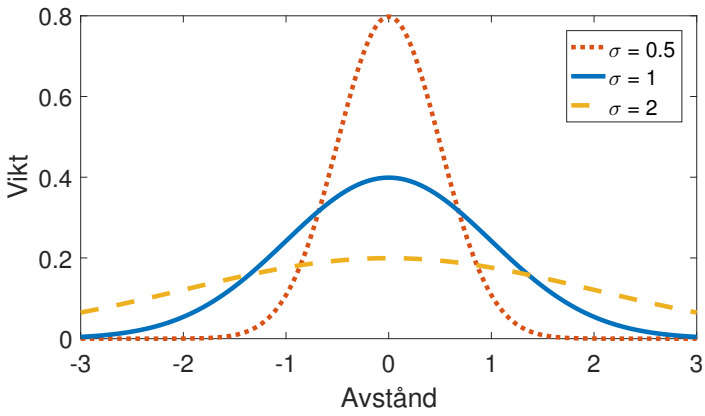


Figur 2.1: En avvikande bildpunkt ersätts med hjälp av ett medianfilter.

Ett Gaussiskt filter[3] räknar istället ut varje bildpunktsvärde utifrån en viktad summa av dess omgivning. Vikten för hur mycket omgivningen påverkar följer en Gaussisk fördelning, även kallat normalfördelning, vilket innebär att närliggande bildpunkter påverkar mer än de som är avlägsna. Den Gaussiska fördelningen defineras som

$$G(r, \sigma) = \frac{e^{-r^2/(2\sigma^2)}}{\sqrt{2\pi\sigma^2}}$$

där r är avståndet från den undersökta bildpunkten, σ är standardavvikelsen. Figur 2.2 visar en grafisk representation av hur fördelningen ser ut vid olika σ.



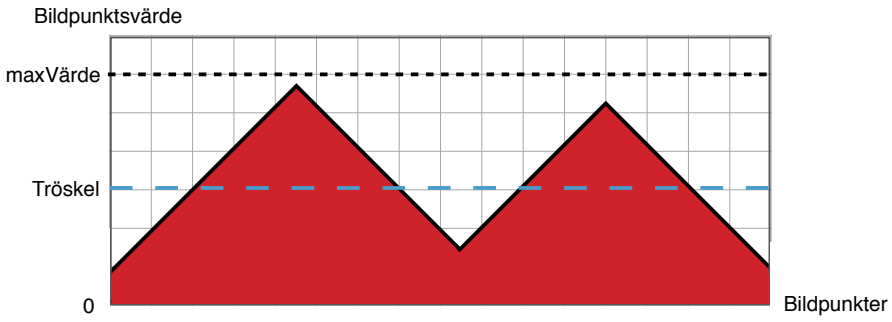
Figur 2.2: Normalfördelningen vid tre olika standardavvikelser. Vid ett lågt värde på standardavvikelsen viktas närliggande punkter starkare än vid ett högt värde.

2.1.2 Tröskeloperationer

Tröskeloperationer behandlar bilder med hjälp av ett tröskelvärde som jämförs med värdet på de bildpunkter i bilden som ska behandlas. Bildpunkter som har ett lägre värde än tröskelvärdet behandlas på ett visst sätt medan de bildpunkter som har ett högre behandlas på ett annat sätt. På så vis kan vissa objekt i bilden framhävas, exempelvis den mörka texten mot den vita bakgrunden på ett kvitto. Tröskelvärdet som används i metoden kan antingen väljas manuellt eller bestämmas med hjälp av Otsus metod[4].

Otsus metod är ett tillvägagångssätt där det optimala tröskelvärdet för en tröskeloperation eftersträvas. Metoden försöker minimera spridningen i förgrund och bakgrund, genom att studera de områden som skulle hamna över och under tröskelvärdet. Variansen i områdena viktas med hur stor andel av bilden som skulle komma att ingå i området. Slutligen tas medelvärdet av de två viktade varianserna, vilket ger ett värde som kan användas som ett betyg på hur väl det valda tröskelvärdet fungerar. Metoden försöker minimera detta värde genom att variera tröskelvärdet. Det tröskelvärde som användes vid det minsta uppnådda värdet för betyg anses som det optimala tröskelvärdet för att både ljusa och mörka områden ska representeras väl.

När ett tröskelvärde väl är bestämt kan olika tröskeloperationer användas. Det som skiljer de olika operationerna åt är hur de behandlar bildpunkter med ett värde över eller under tröskelvärdet. Fem olika operationer studeras och för att illustrera hur de fungerar används en tänkt bild med bildpunktsvärdet $src(x, y)$ på position (x, y) i bilden, vilket syns i Figur 2.3 nedan. För varje tröskeloperation visas sedan vilka nya värden, $dst(x, y)$, bildpunkterna antar i den behandlade bilden. Den streckade linjen i grafen representerar det valda tröskelvärdet. Den punktade svarta linjen representerar ett specificerat maxvärde som bildpunkter sätts till i vissa av tröskeloperationerna.

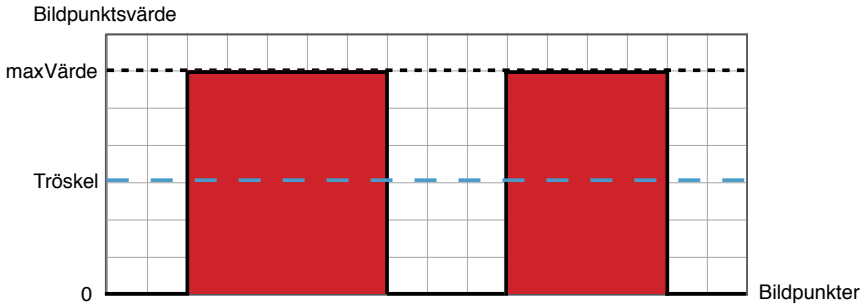


Figur 2.3: Referensfigur för att illustrera tröskeloperationer.

- **Binär avbildning**
  En binär avbildning[5] innebär att en bild behandlas så att bildpunkterna endast kan anta två olika värden, vanligtvis svart och vit. Bildpunkter med ett lägre värde än tröskelvärdet sätts till 0, vilket innebär svart. Övriga bildpunkter sätts till det specificerade maxvärdet, vilket vanligtvis väljs till 255, som innebär vitt. Operationen kan uttryckas som:

$$dst(x, y) = \begin{cases} maxVärde & \text{om } src(x, y) > tröskel \\ 0 & \text{annars} \end{cases}$$

  Hur bildpunkterna från referensbilden i Figur 2.3 behandlas illustreras i Figur 2.4 nedan. I figuren syns att samtliga bildpunkter antingen antar värdet 0 eller det specificerade maxvärdet.

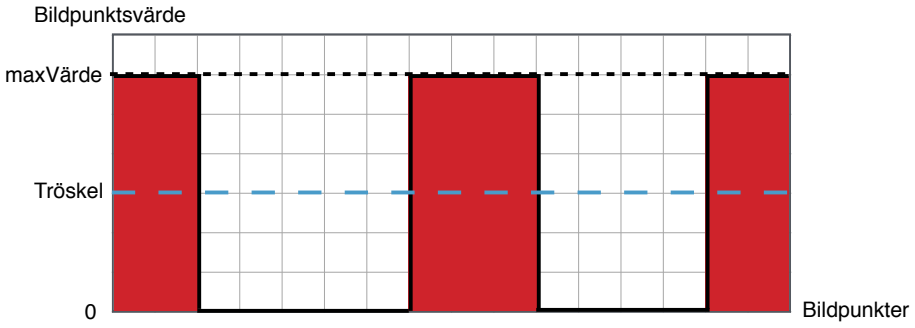


Figur 2.4: Behandling av bildpunkter vid en binär avbildning.

- **Invers binär avbildning**
  Även i en invers binär avbildning[6] antar bildpunkterna endast två olika värden. Vilka bildpunkter som sätts till 0 och det specificerade maxvärdet är dock inverterat jämfört med en vanlig binär avbildning. Operationen kan uttryckas som:

$$dst(x,y) = \begin{cases} 0 & \text{om } src(x,y) > tröskel \\ maxVärde & \text{annars} \end{cases}$$

  Hur bildpunkterna från referensbilden i Figur 2.3 behandlas visas i Figur 2.5 nedan. I figuren syns att bildpunktsvärdena är inverterade jämfört med i Figur 2.4.

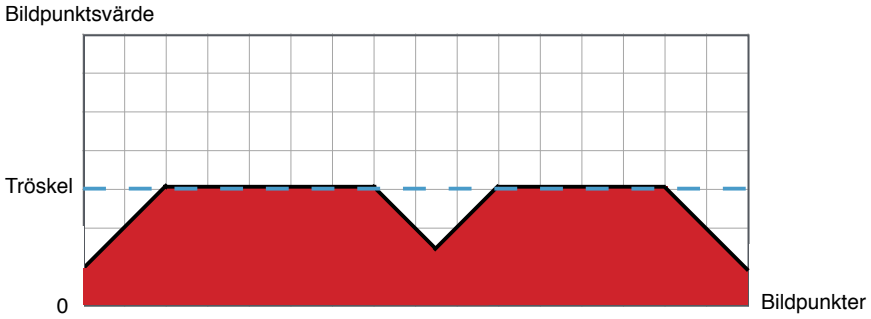


Figur 2.5: Behandling av bildpunkter vid en invers binär avbildning.

- **Stympning**
  Vid en stympning[7] begränsas de bildpunkter som har ett högre värde än tröskelvärdet till just tröskelvärdet. Resterande bildpunkter behåller sitt värde. Operationen kan uttryckas som:

$$dst(x,y) = \begin{cases} tröskel & \text{om } src(x,y) > tröskel \\ src(x,y) & \text{annars} \end{cases}$$

  Hur bildpunkterna från referensbilden i Figur 2.3 behandlas visas i Figur 2.6 nedan.

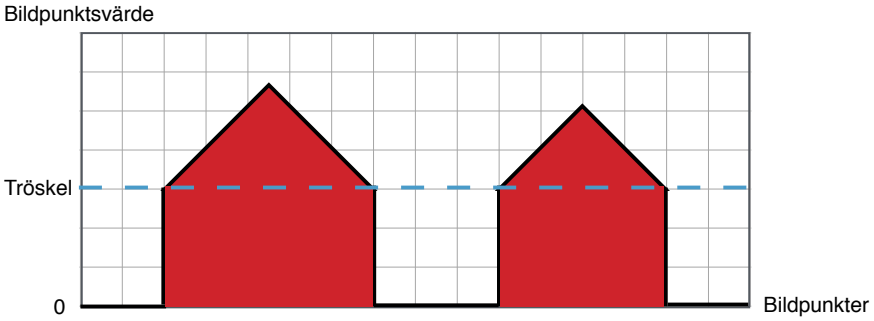


Figur 2.6: Behandling av bildpunkter vid en stympning.

- **Till noll**
  En till noll-operation[8] behåller värdet på de bildpunkter som har ett högre värde än tröskelvärdet. Resterande bildpunkters värde sätts till 0. Operationen kan uttryckas som:

$$dst(x,y) = \begin{cases} src(x,y) & \text{om } src(x,y) > tröskel \\ 0 & \text{annars} \end{cases}$$

  Hur bildpunkterna från referensbilden i Figur 2.3 behandlas visas i Figur 2.7 nedan.

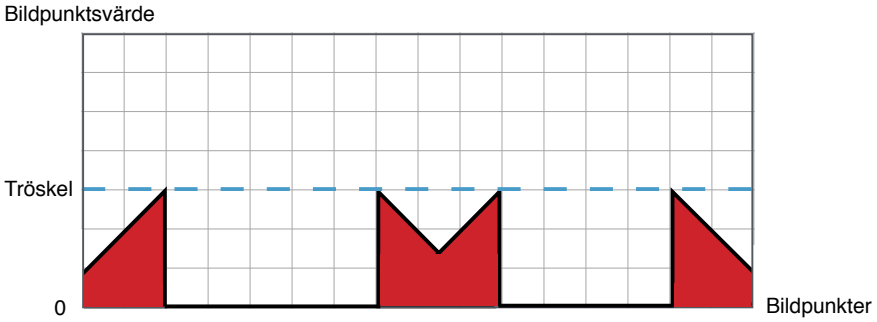


Figur 2.7: Behandling av bildpunkter vid en till noll-operation.

- **Invers till noll**
  En invers till noll-operation[9] sätter de bildpunkter som har ett högre värde än tröskelvärdet till 0. Resterande bildpunkter behåller sitt värde. Operationen kan uttryckas som:

$$dst(x,y) = \begin{cases} 0 & \text{om } src(x,y) > tröskel \\ src(x,y) & \text{annars} \end{cases}$$

  Hur bildpunkterna från referensbilden i Figur 2.3 behandlas visas Figur 2.8 nedan.



Figur 2.8: Behandling av bildpunkter vid en invers till noll-operation.

2.2 Skarphet i en bild

För att avgöra hur skarp en bild är kan Laplaceoperatorn och Tenengrad-algoritmen användas.

2.2.1 Laplaceoperatorn

För ett skalärt fält, vilket en bild analyseras som i en dator, defineras Laplaceoperatorn som divergensen av gradienten av fältet[10]. Detta innebär att ett nytt fält erhålls som består av summan av andra ordningens partiella derivator. Om det ursprungliga fältet bestod av värden på intensitet, är det nya fältet ett mått på hur intensiteten förändras i olika delar av bilden. Beroende på hur datan över intensitetsförändringarna varierar erhålls ett mått på huruvida den ursprungliga bilden är skarp eller ej. Sker små intensitetsförändringar anses det som en utsmetad bild, medan stora förändringar tyder på en skarp bild[11].

För att beräkna skärpan i bilden studeras variansen hos fältet med den utförda Laplaceoperationen. Variansen, vilket är detsamma som standardavvikelsen i kvadrat, ger ett mått på hur avvikande delar av bilden är från medelvärdet. Stora avvikelser tyder som tidigare nämnt på en skarp bild.

2.2.2 Tenengrad-algoritmen

För att få en riktlinje på hur skarp en bild är kan Tenengrad-algoritmen[12] användas. Denna är baserad på gradienten av bilden, vilket innebär att endast första ordningens derivata studeras. Denna derivata räknas ut med en lättberäknad operator vid namn Sobel[13]. Denna metod använder den matematiska operationen faltning på matrisen tillsammans med bilden vilket ger en approximation till derivatan i en rikting. Detta utförs i x- och y-riktning och slås sedan ihop för att ge ett värde på gradienten.

Kalla den ursprungliga bildens intensitet i punkten (x, y) för $I(x, y)$. Efter utförd Sobel-operation och sammanslagning fås gradientens amplitud $S(x, y)$ och är uttryckt som

$$S(x, y) = \sqrt{(i_x * I(x, y))^2 + (i_y * I(x, y))^2}$$

där faltningsmatriserna i_x och i_y uttrycks som:

$$i_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad i_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Denna amplitud är ett värde på hur intensiteten varierar i punkten (x, y). För att ta reda på hur intensiteten varierar över ett område och med detta ge ett värde på hur skarp bilden är tas medelvärdet av S. Ett stort värde på S tyder på en skarp bild.

2.3 HDR

High dynamic range (HDR), på svenska stort ljusomfång, är en teknik som används för att fånga ett större område av luminans i bilder än vad som normalt är möjligt. Då en kamera ej kan fånga hela ljusomfånget tas flera bilder med olika exponeringsvärden. På dessa bilder tillämpas flera algoritmer som sedan sätts ihop till en bild som innehåller ett större område av luminans. Detta gör att bilden är mer lik det som det mänskliga ögat ser än en bild med lägre ljusomfång.

2.3.1 Exponeringsvärden

Exponeringsvärde (EV)[14] används för att sätta exponeringstid och bländartal på kameror och på så vis få en bild med rätt exponering. För en HDR-bild används dessa parametrar för att relativt det optimala exponeringsvärdet få exponeringsvärden som gör att bilden blir under- eller överexponerad. Dessa parametrar kan sättas manuellt och räknas ut enligt formeln

$$EV = log_2 \frac{N^2}{t}$$

där EV är exponeringsvärdet, N är bländartal och t är exponeringstid.

Detta görs i specifika steg som sätter exponeringsvärden till en skala av 2-potenser. I varje steg för överexponering dubbleras mängden ljus och i varje steg för underexponering halveras ljuset. Många kameror har ett läge där parametrarna sätts automatiskt och dessa lämpar sig för de flesta HDR-bilder.

2.3.2 Kamerans svarsfunktion

En bild tagen med en kamera innehåller intensitetsvärden relaterade till ljusstyrkan i motivet. Detta görs med hjälp av kamerans svarsfunktion (från engelskans *Camera Response Function*, CRF)[15]. Om denna funktion är okänd kan den bestämmas genom att korrelera intensitetsvärden mellan bilder med olika exponeringsvärden. När funktionen har bestämts kan den sedan användas för att räkna ut exponeringsvärdet för olika punkter i en bild.

2.3.3 Färgöversättning för skärm

En bild kan ofta innehålla ett större ljusomfång än vad de flesta skärmar kan visa. För att lösa detta används en metod som kallas *tone mapping*[16]. Den mängd ljus och färger som bilden innehåller översätts till en mängd som skärmen kan visa. På så sätt kan ett högre ljusomfång simuleras. Tone mapping används ofta i samband med HDR-algoritmer. Efter att HDR-bilden har räknats ut appliceras algoritmen på bilden för att kunna visa ett större ljusomfång på en skärm.

2.3.4 Debevecs algoritm

Debevecs algoritm för HDR[17] fungerar enligt följande:
Exponeringen X är en produkt av irradiansen[18] E, vilket är effekt per yta, och exponeringstiden Δt:

$$X = E\Delta t$$

Efter det att en bild är tagen får varje bildpunkt ett värde Z som är uträknat från exponeringen X med en funktion:

$$f(X) = Z$$

Funktionen f kan därför ge X genom $X = f^{-1}(Z)$ om f antas vara monoton. Antalet bilder som tagits och exponeringstiden för dessa Δt_j är känt. Om motivet antas vara statiskt och bilderna tas i tillräckligt snabb följd för att inte irradiansen E_i för varje bildpunkt ändras kan den antas vara samma för samma bildpunkt i de olika bilderna. Detta ger ekvationen

$$Z_{ij} = f(E_i\Delta t_j)$$

där ij är bildpunkt i för bild j. Anta att f är monoton, ta inversen och sedan naturliga logaritmen på båda sidor och så kan ekvationen skrivas som

$$g(Z_{ij}) = lnE_i + ln\Delta t_j$$

där Z och Δt är kända. För att få ut g löses ekvationen som en linjär minsta-kvadrat-metod. När funktionen g är känd kan E räknas ut för de olika bilderna. Därefter kan en HDR-bild färdigställas med hjälp av E.

2.3.5 Robertsons algoritm

Robertson et al. har föreslagit följande metod för HDR[19].

Antag att N bilder har tagits i en statisk miljö. Exponeringstiden betecknas t_i där $i = 1, \ldots, N$ och bildpunkt j i bild i betecknas y_{ij}. Målet med algoritmen är att bestämma irradiansen x_{ij}. Endast exponeringstiden varierar och mängden ljus som bidragit till utvärdet y_{ij} blir då $t_i x_j$. Termen N_{ij}^c inkluderar mängden brus i bilden. Kvantiteten $t_i x_j + N_{ij}^c$ ger funktionen g.

$$y_{ij} = g(t_i x_i + N_{ij}^c)$$

Då y_{ij} representerar 8-bitars data översätter g positiva reella tal till ett heltal O som har ett värde i intervallet $0, ..., 255$. Detta ger:

$$g(z) = \begin{cases} 0 & \text{if} \in [0, I_0] \\ m & \text{if} \in [1_m - 1, I_m], m = 1, \ldots, 254 \\ 255 & \text{if} \in [I_{254}, \infty] \end{cases}$$

O kan därför översättas till de reella talen genom

$$g^{-1}(y_{ij}) = t_i x_j + N_{ij}^c = I_{y_{ij}}$$

Då N-termen är oerhört svår att uppskatta ersätts den sedan med Gaussfunktionsvariabler med variansen σ_{ij}^2. De här variablerna ersätts sedan med viktade $w_{ij} = \frac{1}{\sigma_{ij}^2}$. Då g ej är känd behöver värdena $I_m, 0, \ldots, m$ och även x_j bestämmas. Nu kan en optimeringsfunktion tas ut

$$\tilde{\mathrm{O}}(I, x) = \sum_{ij} w_{ij} (I_{y_{ij} - t_i x_j})^2$$

Med hjälp av Gauss-Seidels metod[20] minimeras funktionen först med avseende på I_m och sedan med avseende på x_j Då skalan av x_j beror på skalan av I_m restrikteras I_m så att $I_{128} = 1, 0$. Detta motsvarar en iteration av algoritmen. Därefter itereras algoritmen tills konvergenskriteriet att optimeringsfunktionens lutning går under ett visst minsta värde. g används sedan för att hitta exponeringen i bilderna och en HDR-bild kan sättas ihop.

2.3.6 Mertens exponeringsfusion

Exponeringsfusion[21] (från engelskans *exposure fusion*) är en teknik som skiljer sig från vanliga HDR-algoritmer på så vis att den hoppar över steget att beräkna kamerans svarsfunktion och tittar direkt på bilder med olika exponeringsvärden. Detta gör den genom att använda sig av kvalitetskontroller för mättnad, kontrast samt exponering.

- **Mättnad:** En bild som har överexponerats kan ofta tappa färginnehåll. För att hitta de bildpunkter med högst mättnad räknas standardavvikelsen ut i de olika RGB-kanalerna vid varje bildpunkt.
- **Kontrast:** Ett laplacefilter appliceras på en gråskala av varje bild och där tas absolutvärdet ut från filtret och ger högt värde åt bildpunkter vid bland annat kanter där differensen är stor.
- **Exponering:** Intensiteten i en bildpunkt antar ett värde mellan 0 och 1. Ju närmare 0 den är desto mer underexponerad är den och ju närmare 1 den är desto mer överexponerad är det. Därmed kan de bildpunkter som har rätt exponering väljas. Bildpunkterna viktas med hjälp av en Gaussfunktion[22], som illustreras i Figur 2.2:

  $$exp(-\frac{(i - 0,5)^2}{2\sigma^2})$$

  där σ är satt till 0,2 och i är intensiteten i bildpunkten. Gaussfunktionen appliceras separat på de olika RGB-kanalerna, multipliceras sedan och viktas baserat på hur nära 0,5 resultatet är.

Utifrån dessa kvalitetskontroller viktas sedan varje enskild bildpunkt och jämförs med samma bildpunkt i de olika bilderna och en ny bild kan sättas ihop som innehåller ett större ljusomfång.

2.4 Rektangulär polygonvalidering

En polygon som har en rektangulär form består alltid av fyra punkter $H = \{p_1, ..., p_4\}$, vilka är rektangelns hörn. Hörnvinklarna för en rektangel är alltid exakt 90°. Vinkeln för ett hörn, v, som bildas av punkterna p_1, p_2, p_3, kan hittas med hjälp av cosinussatsen:

$$v(p_1, p_2, p_3) = \arccos\left(\frac{len(p_1, p_2)^2 + len(p_1, p_3)^2 - len(p_2, p_3)^2}{2 \cdot len(p_1, p_2) \cdot len(p_1, p_3)}\right)$$

där avståndet mellan två punkter (x_1, y_1) och (x_2, y_2) fås av

$$len((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Låt H vara den samling punkter som bygger polygonens samtliga hörn, det vill säga att $H = \{\{p_1, p_4, p_2\}, \{p_2, p_1, p_3\}, \ldots\}$. För en rektangel hade som tidigare nämnt kriteriet för att identifiera en rektangel varit $\forall h \in H : v(h) = 90$. I verkligheten är vinklarna sällan räta, vilket är varför en felmarginal δ behöver introduceras. Detta ger det slutgiltiga kriteriet: $\forall h \in H : v(h) = 90 \pm \delta$. Om en polygon uppfyller kriteriet anses den att ha en rektangulär form.

2.5 Polygonreducering

Då det är ett fyrkantigt kvitto som polygonen ska omsluta kan polygonen förenklas till fyra hörn. Detta är då kvittot antas vara av god kvalitet, saknar krökta hörn och har en rektangulär form.

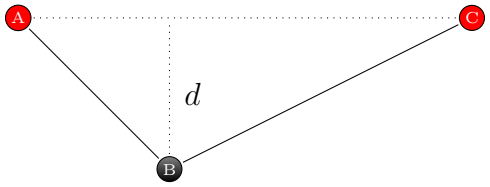
2.5.1 Ramer-Douglas-Peuckers algoritm

Ramer-Douglas-Peuckers algoritm (RDP) är en iterativ metod som används för att hitta en approximation till en kurva bestående av många punkter. Metoden bygger sedan på att de mest signifikanta punkterna i figuren ska hittas. Detta görs genom att kolla på mittpunkten i varje trio av närliggande punkter likt Figur 2.9 nedan. För att metoden ska fungera måste ett gränsvärde ϵ bestämmas. Detta ϵ väljs som en funktion av båglängden på den ursprungliga polygon som ska approximeras, detta för att säkerhetsställa att metoden fungerar oavsett upplösning på bilden. Om avståndet från mittpunkten till den linjära approximationen mellan start- och slutpunkten är större än ϵ anses punkten vara viktig och kommer att behållas[23].

$$\epsilon = c \cdot \text{båglängd(polygon)}$$

Konstanten c bestäms beroende på hur strikt approximationen får vara. Om approximationen får fler än fyra punkter ökas denna konstant med en procentenhet,

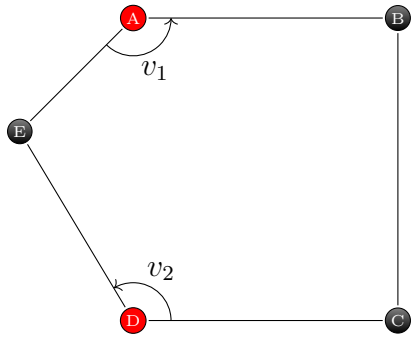
varefter algoritmen körs på nytt. Skulle antalet punkter istället vara färre än fyra, sänks konstanten med en procentenhet.



Figur 2.9: Om punkten B avviker från den linjära approximationen mellan A och C med ett avstånd d större än ϵ kommer denna punkt att behållas.

2.5.2 Vinkelbaserad polygonreducering

Vinkelbaserad polygonreducering kan appliceras på en polygon med mellan fem och nio punkter som ska reduceras till en rektangulär form. Efter reduceringen ska de kvarvarande fyra punkterna ha en vinkel så nära 90° som möjligt. Genom att gå över varje punkt i polygonen kan punktens vinkel tas fram genom att använda cosinussatsen med den aktuella, föregående och nästkommande punkten. Om denna vinkeln är för stor eller för liten är det den första problempunkten, p_1. Låt γ vara hur mycket vinkeln får avvika från 90°. Ekvationen för att hitta problempunkten blir då $|v - 90| \geq \gamma$. Gå sedan baklänges över punkterna och hitta nästa problempunkt, p_2. I Figur 2.10 visas hur punkterna A och D identifieras som problempunkter. I följande exempel gäller därför att $p_1 = A$ och $p_2 = D$.

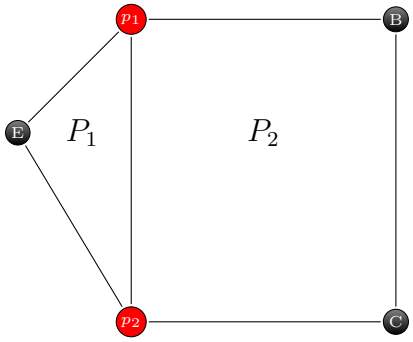


Figur 2.10: Punkterna A och D identifierade som problempunkter då deras vinklar v_1 och v_2 är för vida.

Om polygonen består av punkterna $[A, \ldots, E]$ och de identifierade problempunkterna är p_1 och p_2 kan polygonen delas upp i två delpolygoner. Den första delpolygonen går ett varv från den första problempunkten vidare genom övriga punkter till den andra problempunkten. Dessa problempunkter binds sedan samman. Den andra delpolygonen består av de övriga punkterna mellan problempunkterna enligt:

- $P_1 = ([A, \ldots, E] \setminus [p_1, \ldots, p_2]) \cup \{p_1, p_2\}$
- $P_2 = [p_1, \ldots, p_2]$

I exemplet från Figur 2.10 fås då $P_1 = \{p_1, p_2, E\}$ och $P_2 = \{p_1, B, C, p_2\}$. Denna uppdelning illustreras i Figur 2.11.



Figur 2.11: Polygonen uppdelad i två delpolygoner utefter problempunkterna.

Den delpolygon av P_1 och P_2 som har fyra hörn och har störst area är den sökta polygonen, P_r. I Figur 2.11 gäller därför $P_r = P_1$. Alla olika inkluderingspermutationer av P_r och problempunkterna behöver valideras och betraktas. Den hittade polygonen delas därför upp i följande fall för att inte missa den korrekta kombinationen:

- $P_{r1} = P_r$
- $P_{r2} = P_r \setminus \{p_1, p_2\}$
- $P_{r3} = P_r \setminus \{p_1\}$
- $P_{r4} = P_r \setminus \{p_2\}$

Alla permutationer ovan valideras enligt tidigare nämnd polygonvalidering. Om en permutation klarar valideringen har polygonen framgångsrikt förenklats. I exemplet ovan blir slutresultatet P_{r1}, som har den sökta rektangulära formen.

2.6 Projektiv avbildning

I en projektiv avbildning transformeras perspektivet. Detta görs med matrisen H som omvandlar källans koordinater x_i och y_i till de koordinater som önskas vara placerade vid x_i' och y_i'. Då ingen förändring sker i z-led sätts dess koordinat till 1, både för källan och den projicerade bilden. Om det är önskvärt att ändra bildens storlek i samband med omvandlingen behöver även en skalär s_i införas som skalningskonstant. Med andra ord är H uttryckt som

$$s_i \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = H \cdot \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

Vidare minimeras felet vid återprojektion genom att välja komponenterna i H så att summan

$$\sum_i \left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2$$

minimeras.

2.7 Egenskapsigenkänning

Egenskapsigenkänning (från engelskans *feature matching*), är en metod som bygger på att hitta egenskaper som är typiska för ett område i en bild. Dessa egenskaper söks sedan i andra bilder och vid igenkänning kan dessa bilder antas vara på samma objekt och därför vävas samman[24].

För att avgöra om ett område W är bra för identifiering ska en liten förflyttning av det observerade området ge stor förändring av data. Kalla datan i punkten (x, y) för $I(x, y)$ och förflyttning i x- och y-led för u respektive v. Skillnaden av förflyttningen (u, v) mäts då som summan av skillnaderna i kvadrat och kallas $E(u, v)$. Med andra ord är det uttryckt som

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

Antag att förflyttningen (u, v) är liten. Då kan $I(x + u, y + v)$ approximeras till $I(x, y)+\frac{\partial I}{\partial x}u+\frac{\partial I}{\partial y}v$ med en taylorutveckling av första ordningen. Med detta antagande förenklas $E(u, v)$ till $\sum_{(x,y)\in W}(\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v)^2$, vilket på matrisform innebär att

$$E(u,v) = \begin{pmatrix} u & v \end{pmatrix} \left(\sum_{(x,y)\in W} \begin{pmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{pmatrix} \right) \begin{pmatrix} u \\ v \end{pmatrix}$$

Slutligen kan egenvärdena för

$$\sum_{(x,y)\in W} \begin{pmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{pmatrix}$$

tas fram. Då matrisen är en 2×2-matris kommer det att finnas två egenvärden, λ_1 och λ_2. Vidare kan dessa ge vilken riktning där E ökar respektive minskar snabbast. Om $\lambda_1 \gg \lambda_2$ eller $\lambda_2 \gg \lambda_1$ anses det observerade området W befinna sig över en kant. Däremot om både λ_1 och λ_2 är stora är det ett hörn som är observerat och om de är små har ingen kantstruktur funnits.

Stora värden på båda egenvärdena innebär att E ökar märkbart oavsett vilken rikting förflyttningen sker i. Detta är önskvärt då det underlättar vid igenkänning av det specifika området. Slutligen innebär detta att det är hörnpunkter som lämpar sig väl för identifiering[24].

2.8 Förminskning av bild

Interpolation används för att konstruera nya datapunkter utifrån redan existerande. Det lämpar sig därför att använda metoden vid storleksförändring av en bild för att återge den skalade bilden väl. I programvaran OpenCV finns det färdigutvecklade algoritmer för denna typ av bildbehandling. Dessa fungerar dock olika väl beroende på om bilden ska förminskas eller förstoras[25].

En färdig algoritm som kan vara önskvärd vid nedskalning är *inter area*. Som namnet antyder är metoden baserad på hur stor area samma bildpunksvärde befinner sig inom. Interpolationen påminner om närmaste-granne-interpolation (från

engelskans *nearest-neighbor interpolation*) vilket innebär att det område som skapas genom att ta en datapunkt och gå halvägs till närmaste ny datapunkt i alla riktningar har samma värde. Detta innebär att det kommer befinna sig skarpa kanter i bilden.

2.9 Levenshteinavstånd för textjämförelse

Levenshteinavståndet beskriver hur lika två texter är genom att ange hur många ändringar som krävs för att göra de två texterna identiska. Ändringarna som kan göras är antingen *tillägg*, *modifikation* eller *borttagning*[26].

Med *tillägg* menas det scenario då en text kan göras till en annan genom att lägga till tecken. I fallet då texten *ab* ska omvandlas till *abc* blir Levenshteinavståndet ett, ty texterna blir identiska om tecknet *c* läggs till sist i den första texten.

En *modifikation* syftar istället på att ett existerande tecken ändras. Exempelvis blir texterna *abd* och *abc* lika om tecknet *d* i den första texten modifieras till ett *c*. Då endast en förändring har skett blir Levenshteinavståndet även i detta fall ett.

Slutligen studeras fallet för *borttagning* mellan texterna *abcd* och *abc*. Dessa görs identiska genom att tecknet *d* tas bort från den första texten, vilket ger Levenshteinavståndet ett.

Ett lågt Levenshteinavstånd innebär alltså att två texter är lika varandra, medan ett högt avstånd beskriver motsatsen. Värt att notera är att Levenshteinavståndet inte noterar vilken typ av ändring som behöver göras, utan endast ger ett värde på antalet modifieringar som krävs. Detta värde är heller inte beroende av hur lång texten är.

3
Metod

Som beskrevs i avsnitt 1.2 valdes tidigt några potentiella förbättringsmetoder ut. Hur dessa förbättringsmetoder utvecklats och använts vid inskanning av kvitton beskrivs ingående i detta kapitel. Dessutom behandlar kapitlet hur resultatet av de olika metoderna testas.

3.1 Skapa en superbild genom videobehandling

En egenutvecklad förbättringsmetod är att filma ett kvitto för att sedan slå ihop de bästa delarna av videons bildrutor till en så kallad superbild. Nedan förklaras hur förbättringsmetodens olika steg går till mer detaljerat.

3.1.1 Dela upp video i bildrutor

Metodens första steg är att dela upp en video av ett kvitto i bildrutor. Målet med uppdelningen är att sprida ut reflektioner över olika positioner i olika bildrutor. Detta sker naturligt om kameran flyttas över kvittot under filmningen. Varje bildruta tar dock en avsevärd tid att behandla, vilket är anledningen till att mängden utvalda bildrutor behöver minskas. Ett enkelt sätt att minska antalet bildrutor är att från början godtyckligt bestämma hur många bildrutor som söks. Därefter väljs bildrutor ut uniformt från videon, det vill säga att samma antal bildrutor bortses från mellan varje vald bildruta. Om behandlingstiden är irrelevant fås det bästa resultatet genom att behandla alla bildrutor.

3.1.2 Förminskning av bild

Tiden det tar att behandla bildrutorna beror på deras storlek. För att minska behandlingstiden görs en förminskning av bildrutorna med hjälp av metoden *inter area*. En förminskningsfaktor beräknas så att den största dimensionen på bilden är 1000 bildpunkter. Denna förminskningsfaktor används även för den andra dimensionen på bilden, vilket gör att förhållandet mellan bredd och höjd oförändrat. Skulle upplösningen på videon vara så låg att den största dimensionen är mindre än 1000 bildpunkter görs ingen storleksförändring.

3.1.3 Utvinna kvitton från bilder

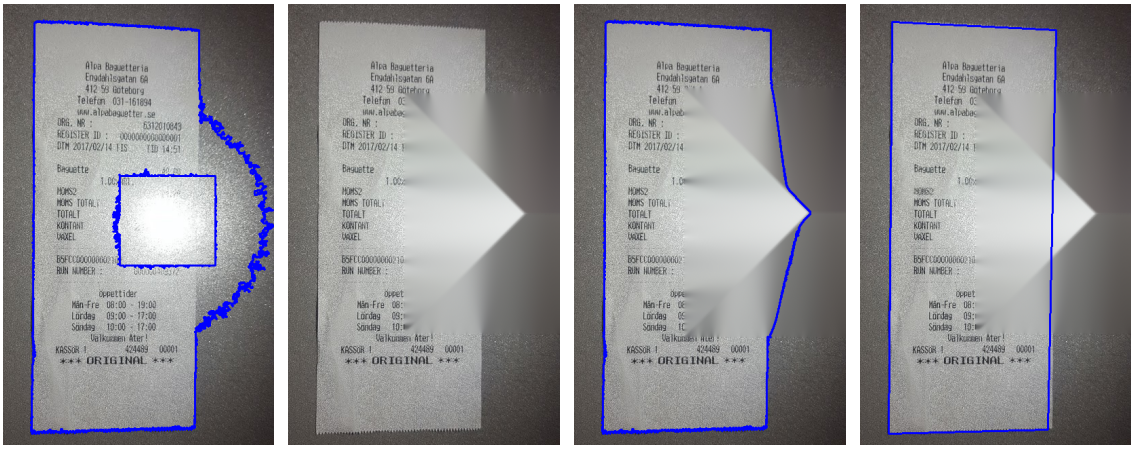
Då kameran förflyttas vid inspelning av en video kan det antas att perspektivet av kvittot kommer att förändras mellan olika bildrutor. För att underlätta vid ihop-

sättning av videon till en bild kommer denna perspektivförändring att behöva kompenseras. Detta utförs genom att i varje vald bildruta från videon hittas kanterna på kvittot. Därefter kan en projektiv avbildning transformera bilden av kvittot till att ha samma perspektiv mellan de olika bildrutorna.

Vid identifiering samt framtagning av ett kvitto ur en bild minskas först effekten av blixtreflektioner. Detta på grund av att en blixtreflektion gör det betydligt svårare att extrahera kvittot från en bild. Dessutom kan reflektionen vara destruktiv för vidare behandling. Därför är det viktigt att denna upptäcks och vid behov behandlas tidigt i framtagningen av kvittot. Ett exempel på en reflektion som försvårar kvittodetektion syns i Figur 3.1a.

För att identifiera var en eventuell blixtreflektion är placerad genomgår en kopia av bilden en binär avbildning med ett strikt tröskelvärde. Från denna ansätts en polygon som omsluter blixtreflektionen. Vidare finns det en algoritm författad av Alexandru Telea som smetar ut och raderar en del av en bild på ett sådant vis att den imiterar det kringliggande området[27]. Med hjälp av den algoritmen kan reflektionen smetas ut för att minska effekten den har på kommande steg. Användning av algoritmen illustreras i Figur 3.1b.

Efter att reflektionen har behandlats upprepas steget med att ta en binär avbildning. Däremot är tröskelvärdet i detta fall inte lika snävt utan istället anpassat för att utvinna kvittot. Anledningen till att en binär avbildning utförs är att OpenCV har färdiga algoritmer som kan hitta konturer från en binär bild[28]. Däremot resulterar algoritmerna ofta i flera olika konturer och motsvarande polygoner, där det antas att kvittot är polygonen med störst area. Kopian av ursprungsbilden kan nu bortses från då den potentiellt fått en informationsförlust under behandlingen. Den hittade polygonen gäller dock för ursprungsbilden som används i de kommande stegen. Detektion av kvitto efter reflektionsbehandling visas i Figur 3.1c.



(a) Deformation **(b)** Utsmetning **(c)** Konturdetektion **(d)** Reducering

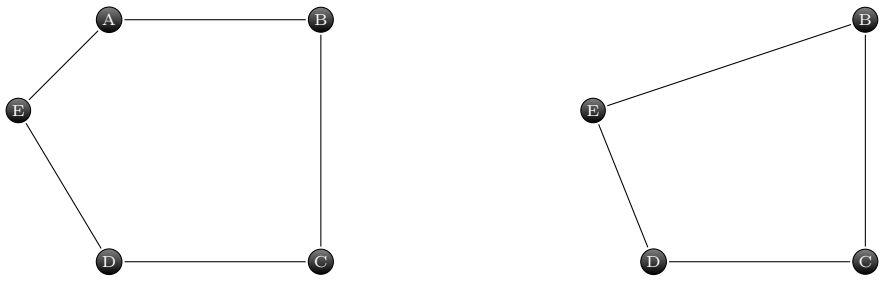
Figur 3.1: I (a) har en blixtreflektion deformerat den omslutande polygonen. Reflektionen smetas ut med hjälp av Alexandru Teleas algoritm i (b) och en ny omslutande polygon hittas med hjälp av binär avbildning och konturdetektion i (c). Slutligen i (d) kan den omslutande polygonen förenklas för att ta bort den avvikande delen från blixtreflektionen.

Den valda polygon som motsvarar kvittot består ofta av tusentals punkter. För att

arbeta vidare med polygonen behöver den förenklas till att bestå av fyra omslutande hörnpunkter. Till en början appliceras RDP-reducering upp till 20 gånger med olika tröskelvärden. På bilder utan blixtreflektioner är detta ofta nog för att få en bra reduktion till 4 omslutande punkter. Vissa mer komplicerade polygoner behöver däremot genomgå kompletterande vinkelbaserad reducering efter processen med den första reduceringen. Se Figur 3.1d för en lyckad polygonreducering.

RDP-reducering är dock inte alltid passande för rektangulära polygoner med kraftigt utstickande punkter, vilket exempelvis kan uppstå vid en blixtreflektion. Under sådana förhållanden kan en sådan reducering vara direkt destruktivt för framtida behandling, se Figur 3.2 nedan. För att undvika att den första reduceringen ska vara destruktiv förändras gränsvärdet med små steg.

Om reduceringen misslyckas, det vill säga att den inte ger fyra hörn eller om polygonvalideringen ger negativt utslag, kasseras bildrutan. Det sista steget för att utvinna kvittot är att applicera en projektiv transformering med avseende på de fyra framtagna hörnpunkterna. Detta utförs i syfte att få ett så likt perspektiv som möjligt mellan bildrutorna.

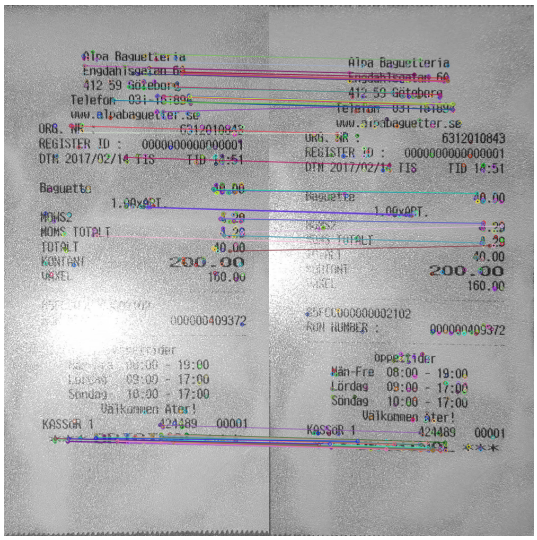


Figur 3.2: Polygonen till vänster är orginalpolygonen och polygonen till höger är efter Ramer-Douglas-Peucker-reducering. Den kritiska punkten A har tagits bort istället för den önskade E.

3.1.4 Linjera utvunna kvitton

Kantdetekteringen som används har en viss felmarginal, vilket gör att innehållet i varje bildruta sannolikt kommer vara förskjutet några bildpunkter jämfört med varandra. När bildrutan sedan genomgår en projektiv avbildning introduceras ytterligare en risk att innehållet även blir något utdraget. Detta kan utgöra ett problem när bildrutorna senare ska klippas ihop, oavsett hur smart klippningen sker. Därför behöver bildrutorna linjeras mot en utvald referensbild.

Detta görs genom att godtyckligt välja ut en referensbild som följande bildrutor kommer att linjeras mot. Med hjälp av OpenCVs inbyggda stöd för egenskapsigenkänning[29] hittas nyckelpunkter för varje bildruta. Därefter paras nyckelpunkterna ihop mellan referensbilden och varje bildruta. Varje bildruta innehåller ofta en stor mängd nyckelpunkter, men för att få ett bra slutresultat är det viktigt att få med parade nyckelpunkter i varje del av bilden, annars finns det en risk att den uteblivna delen förvrängs i nästa steg.



Figur 3.3: Två bildrutor där detekterade nyckelpunkter parats ihop mot varandra och visualiseras med linjer mellan.

Nu finns det en mängd nyckelpunkter för varje bildruta som behöver flyttas till motsvarande punkts position i referensbilden. Det sker genom att göra en projektiv avbildning, vilket resulterar i att bildrutorna blir linjerade.

3.1.5 Klippa ihop bildrutor

En bildruta kan delas upp i sektioner som består av rader och/eller kolumner. När kolumner används tenderar texten på den ihopslagna bilden att bli dålig, främst då text har ett naturligt horisontellt format. Därför delas bilder upp i rader.

Genom att uniformt dela upp bilder i ett bestämt antal rader är risken stor att en textrad blir avklippt. Därför behöver sektionerna väljas ut med stor noggrannhet på så vis att ingen text bryts av. Från en video väljs godtyckligt en referensbild ut. Bilden transformeras med hjälp av en binär avbildning för att lättare skilja på text och bakgrund. För att en sekvens av rader ska anses vara en giltig sektion ska den vara omringad av minst α stycken mellanrumsrader. En mellanrumsrad är en rad av bildpunkter där andelen bildpunkter som klassas som vita är minst β. En bildpunkt klassas som vit om dess snittvärde av RGB-kanalerna är minst γ. Parametrarna α, β och γ behöver anpassas efter referensbilden och dess egenskaper. Parametern α beror på upplösningen av bilden, medan β är beroende på bildens innehåll och γ varierar med avseende på vilket ljus bilden är tagen i. I det allmänna fallet där bilden har mycket bra kontrast är däremot följande en bra utgångspunkt:

- $\alpha = 5$
- $\beta = 0.98$
- $\gamma = 250$

Algoritmen ovan ger en mängd sektioner som är garanterade att inte vara destruktiva när dessa slås samman till en superbild. Därefter beräknas graden av skarphet för varje sektion och bildruta, och de skarpaste sektionerna läggs i en lista. Med OpenCV kan sedan de isärplockade sektionerna läggas ihop till en stor superbild, som då består av de skarpaste delarna från alla bilder.

3.2 HDR

För att skapa en HDR-bild tas flera bilder i snabb följd. Bilderna tas med olika exponeringsvärden för att fånga ett större ljusomfång än vad endast en bild skulle kunna fånga. Tillsammans med mobilkamerans HDR utvärderades tre befintliga algoritmer; Debevec, Robertson och Mertens fusion. Robertson och Debevec är två algoritmer som använder sig av exponeringstider för att uppskatta kamerans responstid. I dessa algoritmer görs sedan en tone mapping för att kunna visa bilden på en skärm. Mertens slår istället ihop bilden direkt med hjälp av en referensbildpunkt utan hänsyn till exponeringstider.

3.3 Bildförbättring

Utjämnande filter och tröskeloperationer testas och utvärderas fristående från videolösningen och HDR.

3.3.1 Utjämnande filter

Vid applicering av de utjämnande filter som undersöks finns ett antal parametrar som ska bestämmas. Ett exempel på en sådan är storleken på det område som ska användas för att bestämma ett nytt bildpunktsvärde. Vilka värden på parametrarna som ger optimala resultat varierar beroende på den bild som filtret appliceras på. Dessa värden går inte att veta på förhand utan prövas fram vid varje testfall.

3.3.2 Tröskeloperationer

Även tröskeloperationer kräver att ett antal parametrar, vars optimala värden är beroende av den bild som ska behandlas, bestäms. Den mest centrala parametern är tröskelvärdet som både kan bestämmas manuellt eller med Otsus metod. Vid varje testfall prövas ett optimalt tröskelvärde fram manuellt. Resultatet som fås med detta tröskelvärde jämförs med resultatet som fås då Otsus metod används för bestämning av tröskelvärdet.

3.4 Testning av förbättringsmetoder

För att undersöka hur väl förbättringsmetoderna som framställs fungerar behöver de testas. I detta avsnitt beskrivs den miljö och det material som används vid framtagande av testdata samt hur testningen går till.

3.4.1 Testmiljö och material

Testdata har framtagits i en miljö utan möjlighet att på ett exakt vis styra ljusstyrkan. Då projektets syfte är att göra en undersökning i mörka förhållanden valdes ett mörkt rum med viss ljusinströmning för framtagande av testdata. Ljusförhållandet var konstant mellan framtagandet av olika bilder och videor. Även avståndet mellan

kvitto och mobilkamera var konstant. Mobiltelefonen som använts för samtliga bilder under rapporten är en Samsung Galaxy S6. För att förstärka reflektionen från kamerablixten placerades kvittot i en plastficka.

3.4.2 Jämföra texter

I syfte att testa hur väl en förbättringsmetod fungerar används Levenshteinavståndet för att jämföra texten som läses av från en producerad bild med texten som faktiskt finns på kvittot. Med hjälp av Levenshteinavståndet räknas ett likhetsvärde mellan texterna ut. Detta görs då resultatet är beroende av kvittotextens längd. Antag som exempel ett scenario där två olika kvitton med olika långa texter skannas. Om båda de skannade texterna har samma Levenshteinavstånd till sina respektive kvittons orginaltexter, ska den inskanning som är gjord på kvittot med den längsta texten anses vara bäst. Likhetsvärdet beräknas enligt formeln

$$likhet = \frac{max(l(a), l(b)) - L(a, b)}{max(l(a), l(b))}$$

där a och b är kvittots originaltext respektive den inskannade texten, $l(x)$ är längden på texten x, och $L(x, y)$ är Levenshteinavståndet mellan texterna x och y.

Likhetsvärdet blir alltså ett värde mellan 0 och 1 och kan uttryckas i procentform. Om exempelvis ett kvitto som har en text med 100 tecken skannas in, den inskannade texten också har 100 tecken och Levenshteinavståndet mellan texterna är 10, blir likhetsvärdet:

$$likhet = \frac{100 - 10}{100} = 0.9 = 90\%$$

4

Resultat

I detta kapitel presenteras resultat för de olika förbättringsmetoder som undersökts. För att sätta de olika metodernas effektivitet i perspektiv jämförs de med varandra.

4.1 Videobehandling

Från en given video produceras en superbild som är bättre än varje enskild bildruta, vilket kräver en komplicerad algoritm. Nedan listas olika problem som ingår i videobehandlingen och deras eventuella lösningar, tillsammans med intressanta resultat.

4.1.1 Destruktiva perspektivförändringar

Efter att ett kvitto detekterats på en bildruta kommer bildrutan att genomgå perspektivförändringar utifrån den omslutande polygonen. Om denna detekterats felaktigt kommer resultatet att vara oanvändbart, som demonstreras i Figur 4.1. Om sektionerna i den felaktigt förvrängda bildrutan är tydliga kommer de även att inkluderas i den slutgiltiga superbilden. Därför är det kritiskt att omslutande polygoner valideras strikt då superbilden aldrig ska vara sämre än varje enskild bildruta. Det är bättre att vara strikt och bortse från potentiellt acceptabla bildrutor än att inkludera potentiellt felaktiga bildrutor.

Figur 4.1: Bildruta efter en perspektivförändring där det övre högra hörnet detekterades fel.

4.1.2 Kontrastkrav mellan kvitto och underlag

Videometoden kan extrahera kvitton även om kontrasten mellan kvittot och underlaget är låg. I fallet av mycket låg kontrast, exempelvis ett vitt kvitto mot ett vitt underlag, är det inte möjligt att detektera kvittot. Då kommer resultatet från videometoden istället att vara den bildruta med högst skarphetsvärde.

4.1.3 Linjering av utvunna kvitton

När en bildrutas sektioner väljs ut baserat på textradernas position krävs ett antal tomma rader mellan varje sektion. Detta för att textradernas position kan skilja sig några bilderpunkter mellan bildrutorna. Antalet tomma rader som behövs ska minimeras, eftersom ju fler tomma rader som krävs, desto färre sektioner kommer att hittas. Detta är positivt då fler sektioner ger en bättre superbild. På kvitton med bra utformning, textavstånd och perspektiv kan sektioner hittas mellan varje textrad förutsatt att antalet tomma rader som krävs är lågt. Om istället ett högre antal tomma rader krävs kommer sektionerna bli större och endast inkludera stycken eller i värsta fallet hela bilden.

(a) Bildruta 1 som anses linjerad

(b) Bildruta 2 innan linjering

(c) Bildruta 2 efter linjering

Figur 4.2: Linjering av en bildruta mot en annan. I detta fall linjeras bild (b) efter bild (a).

Det syns en knapp skillnad mellan var texten **** ORIGINAL **** är positionerad i Figur 4.2a och Figur 4.2b. Denna skillnad i position skulle kunna anses vara försumbar för blotta ögat, men är destruktiv vid sammanslagningen av superbilden. För att på ett tryggt sätt kunna skapa en superbild med dessa bildrutor krävs ett högre antal tomma rader mellan sektionerna. I Figur 4.2c har bildrutan linjerats efter 4.2a och samma avstånd blir betydligt mindre. Antalet tomma rader som krävs för att skapa en sektion kan därför sänkas och fler sektioner kan potentiellt hittas, vilket ger en bättre superbild. I det fallet att en linjering skulle ge ett perfekt resultat skulle däremot det minsta antalet tomma rader inte kunna sättas till endast en

rad. Detta då vissa bokstäver som exempelvis *Ö* i så fall skulle förlora sina prickar då dessa potentiellt kan uppfattas som brus av algoritmen som beräknar hur vit en bildpunktsrad är.

4.1.4 Bästa bildrutan jämfört med en superbild

Det är viktigt att superbilden från videobehandlingen inte förvärrar avläsningsprecisionen. Superbilden ska därför ha ett skarphetsvärde som är lika högt eller högre som den bästa delrutan. Då superbilden är uppbyggd av de skarpaste sektionerna garanterar det att superbilden aldrig kan vara suddigare än den skarpaste delrutan. I fallet då en bildruta är skarpare i alla sektioner än övriga bildrutor är superbilden helt enkelt identisk med den skarpaste bildrutan.

(a) **(b)**

Figur 4.3: (a) Bildruta med högst skarphetsvärde, med ett uträknat Levenshteinavstånd på 200 som ger ett likhetsvärde på 0.53. (b) Kombinerad superbild, med ett uträknat Levenshteinavstånd på 130 som ger ett likhetsvärde på 0.69. Notera att blixtreflektionen raderades i samband med videobehanlingen.

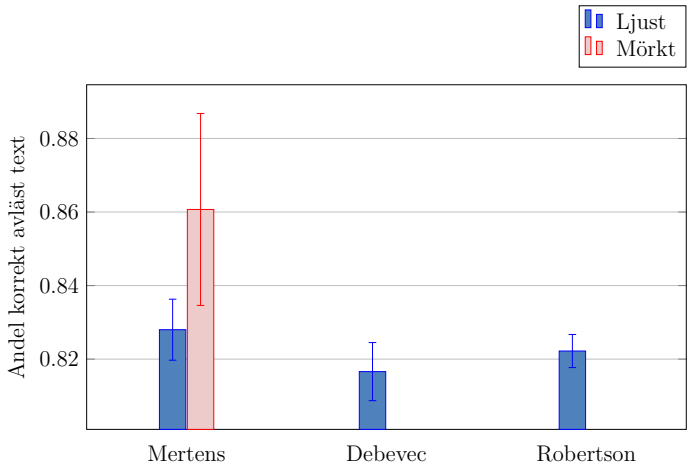
I videon som används i Figur 4.3 har alla bildrutor olika sektioner täckta av en blixtreflektion. Då videobehandlingen väljer ut sektioner utan reflektion får superbilden i detta fall ett lägre Levenshteinavstånd med 35%. Även i Figur 4.4 har superbilden ett lägre Levenshteinavstånd. I detta fall förekommer ingen blixtreflektion. Trots detta sänks Levenshteinavståndet med 39%, vilket är en avsevärd förbättring.

(a) **(b)**

Figur 4.4: (a) Bildruta med högst skarphetsvärde, med ett uträknat Levenshteinavstånd på 109 som ger ett likhetsvärde på 0.75. (b) Kombinerad superbild, med ett uträknat Levenshteinavstånd på 67 som ger ett likhetsvärde på 0.84.

4.2 HDR

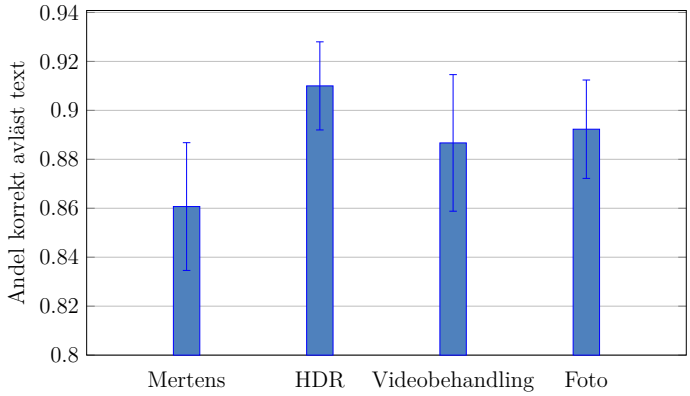
Tre olika algoritmer används för att skapa HDR-bilder med hjälp av OpenCV. I Figur 4.5 presenteras resultaten för de tre algoritmerna i både ljusa och mörka förhållanden. Robertsons och Debevecs får båda färger som inte återger verkligheten, detta påverkar dock inte maskinläsningen då texten ej har påverkats märkbart. När bilderna tas i mörka förhållanden blir resultatet annorlunda. Mertens algoritm ger bättre resultat än för ljusa förhållanden. Däremot blir kvittot nästan helt svart med Debevecs algoritm och nästan helt vitt med Robertsons algoritm vilket gör att ingen text kan avläsas.



Figur 4.5: Genomsnittlig andel av text som blir korrekt avläst inklusive standardavvikelse. De olika HDR-algoritmerna är utförda på foton med bra kontrast mellan kvitto och underlag vid ljusa samt mörka förhållanden. Standardavvikelsen är som synes störst i fallet vid mörka förhållanden.

4.3 Jämförelse mellan videolösning och HDR

I syfte att se hur de olika metoderna presterar mot varandra har det för varje metod utförts tio mätningar i mörka förhållanden. Resultaten presenteras i Figur 4.6. Kamerans inbyggda metod för HDR tagen med blixt ger högst andel avläst text.



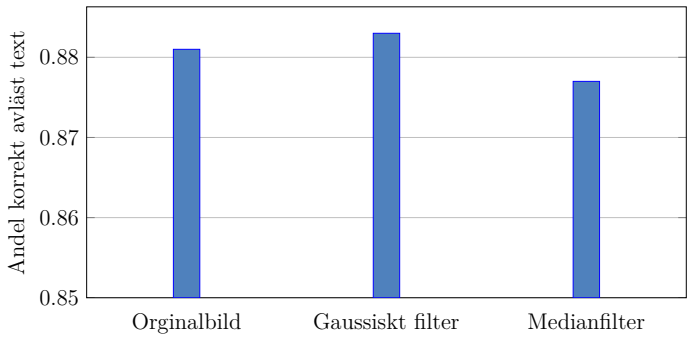
Figur 4.6: Genomsnittet inklusive standardavvikelse av hur stor andel av texten som blir korrekt avläst för olika metoder. Bilderna är tagna med bra kontrast mellan kvitto och underlag samt i mörka förhållanden. Stapeln för HDR betraktar kamerans inbyggda metod tagen med blixt, medan Mertens syftar på den egna implementationen. Även det vanliga fotot har använt blixt.

4.4 Bildförbättring

De metoder för bildförbättring som togs upp i avsnitt 3.3 testas också genom att studera Levenshteinavståndet mellan ett kvittos orginaltext och texten som läses av från en behandlad bild. Dessa resultat jämförs med resultatet för den obehandlade bilden för att undersöka om det faktiskt har skett någon förbättring vid behandlingen.

4.4.1 Utjämnande filter

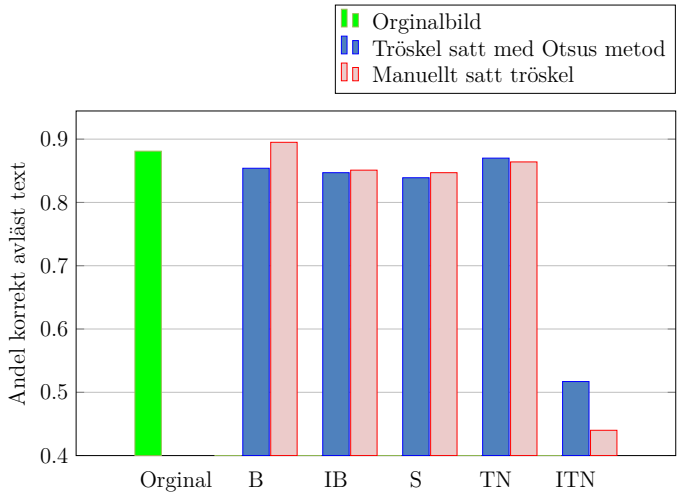
Figur 4.7 visar hur stor andel av texten som avläses korrekt efter användning av utjämnande filter. Det Gaussiska filtret ger en knapp förbättring, medan medianfiltret ger en försämring jämfört med orginalbilden.



Figur 4.7: Resultat för utjämnande filter

4.4.2 Tröskeloperationer

De fem tröskeloperationer som togs upp i avsnitt 2.1.2 har undersökts, både då Otsus metod använts för att bestämma tröskelvärdet och då ett optimalt tröskelvärde prövats fram manuellt. Resultatet av detta visas i Figur 4.8. i figuren syns att en binär avbildning med ett manuellt satt tröskelvärde ger en marginellt större andel korrekt avläst text än orginalbilden. Övriga tröskeloperationer försämrar avläsningen.



Figur 4.8: Hur stor andel av texten som blir korrekt avläst vid användning av tröskeloperationer filter, både med och utan användning av Otsus metod. Tröskeloperationerna är förkortade enligt följande: B = Binär avbildning, IB = Invers binär avbildning, S = Stympning, TN = Till noll-operation, ITN = Invers till noll-operation.

5
Mobilapplikation

I detta kapitel presenteras den mobilapplikation som är utvecklad som ett hjälpmedel för att demonstrera algoritmerna för video- och HDR-behandling. Applikationen har inte någon direkt påverkan på de förbättringsmetoder som undersökts och utvecklats och all information som är relaterad till den har därför samlats i detta kapitel. Algoritmerna som utför videobehandlingen visade sig vara alltför krävande för att utföra på en mobiltelefon och körs därför på en separat server.

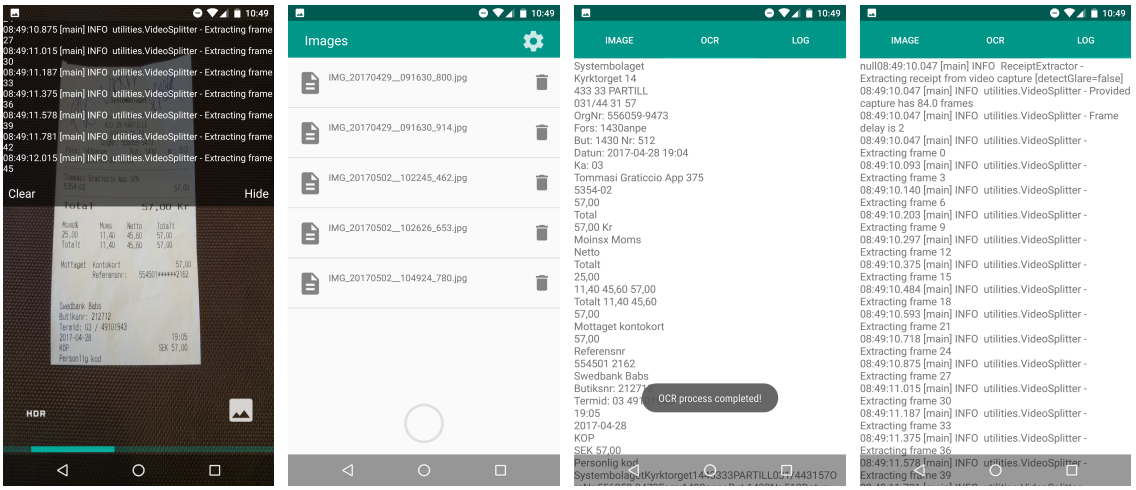
5.1 Klient

Klienten är utvecklad i Java för operativsystemet Android. Med hjälp av Androids bibliotek Camera2[30] kan mobilkamerans funktioner användas. Klienten består av en huvudvy för kameran och en listvy där de behandlade bilderna visas. Dessa vyer syns i Figur 5.1a och 5.1b. I kameravyn filmas kvittot varefter videon skickas till en server för behandling. I klienten går det inte att filma längre än sju sekunder för att förhindra att allt för mycket data behöver behandlas. När servern har behandlat datan och sedan skickat tillbaka den framställda bilden till klienten, sparas denna bild i minnet lokalt på användarens mobiltelefon. Användaren får också en logg tillbaka från servern som innehåller information och meddelanden från själva processen. Där kan alla delmoment som gjorts av förbättringsmetoderna avläsas. När bilden är färdigbehandlad och har sparats till minnet på enheten kan användaren klicka på bilden i listvyn. Då ges möjligheten att läsa av texten på bilden med hjälp av maskinläsning. Vyer för att visa den inlästa texten och för att läsa loggen visas i Figur 5.1c och 5.1d.

5.2 Server

Serverapplikationen är skriven i det Javascript-baserade språket NodeJS[31]. För att kunna använda de algoritmer som utvecklats exporterades projektet till en körbar Java-fil. Argument skickas till huvudklassen för att algoritmerna ska kunna exekveras. Detta argument specificerar sökvägen till videon som användaren vill behandla.

Servern lyssnar på en port och inväntar trafik. Videor och HDR-bilder skickas på liknande sätt men behandlas olika av servern. Användaren spelar in en video av kvittot som sedan skickas till servern i form av en lista med bytes. Servern skriver dessa bytes till en fil lämplig för videoformat. Denna fil används sedan av algoritmerna för att skapa en läsbar bild. Slutligen skickas denna bild tillbaka till användaren för att till sist kunna visas upp i mobilapplikationen.



(a) Klientens huvudvy medan en kvittovideo behandlas

(b) Klientens listvy där användaren kan kolla på sparade och behandlade bilder

(c) Klientens vy som visar den inlästa texten efter att bilden har behandlats

(d) Klientens loggvy där användaren kan kolla på sparade loggar från servern

Figur 5.1: Klientens olika vyer. Dels de som berör behandling av data men även de där data är behandlad.

För att behandla en HDR-bild med de egenimplementerade HDR-algoritmerna tar användaren tre bilder med olika exponeringstid i mobilapplikationen. Dessa tre bilder skickas till servern och skrivs till tre olika bildfiler. Med hjälp av de tre bilderna kan HDR-algoritmerna slå ihop det till en HDR-bild. Denna bild skickas sedan tillbaka till användaren i mobilapplikationen.

En nackdel med att använda en server är att det krävs mycket datakommunikation för att skicka bilder och videor. Dock vägdes fördelen när det gäller bearbetningstid högre och av denna anledning blev det nödvändigt att utföra algoritmerna i en serverapplikation.

6
Analys av metoder

I detta kapitel förklaras och analyseras de metodval som gjorts under projektet. Metoderna som valts motiveras och jämförs med eventuella alternativa metoder.

6.1 Skapa en superbild genom videobehandling

Genom att skapa en superbild utifrån en video kan blixtreflektioner elimineras, vilket även ligger till grund att metoden valdes att studeras. I syfte att uppnå en prestandanivå bekväm för testning, delades videon upp i ett färre antal bildrutor som även förminskades. Vidare val av metoder som berör videobehandling diskuteras i detta avsnitt.

6.1.1 Egenskapsigenkänning

En brist med egenskapsigenkänning är att den endast hanterar stela transformationer (från engelskans *rigid transformation*), och kan således inte identifiera en skalad bild. Exempelvis kan den uppfatta ett uppskalat hörn som en kant. Däremot fungerar den väl då en bild är roterad. För att kompensera för bristerna i metoden skalas de bilder som metoden ska utföras på, så att de ser liknande ut. Denna skalningen sker innan egenskapsigenkänningen utförs.

6.1.2 Förminskning av bild

Det finns ett flertal olika algoritmer för att storleksförändra en bild. En sådan metod är *inter cubic* som kan användas för att ändra storleken på en bild utan att skapa skarpa kanter. Metoden lämpar sig dock främst vid förstoring, vilket inte är eftersträvansvärt i detta fall. Istället önskas endast en förminskning av bilden då målet med storleksförändringen är att öka prestandan.

Ett annat alternativ för storleksförändring är att använda metoden *inter linear*. Denna metod är standardvalet vid interpolation[32] och fungerar väl vid både förstoring och förminskning av bild. Däremot har den jämförts med *inter area*, vilken fungerade bättre i fallet för att skala ned en bild[33]. På grund av detta valdes *inter area* som metod när en bild ska minskas i storlek.

6.1.3 Klippa ihop bildrutor

En superbild bygger på en sammanslagning av de bästa delarna från de studerade bildrutorna. För att foga samman dessa bilder studerades två olika metoder. Den ena

bygger på att sammansätta en mängd delbilder av samma storlek, medan den andra metoden studerar bildens innehåll och delar upp bilden i olika sektioner beroende av detta.

Denna studie behandlar ett objekt med textrader, vilket medför att en uppdelning beroende på sektioner är naturlig. Hade det varit andra objekt som skulle identifieras hade det möjligen inte vara lika optimalt med en sektionsuppdelning. Fördelen i detta fall är att skarvar som kan uppstå vid sammanslagning ej sker i ett tecken, vilket kan hända om sammanslagning sker på storlek av delbilden. Detta syns tydligt i Figur 6.1.

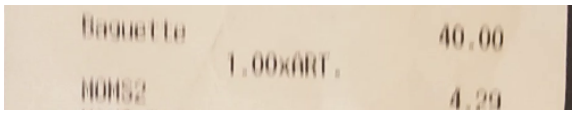
(a) Superbild skapad med uniformt valda sektioner

(b) Superbild skapad med icke textbrytande sektioner

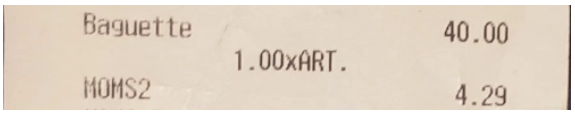
Figur 6.1: Två superbilder med olika metoder för att dela upp bildrutor i sektioner.

6.1.4 Avgöra hur skarp en bild är

Den metod som förklaras i avsnitt 2.2.1, och använder Laplace-operatorn som grund, kan användas för att sätta ett värde på hur skarp en given bild är. Motiveringen till att använda denna metod är att en studie av andraderivatorna ger en god uppskattning av bildens variationer, vilket indikerar hur tydlig bilden är. Däremot är metoden känslig för brus och fungerar därför endast bra då inga reflektioner finns i bilden. Detta demonstreras i Figur 6.2. I samma figur under (b) syns det tydligt att Laplace-algoritmen tolkar reflektioner som skarpa, vilket ger ett högt värde när det egentligen borde vara lågt. Ett val av en suddig sektion istället för en skarp är destruktivt för slutresultatet. Därför används istället Tenengrad-algoritmen som hanterar reflektioner korrekt, vilket syns i Figur 6.3.



(a) Skarphetsvärde: 495.38



Skarphetsvärde: 659.86

(b) Skarphetsvärde: 12927.78

Skarphetsvärde: 12814.61

Figur 6.2: Skarphetsvärden uträknade på respektive bild med Laplace-algoritmen. Ett högre värde indikerar en skarpare bild.

(a) Skarphetsvärde: 24791.49

(b) Skarphetsvärde: 28200.06

Figur 6.3: Skarphetsvärden uträknade på respektive bild med Tenengrad-algoritmen. Högre värde indikerar en tydligare bild.

6.2 HDR

Tekniken för HDR baseras på olika exponeringsvärden. I mörka förhållanden har kameran problem med att den inte får tillräckligt mycket ljus. Bilderna som slås ihop av HDR-algoritmer kan använda bilder med ökad exponeringstid jämfört med ett vanligt foto och därför ta in mer ljus vilket gör att mörka ytor på bilden blir ljusare.

Det finns ett flertal olika algoritmer för HDR, där de flesta kameratillverkare har implementerat sin egna version som kommunicerar direkt med kamerahårdvaran. På grund av detta kan algoritmen sätta ihop olika bilder väldigt effektivt då den vet precis hur bilden har tagits och hur översättningen mellan kamera och bild ser ut. Utöver detta finns det HDR-algoritmer som kan arbeta med bilder där den inte känner till kamerahårdvaran och därför får annorlunda resultat. Några av dessa kan implementeras med hjälp av OpenCV och valdes därför för undersökning.

6.2.1 Algoritmer med uppskattning av kamerasvarsfunktion

Robertsons och Debevecs algoritmer fungerar utan att känna till kamerahårdvaran. De behöver olika exponeringstider som indata, med hjälp av vilka de utför beräkningar för att uppskatta kamerasvarsfunktionen. Detta gör att de väljer hur den behandlade HDR-bilden kommer att se ut på ett annat sätt än en HDR-algoritm som känner till kamerahårdvaran. De här metoderna har undersökts för att se om bättre resultat än med en inbyggd HDR-algoritm kan fås. Problemet med algoritmerna är att olika exponeringsvärden inte alltid betyder olika exponeringstid. Det

gör att de ofta får dålig färgåtergivning vilket kan ses i Figur 6.4 i ljusa förhållanden och Figur 6.5 i mörka förhållanden.

6.2.2 Exponeringsfusion

Mertens algoritm fungerar annorlunda än HDR-algoritmer som antingen känner till kamerans svarsfunkion eller uppskattar den. Genom att istället undersöka bildpunkter och vikta dessa kan potentiellt annan information tas ut från de olika bilderna, eftersom det inte sker uppskattningar som i Robertson och Debevecs algoritmer. Den här algoritmen utför dessutom färre beräkningar vilket gör den effektiv ur prestandasynpunkt. Resultaten för den här algoritmen har bättre färgåtergivning och detta kan ses i fig 6.4 i ljusa förhållanden och fig 6.5 i mörka förhållanden.

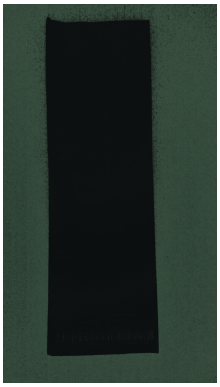
(a) HDR-bild behandlad av Mertens algoritm

(b) HDR-bild behandlad av Debevecs algoritm

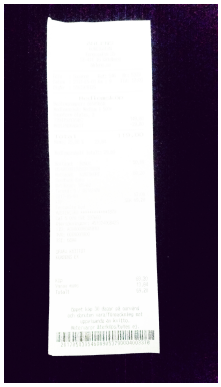
(c) HDR-bild behandlad av Robertsons algoritm

Figur 6.4: Tre bilder tagna i ljusa förhållanden som visar bilder behandlade av de olika HDR-algoritmerna

(a) HDR-bild behandlad av Mertens algoritm



(b) HDR-bild behandlad av Debevecs algoritm



(c) HDR-bild behandlad av Robertsons algoritm

Figur 6.5: Tre bilder tagna i mörka förhållande som visar bilder behandlade av de olika HDR-algoritmerna

6.3 Bildförbättring

Utjämnande filter och tröskeloperationer är generella operationer som kan appliceras på godtyckliga bilder. Operationerna tar inte hänsyn till om bilden som ska behandlas är oprocesserad eller om den är skapad med videolösningen eller HDR. De utvärderas därför separat från de övriga förbättringsmetoderna.

De parametrar som behöver bestämmas vid de olika operationerna prövas fram manuellt för varje bild för att optimala resultat ska uppnås. Detta gör att operationerna blir omständiga att använda. Istället skulle metoder likt Otsus metod behövas för att automatiskt välja optimala parametrar för bilden som ska behandlas.

6.4 Testning

Primärt togs två olika sätt att testa hur väl en maskinläsare kan läsa bilder skapade av de utvecklade förbättringsmetoderna fram. Det ena var att leta efter den mest relevanta informationen på kvittot, såsom totalpris och skatt, och jämföra hur väl den avlästa informationen stämmer överens med den information som faktiskt finns på kvittot. Det andra var att studera hela kvittot och jämföra samtliga tecken som läses av med samtliga tecken som faktiskt finns på kvittot. Då syftet med projektet är att förbättra skanningen av ett kvitto i allmänhet och inte bara att hitta viss specifik information på kvittot, valdes att studera kvittotexten i sin helhet. För att kunna utföra den här testningen behövs en maskinläsare samt en metod för att jämföra strängar. Valen av dessa diskuteras nedan.

6.4.1 Maskinläsare

Det finns flera tillgängliga färdigimplementerade maskinläsare som är fria att använda. De två som främst undersöktes i detta projekt var Microsofts *Computer Vision API* och Googles *Google Vision API*. Valet föll på Google Vision API då det har flera fördelar. Den största fördelen är att Google Vision API visar var i bilden varje ord har hittats, vilket underlättar för att upptäcka eventuella felaktiga avläsningar. Maskinläsaren är dessutom enkel att använda och ger goda resultat vid avläsning.

6.4.2 Jämföra texter

För att jämföra texter används Levenshteinavstånd, som definieras som antalet ändringar som behöver göras på den ena strängen för att skapa den andra. Denna metod kan ge oönskade resultat om texten som undersöks är omvänd, vilket den exempelvis kan vara vid utskriften av datum. Texterna *23 maj* och *maj 23* innehåller egentligen samma information men har trots det ett högt Levenshteinavstånd. Vid det här projektets tester kommer dock kvitton att läsas av rad för rad, uppifrån och ned, vilket gör att informationen alltid kommer i samma ordning. Därför lämpar sig Levenshteinavståndet väl för detta ändamål.

6.4.3 Testmiljö och material

Anledningen till att en Samsung Galaxy S6 valdes vid framtagande av testbilder är främst på grund av att den använder Android som operativsystem. Dessutom fanns en sådan tillgänglig i gruppen. Ett bättre alternativ skulle vara att ha ett urval av olika mobiltelefoner för att kunna ge mer generella resultat.

Eftersom det tidigt visade sig vara svårt att få en blixtreflektion i kvittot, placerades det i en plastficka innan bilder och videor togs. Plastfickan gav väldigt stora reflektioner som inte riktigt återspeglar verkligheten.

Miljön som användes vid framtagande av testdata var inte ett laboratorium med mätbara ljusförhållanden. All testdata har dock tagits fram vid uppskattningsvis likvärdiga förhållanden.

6.5 Utvecklingsmiljö

För att underlätta vid utvecklingen av förbättringsmetoder implementerades projektet till stor del med hjälp av färdig programvara och existerande tekniker.

6.5.1 Bildbehandlingsbibliotek

OpenCV är ett gratis och lättanvänt alternativ för bildbehandling och spelade en viktig roll på vägen mot en färdig produkt. Större delen av all bildbehandling som sker är med hjälp av OpenCV.

6.5.2 Programmeringsspråk

I början av projektet diskuterades ett antal olika programspråkspråk, och planen var inledningsvis att skriva all kod i C# för att underlätta en eventuell integration med Bokios system. Det visade sig dock att OpenCV har bättre stöd för Java än för C#, varför Java valdes i slutändan. Att välja ett språk där det var enkelt att integrera OpenCV var viktigt för att slippa onödig konfiguration och anpassning.

6.5.3 Plattform för mobilapplikation

Operativsystemet Android valdes över iOS, då det är baserat på Java. Ett alternativ skulle vara att använda ett verktyg som Xamarin för att göra applikationen plattformsoberoende. Den skulle då fungera på såväl Android, iOS som Windowstelefoner. Då det beslutades att projektet skulle skrivas i Java valdes dock att enbart bygga applikationen för Android. Det faktum att applikationen främst är byggd i demonstrationssyfte gjorde också att det egentligen inte fanns något behov av att applikationen skulle vara plattformsoberoende.

6.5.4 Plattform för serverapplikation

Serverapplikationen är implementerad med NodeJS, som både kan arbeta asynkront och synkront. Flera processer kan därmed behandlas parallellt och således svara på en förfrågan på en kortare tid.

7
Diskussion

I detta kapitel diskuteras och analyseras de resultat som uppnåtts i projektet. I kapitlet diskuteras även projektets samhällspåverkan, felkällor samt förslag på hur projektet skulle kunna vidareutvecklas.

7.1 Jämförelse av förbättringsmetoder

Redan tidigt under projektet visade det sig vara ett ovanligt problem med blixtreflektion i bilder. Troligtvis är detta på grund av att dagens moderna mobilkameror har en tillräckligt förfinad metod för att ta bilder med blixt så att en reflektion inte uppstår. Detta gjorde att den formulerade problemställningen inte helt stämde överens med verkligheten. För att simulera en blixtreflektion placerades en plastficka framför kvittot. Inget försök att återskapa en blixtreflektion utan plastficka lyckades, och således fanns ingen möjlighet att vetenskapligt testa dess förbättringspotential. Subjektiva observationer från den testdata som togs med plastficka indikerade att det finns möjlighet att minska blixtreflektionen i superbilden, vilket syns i Figur 4.3.

7.1.1 Videolösning

Videolösningen som implementerades har sin styrka då en reflektion har uppstått, oavsett om den kommer från kamerans blixt eller från en extern ljuskälla. Detta är på grund av att reflektionpunktens placering i bilden kommer att förflyttas, vilket innebär att den sedan kan tas bort helt och hållet när den slutliga bilden sätts samman.

7.1.2 HDR

De resultat som har observerats har påvisat att de egenutvecklade implementationerna av HDR-algoritmer inte är bättre än mobilkamerans inbyggda HDR-funktion. Mobilkamerans HDR har ett övertag då den känner till hur kamerasvarsfunktionen ser ut genom direkt tillgång till kamerahårdvaran. Robertson- och Debevecalgoritmernas uppskattningar av denna funktion är inte tillräckligt pålitlig för att återskapa ett motiv. På grund av detta lyckas de inte ta fram en HDR-bild i mörker som kan användas för maskinläsning. Mertens algoritm visar att det inte är nödvändigt att använda denna metod för att få ett acceptabelt resultat. Överlag har styrkan med HDR påvisats då belysningen ej är konstant över kvittot. När kvittot har både ljusa och mörka delar kan HDR ge rätt exponering åt både de mörka och de

ljusa delarna. Om däremot en reflektion skulle uppstå saknar HDR tillvägagångsätt för att få fram det innehåll som döljs av reflektionen.

7.1.3 Jämförelse av videolösning och HDR

Vilken metod som fungerar bäst varierar mellan varje specifikt scenario. Det skulle således vara optimalt att kunna avgöra vilken förbättringsmetod som löser just den aktuella bilden på bästa sätt. En första idé för hur detta kan lösas var att testa att ta en bild med mobiltelefonens inbyggda HDR först. Ett sätt att avgöra om denna bild är tillräckligt bra för avläsning är att undersöka nivån av skarphet i bilden. Om den inte skulle vara bra nog skulle applikationen istället automatiskt skifta över till videoläge. I videolösningen tas skarphetsvärdet redan ut för att hitta de bästa sektionerna från delbilder, men scenariot i applikationen blir lite annorlunda. I videolösningen plockas de delbilder med högst skarphetsvärde ut för att bygga superbilden, oavsett vad värdet är, men för att validera en HDR-bild behöver däremot ett tröskelvärde hittas för att avgöra när en bild inte anses vara bra nog.

7.1.4 Bildförbättring

Resultaten visar att ett Gaussiskt filter kan förbättra maskinläsningen av en bild något. Filtrets påverkan skiljer dock mycket mellan olika bilder och det är inte alltid maskinläsningen blir bättre. Även medianfiltrets påverkan kan skilja mycket mellan olika bilder och det är därför svårt att dra några konkreta slutsatser om filtren.

Vad gäller tröskeloperationer kan en binär avbildning också förbättra maskinläsningen en aning. Detta är dock när Otsus metod inte används, utan tröskelvärdet prövas fram manuellt. Detta visar att Otsus metod inte är optimal för att välja ett tröskelvärde som förtydligar texten, utan en bättre metod för detta avseende skulle behövas. Precis som för de utjämnande filtren skiljer sig påverkan av tröskeloperationerna mellan olika bilder, och det är därmed svårt att dra slutsatser om dessa också.

7.2 Relaterade arbeten

Inga relaterade arbeten som jämför de förbättringsmetoder som projektet föreslår har hittats. Däremot finns det ett stort antal dokumenterade arbeten för delarna som rör bildförbättring och HDR, av vilka ett urval används.

7.3 Projektets samhällspåverkan

Projektet är av undersökande natur och ämnar att hitta den bäst lämpade metoden för att skanna in kvitton i mörker. Det är således ett begränsat område där ingen större samhällspåverkan sker. Ur ett hållbarhetsperspektiv är det främst aspekterna gällande social och ekonomisk hållbarhet som påverkas. Med hjälp av en förbättrad skanning kan användaren spara tid jämfört med att manuellt digitalisera information som inte kan urskiljas vid maskinläsning. Att underlätta digitaliseringen har stor

betydelse då dagens samhälle går mer och mer mot en fullständigt digitaliserad värld.

Eftersom resultatet integreras i en mjukvaruprodukt krävs ingen produktion och således ingen materialanvändning. Mjukvaran kräver dock tillgängliga resurser på en webbserver som ska tillhandahållas, exempelvis i en serverhall. I denna serverhall är kylning viktigt och mycket el utnyttjas för att nå eftersträvad temperatur.

7.4 Felkällor

Det finns ett flertal olika felkällor kopplade till olika områden inom projektets ramar. Först och främst var det en stor variation av resultatet mellan de olika mätningarna. I syfte att få ett mer tillförlitligt resultat gjordes tio mätningar för varje scenario. De resultat som presenteras är ett medelvärde av dessa.

Dessutom skulle det ha varit bra att använda ett stativ vid framtagning av testdata. Detta skulle innebära att avståndet mellan kameralinsen och kvittot är likvärdigt mellan de olika testfallen. Istället för detta användes en egenbyggd kameraramp, där avståndet kan ha varierat en aning.

Vidare fanns det en maximal filstorlek på fyra megabyte på den maskinläsning som användes. För att uppfylla detta krav beskars bilderna manuellt i syfte att exkludera irrelevant bakgrund. Det är möjligt att detta kan ha haft en betydelse för kvaliteten på avläsningen.

Slutligen har den inbyggda HDR-funktionen för tre olika mobiltillverkare testats eftersom denna kan variera mellan olika mobiltelefoner. Alla resultat i projektet kommer dock från en mobiltelefon av samma tillverkare, vilket inte ger lika generella resultat som ifall olika tillverkares HDR-funktioner hade använts.

7.5 Vidareutveckling

I detta avsnitt diskuteras tekniker som hade varit intressanta att inkludera i projektet för utveckla det vidare.

7.5.1 Prestanda

Det tar lång tid att behandla videon i OpenCV. Prestanda var något som projektet avgränsades ifrån, men för att applikationen ska kunna användas i kommersiellt syfte skulle den behöva bli betydligt snabbare. Det är inte möjligt att optimera de metoder som används i OpenCV. Om applikationen skulle frikopplas från OpenCV och istället implementera de metoder som krävs på egen hand skulle optimering vara möjlig. Ett alternativ till detta skulle vara att börja plocka ut delbilder ur videon medan den filmas. Det skulle göra att den mest tidskrävande processen, att urskilja delbilder från videon, kan ske i bakgrunden i realtid.

7.5.2 Kantdetektion

Applikationen skulle behöva hantera kantdetektion med vit bakgrund på ett bättre sätt än vad den gör i nuläget. Detta för att öka användarvänligheten och minska felkällorna. Applikationen behöver också kunna hantera ett scenario då videon inte lyckas fånga hela kvittot.

7.5.3 Neuralt nätverk

En intressant aspekt på vidareutveckling av applikationen skulle vara att introducera ett neuralt nätverk. Denna typ av maskininlärning skulle göra att systemet kan förbättra bilden utan att behöva videolösningen. Istället baseras förbättringen på tidigare erfarenheter och lärdomar som systemet gjort med träningsdata.

Ett alternativ till neurala nätverk skulle vara att på algoritmiskt vis försöka förbättra bilden baserat på rörelser under kamerans exponeringstid. Det kan då gå att spåra hur en bildpunkt smetas ut och gör bilden suddig.

7.5.4 Användarvänlighet

För att minska felkällorna skulle användarvänligheten i applikationen kunna förbättras. Detta var aldrig prioriterat i projektet eftersom fokus var på algoritmdelen, men en sådan förbättring skulle kunna fånga många fel innan hela videon behöver behandlas på servern.

7.5.5 Skanna flera kvitton på samma gång

Tillsammans med Bokio har en vidareutvecklingsmöjlighet till projektet diskuterats. För företagets del kan det finnas kunder som vill bokföra en mängd kvitton på samma gång. Därför skulle en lösning där användaren skannar flera kvitton med sin telefon på samma gång vara värdefull. Det finns stora likheter mellan en sådan lösning och den som har utvecklats under detta projekt. För varje kvitto skulle en superbild genereras för att bli av med en eventuell reflektion samt för att förbättra skärpan. Det finns dock flera problem som måste lösas för att en sådan lösning ska fungera.

- Lösningen måste kunna hantera ett scenario där det inte kan bildas en omslutande polygon baserat på fyra hörnpunkter, samt det fall då fyra hörn som tillhör olika kvitton hittas.
- Lösningen behöver kunna identifiera unika kvitton så att inte samma kvitto hittas två gånger om användaren skulle skanna i en sådan rörelse att start och slut är på samma ställe.
- Lösningen behöver framgångsrikt kunna hitta kvitton mot vit bakgrund. Om flera kvitton ska skannas kan det vara svårt att hitta en mörk bakgrund stor nog för att omsluta alla kvitton. Lösningen behöver därför bli bättre på att hitta kvittons hörn och omslutande polygoner även med vit bakgrund.
- Metoden är i nuläget långsam då bara ett kvitto skannas, och skulle följaktligen bli ännu långsammare om fler kvitton skulle behandlas.

7.5.6 Reducering till fyra hörn

Anledningen till att polygonen som omsluter kvittot begränsas till att ha fyra hörn är för att kvittot antas vara av god kvalitet. Däremot är detta en förenkling som inte alltid är realistisk, då kvitton lätt kan få hörn som blir krökta. I fallet när det finns ett krökt hörn har en korrekt omslutande polygon istället fem hörn. För att perspektivmetoderna, som bygger på att det finns fyra hörn, ska fungera korrekt kan ett virtuellt hörn skapas.

8

Slutsats

Syftet med projektet var att undersöka metoder för att förbättra skanning av kvitton i mörka förhållanden. De olika metoderna har utvärderats med hjälp av maskinläsning samt Levenshteinavstånd.

Superbilden som sammansätts från en video fick något sämre resultat än ett vanligt fotografi med blixt. Anledningen är att bildrutor från en rörlig video inte kan uppnå samma kvalitet som ett fotografi där större delen av kvittot hamnar i fokus. Videolösningen var däremot effektiv för att bli av med reflektionsytor som är avgränsade till delar av ett kvitto. Detta gör att den kan användas på ytor som ger mycket reflektioner och där avläsningen drabbas av detta.

Den inbyggda HDR-funktionen i mobilen fick minimalt bättre resultat än ett vanligt foto. Detta förutsatt att bilderna tagna för HDR togs med exakt samma perspektiv. I de fall där kameran rört sig mellan bilderna fås en suddig HDR-bild vilket leder till sämre resultat. De olika egenutvecklade implementeringarna av HDR-algoritmer presterade olika bra i olika förhållanden. Mertens algoritm var den enda av dem som gav resultat i mörka förhållanden.

Mobilkamerans inbyggda HDR presterade bättre än de egenutvecklade implementeringarna. Detta är dock svårt att dra slutsatser kring då det ej går veta hur en enskild mobilenhets inbyggda HDR fungerar, samt vilka algoritmer som används.

Videolösningen gav inte ett bättre resultat än en HDR-bild tagen med en mobilkamera. Dagens mobiltelefoner har tillräckligt kraftfulla kameror för att problemet med reflektionsytor inte ska uppstå. Således är problemet minimalt och kräver inte att videolösningen används för att förhindra reflektionsytor.

Utjämnande filter och tröskeloperationer har potential att underlätta maskinläsning av kvittobilder något men deras påverkan skiljer sig mellan olika bilder. För att operationerna ska vara användbara i verkligheten skulle metoder som automatiskt utnyttjar dem optimalt för varje enskild bild behövas.

Mobilapplikationen som utvecklades fungerade som ett verktyg för att demonstrera de utvecklade algoritmerna. I verkligheten är i nuläget ingen av dem lämpliga för kommersiellt bruk då de inte är tillräckligt effektiva tidsmässigt.

Litteraturföreteckning

[1] Bokio. (2017). Bokio, URL: https://www.bokio.se (hämtad 2017-05-03).

[2] E. Arias-Castro och D. L. Donoho, "Does median filtering truly preserve edges better than linear filtering?", *The Annals of Statistics*, s. 5, 2009.

[3] NIST/SEMATECH, *E-handbook of statistical methods*, http://www.itl.nist.gov/div898/handbook/eda/section3/eda3661.htm, 2013.

[4] N. Otsu, *A threshold selection method from gray-level histogram*, http://ieeexplore.ieee.org/document/4310076, 1979. (hämtad 2017-04-05).

[5] OpenCV, *Basic thresholding operations - opencv 2.4.13.2 documentation*, http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html#threshold-binary, 2014. (hämtad 2017-05-03).

[6] ——, *Basic thresholding operations - opencv 2.4.13.2 documentation*, http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html#threshold-binary-inverted, 2014. (hämtad 2017-05-03).

[7] ——, *Basic thresholding operations - opencv 2.4.13.2 documentation*, http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html#truncate, 2014. (hämtad 2017-05-03).

[8] ——, *Basic thresholding operations - opencv 2.4.13.2 documentation*, http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html#threshold-to-zero, 2014. (hämtad 2017-05-03).

[9] ——, *Basic thresholding operations - opencv 2.4.13.2 documentation*, http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html#threshold-to-zero-inverted, 2014. (hämtad 2017-05-03).

[10] C. Nordling och J. Österman, *Physics Handbook*. 2013, ISBN: 978-91-44-04453-8.

[11] J.L.Pech-Pacheco, G. Cristóbal, J. Chamorro-Martínez och J. Fernández-Valdivia, *Diatom autofocusing in brightfield microscopy: A comparative study*, http://optica.csic.es/papers/icpr2k.pdf, 2000. (hämtad 2017-04-19).

[12] A. Campilho och M. Kamel, *Image analysis and recognition*, https://books.google.se/books?id=Phlog9JD-uQC, 2008.

[13] OpenCV, *Sobel operator*, http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html, 2009.

[14] F. S. Ray, "Camera exposure determination", *The manual of photography: Photographic and digital imaging*, 2000.

[15] M. Grossberg och S. Nayar, "Determining the camera response from images: What is knowable?", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, s. 1455–1467, 11 nov. 2003.

[16] R. Hunt, *The reproduction of colour in photography, printing and television*. John Wiley & Sons, 2004.

[17] P. E. Debevec och J. Malik, "Recovering high dynamic range radiance maps from photographs", aug. 2008.

[18] "Thermal insulation – heat transfer by radiation – physical quantities and definitions", International Organization for Standardization, Standard ISO 9288:1989(E), 1989.

[19] M. Robertson, S. Borman och R. Stevenson, "Dynamic range improvement through multiple exposures", vol. 4, sept. 1999.

[20] N. Black och S. Moore, *Gauss-seidel method*, `http://mathworld.wolfram.com/Gauss-SeidelMethod.html`. (hämtad 2017-05-01).

[21] T.Mertens, J.Kautz och F. Reeth, "Exposure fusion: A simple and practical alternative to high dynamic range photography", *Computer graphics forum*, vol. 28, s. 161–171, 1 mars 2009.

[22] E. Weisstein, *Gaussian function*, `http://mathworld.wolfram.com/GaussianFunction.html`, Mathworld–A Wolfram Web Resource. (hämtad 2017-05-11).

[23] W. H. Encyclopedia, *Ramer–douglas–peucker algorithm*, `http://www.worldlibrary.org/articles/ramer-douglas-peucker_algorithm`.

[24] R. Rao, *Lecture on features and image matching*, `https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect6.pdf`, 2009. (hämtad 2017-04-25).

[25] OpenCV, *Geometric transformations*, `http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html`, 2009.

[26] T. A. S. Foundation, *Levenshtein distance*, `https://commons.apache.org/proper/commons-lang/apidocs/org/apache/commons/lang3/StringUtils.html#getLevenshteinDistance-java.lang.CharSequence-java.lang.CharSequence-`. (hämtad 2017-04-25).

[27] A. Telea, "An image inpainting technique based on the fast marching method", *Journal of graphics tools*, vol. 9, nr 1, s. 23–34, 2004.

[28] OpenCV, *Contours : Getting started*, `http://docs.opencv.org/trunk/d4/d73/tutorial_py_contours_begin.html`, 2009.

[29] ——, *Feature matching*, `http://docs.opencv.org/trunk/dc/dc3/tutorial_py_matcher.html`, 2009. (hämtad 2017-04-25).

[30] Google, *Camera2*, `https://developer.android.com/reference/android/hardware/camera2/package-summary.html`, 2017. (hämtad 2017-04-29).

[31] N. Foundation, *Nodejs*, `https://nodejs.org/en/`, 2017. (hämtad 2017-05-02).

[32] OpenCV, *Geometric transformations*, `http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#resize`, 2009.

[33] A. Tanbakuchi, *Comparison of opencv interpolation algorithms*, `http://tanbakuchi.com/posts/comparison-of-openv-interpolation-algorithms/`, 2016. (hämtad 2017-05-24).