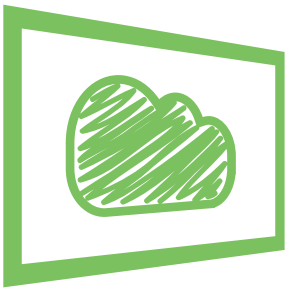
CHALMERS



Tavlan

Automatiserat system för insamling och distribuering av information skriven på krittavlor

Kandidatarbete inom information och datavetenskap

Pontus Andersson, Filip Bertilsson,
Felix Hallqvist, Simon Nilsson,
Jonathan Olsson, Mats Uddgård

Chalmers Tekniska Högskola
Institutionen för Data- och Informationsteknik
Göteborg, Sverige, Juni 2014

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet. The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Automatic system for collecting and distributing written information on blackboards.

P. Andersson
F. Bertilsson
F. Hallqvist
S. Nilsson
J. Olsson
M. Uddgård

© P. Andersson, Juni 2014.
© F. Bertilsson, Juni 2014.
© F. Hallqvist, Juni 2014.
© S. Nilsson, Juni 2014.
© J. Olsson, Juni 2014.
© M. Uddgård, Juni 2014.

Examiner: A. Linde

Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: Our product logotype.

Department of Computer Science and Engineering
Göteborg, Sweden, June 2014

Abstract

This paper describes the development of an automatic system for capturing images of blackboards, image processing and publishing the resulting images to a web server. At the time of this paper much of the education was still done with the use of blackboards. During lectures it can be hard to follow the lecturers reasonings and conclusions while trying to write down what has been written on the board. The purpose of this bachelor's thesis is to make it easier for students by creating an automatic note taking system.

The thesis describes the early stages of the project with planning and feasibility studies in order to be able to decide how to best develop the product. The development process describes all the design choices and the different modules like the camera unit and the server. With a working prototype, testing was made on all the different modules of the project to be able to evaluate and assess the safety of the product.

The result is later processed and a discussion is made on how the product can be made better in the future. The product resulted in a prototype where all expected basic functionality were fully operational. The product were shown to a number of people and most of them had a positive attitude towards the product.

It was decided to use one single-board computer in every lecture hall with one camera for each blackboard column and one sensor for eack blackboard. Raspberry Pi was chosen to be used as the single-board computer. Raspberry Pi Camera Modules and photomicrosensors were chosen as components to implement our prototype. Linux was used as operating system for both the camera unit as well as the server. A blackboard is photographed when it is in top position in its respective column and the resulting image is sent to the server.

A web server was created where one can search and find lecture notes. An administrator interface was created to be able to manage camera units, courses and users.

Image processing was implemented from scratch through the use of relevant litterature. Image processing incorporates cropping, skew correction, determining if something is written on the blackboard and the ability to compare earlier blackboard images with newly taken ones to reduce the number of duplicates.

Sammanfattning

Denna rapport beskriver utvecklingen av ett autonomt system för bildtagning av krittavlor, bildbearbetning samt uppläggning av detta material på en webbserver. I skrivande stund så sker mycket undervisning fortfarande på krittavlor. Under föreläsningar kan det vara hektiskt att följa föreläsarens resonemang och slutledningar samtidigt som man försöker föra anteckningar från tavlan. Av denna anledning fokuserar detta kandidatarbete till att försöka underlätta för studenter genom att skapa ett autonomt anteckningssystem.

Kandidatrapporten beskriver hur det i början handlade om planering och förstudier för att kunna välja på vilket sätt produkten skulle sättas ihop. Utvecklingsprocessen beskriver alla designval och vilka moment som gjordes i varje separat del av projektet så som för kameraenhet och server. Med en fungerande prototyp genomfördes testning av alla delar av projektet för att kunna göra en utvärdering och en säkerhetsbedömning.

Det valdes att placera en enkortsdator i varje med upp till tre kameror och en givare per krittavla i varje föreläsningssal. Som enkortsdator valdes det att använda en Raspberry Pi. Till enkortsdatorn anslöts kameror av typen Raspberry Pi Camera Module och läsgafflar som givare för att determinera positionen av krittavlor. Både på kameraenheten och den centrala servern användes operativsystemet Linux. En krittavla fotograferas när denna är vid sitt toppläge.

En webbsida gjordes där det går att söka och visa föreläsningsanteckningar. Det gjordes även ett administrationsgränsnitt för att kunna administrera kameraenheter, kurser och användare.

Bildbehandlingen implementerades från grunden med hjälp av relevant litteratur. Bildbehandlingen innefattar utskärning av krittavlan, skevhetskorrigering av den utskurna tavlan, determinering om något är skrivet på en krittavla och jämföra tidigare krittavlor med varandra för att minska antalet dubbletter av bilder.

Förord

Denna rapport behandlar ett kandidatarbete som genomfördes som ett samarbete mellan studenter på Chalmers Tekniska Högskola. Kandidatarbetet har utförts under en termin och omfattar 15 hp. Denna rapport beskriver utvecklingen av ett autonomt föreläsningsanteckningssystem.

Vi vill speciellt rikta ett stort tack till vår handledare Lena Peterson, vår examinator Arne Linde, vår inspirerande föreläsare Johan Karlsson på Matematiska vetenskaper och andra inblandade personer. Vi tackar även arkitekternas eminenta A0-printer som förgyllde vårt grupprum.

Ordlista

API Application Programming Interface. En regeluppsättning för hur ett visst program kan kommunicera.

Binär bild En bild där pixlarna är binära. Man kan säga att antingen finns pixeln eller ej.

Breadcrumbs Ett designmönster för att visualisera hierarkin på ett grafiskt gränssnitt.

CPLD Complex Programmable Logic Device. En krets med programmerbar logik.

Datorseende När en dator analyserar en bild för att förstå innehållet.

Enkortsdator En dator som är på ett enda kretskort och kan utföra samma uppgifter som en mikrodator.

ER-diagram Ett diagram över en relationsbaserad databas modellerad efter standarden "Entity-relationship model".

FPGA Field-Programmable Gate Array. En krets med programmerbar logik. Ungefär som en CPLD men uppbyggd på ett annat sätt.

Färgkanal En endimensionell skalär för att beskriva magnituden av en färg i pixlarna.

GPIO General-Purpose Input/Output, Programmerbara pins för input/output.

Gradient En vektor uppbyggd av partialderivatorna till en funktion med flera variabler. Det vill säga att om f(x,y,z) så blir grad f = (df/dx, df/dy, df/dz).

Gråskala En kanal som summerar de tre färgkanalerna (röd, grön och blå) och beskriver bilden i svartvitt.

Hough-transform En transform som transformerar pixeldata till geometriska figurer. Transformen är uppkallad efter Paul Hough.

HTTP Hyper Text Transfer Protocol. Det protokoll som används av webbläsare för att hämta data från en webbserver.

Kameraenhet Datorn som befinner sig i varje föreläsningssal som tar kort.

MIPI Mobile Industry Processor Interface. En global organisation som utvecklar gränssnitt specifikationer för det mobila ekosystemet.

MVC-modell Model View Controller. Ett designmönster för att dela upp en applikation i en del för data(Model), en del för användargränssnittet(View) och en del för att hantera indata från användaren(Controller).

Referenspunkt En punkt man utgår från när man ska beskriva allt annat. Beskrivs i två dimensioner oftast som (0,0).

Relationsdatabas En databas där information ordnas med relationer med eventuella restriktioner.

SCP Secure Copy. Ett protokoll för säker överföring av filer från en dator till en annan.

Slutanvändare En person som kommer att använda sig av produkten. I detta fall kommer det primärt vara en student.

Tröskelvärde Ett värde som används för att undersöka om andra värden är över eller under en viss gräns. Används ofta för filtrering för att bli av med för låga värden.

Vikt Ett värde som representerar hur signifikant något är.

XML Extensible Markup Language. Ett sätt att strukturera data.

Innehåll

1 Inledning

Föreläsningar utgör en stor och betydande del för en students inlärning. Det är därför väldigt viktigt att kunna tillgodogöra sig så mycket som möjligt av den undervisning som ges. Under en föreläsning för eftergymnasial utbildning så brukar det vara mycket att ta in. Att föra anteckningar samtidigt som man skall hänga med på föreläsarens tankegångar och retorik kan vara väldigt ansträngande. Användning av krittavlor är fortfarande populärt bland högskolor och därför skulle en teknik för att automatiskt digitalisera anteckningarna från fysiska tavlor vara en attraktiv lösning. Att föreläsningsanteckningar är åtråvärda för studenter kan lätt ses. Redan för 30 års sedan fanns det personer som betalades för att se till att föreläsningsanteckningar skulle spridas till gemene student. Det här berättade Peter Lundin, avdelningschef för Nätverk och system vid Chalmers tekniska högskola, 2014-06-03.

1.1 Bakgrund

Idén uppkom under en föreläsning. Från början av universitetstiden försökte vi hinna med att både anteckna och följa föreläsarens tankegångar samtidigt, detta gick inte så smidigt då det blev för mycket att ta in på en gång. Efter det testade vi att ta kort med hjälp av en mobiltelefon men en del av bilderna blev dåliga och det blev svårt att se vad som var skrivet på tavlan. Det var också svårt att hitta rätt föreläsningsanteckningsbild i mobiltelefonen på grund av att bilderna där inte var sorterade på något bra sätt. Allt detta ledde till uppkomsten av idén.

1.2 Utmaningar

Ett system skall konstrueras där allt som skrivs av föreläsaren på tavlan automatiskt ska lagras i en databas, som är tillgänglig för studenter via en hemsida. Systemet skall vara kostnadseffektivt och användarvänligt för alla berörda. Första steget är att kunna identifiera när en tavla är färdigskriven så systemet vet när ett foto skall tas. När ett foto tagits skall det behandlas för att undvika en skeva bilder med onödig information omkring tavlan. Efter en bild är behandlad skall den sorteras och kategoriseras i en databas så att specifika föreläsningsanteckningar blir lätta att hitta. För att inte databasen skall fyllas utav bilder som är snarlika, tomma eller tagna vid fel tillfälle måste även alla bilder från en specifik föreläsning granskas individuellt samt jämföras mot tidigare tagna bilder från föreläsning. Till sist krävs ett system för att studenter lätt skall kunna komma åt föreläsningsanteckningarna via en hemsida där en verifiering på behörighet till universitetets utbildning krävs för att visa innehållet. Systemet i sin helhet skall inte innebära nämnvärt mer arbete för föreläsaren.

1.3 Syfte

Syftet med projektet är att skapa en produkt som är menad att underlätta för studenter på eftergymnasiala utbildningar. Produkten skall tillåta studenterna att fokusera på själva föreläsningen som helhet istället för att stressa för att hinna med att anteckna. Tanken är att studenten under lektionen inte skall behöva lägga allt för mycket fokus på att anteckna utan kan fokusera vidare på de visuella och auditiva intrycken och ta korta notiser vid behov.

1.4 Avgränsningar

Dessa är våra avgränsningar för projektet:

1. Vi antar att föreläsaren är inbegripen i hur systemet fungerar och om så inte är fallet kommer det i projektet inte finnas någon försäkring om att bilderna kommer laddas upp på webbsidan.

2. Vi kommer inte beakta lagar och regler gällande distribuering av föreläsningsanteckningar ägda av föreläsare eller andra personer.

3. Vi förväntar oss inte att den slutliga prototypen innehåller alla aspekter som en slutprodukt för distribution bör ha.

2 Teori och teknisk bakgrund

När en människa ser och identifierar ett objekt så sker detta intuitivt och är inget som sker medvetet, men en dator kan inte analysera bilder på ett enkelt vis. Istället behöver den använda relativt tunga och komplicerade algoritmer för att få relevant information ur bilden, vilka dessutom sällan är generella utan brukar vara designade för specifika ändamål. Detta kapitel tar därför upp teorin bakom de algoritmer och datastrukturer som systemets bildbehandling och bildjämförelse använder.

Kapitlet tar upp teorin för att identifiera objekts kanter och hur kantpixlar i en linje transformeras till deras geometriskt beskrivna linje, då systemet behöver hitta krittavlans ram. Geometriska egenskaper för de krittavlor systemet ska identifiera beskrivs, vilket kommer användas för att datorn ska känna igen dem. Sist förklaras vad ett hierarkiskt rekursivt rutnät är då detta används i bildjämförelsen.

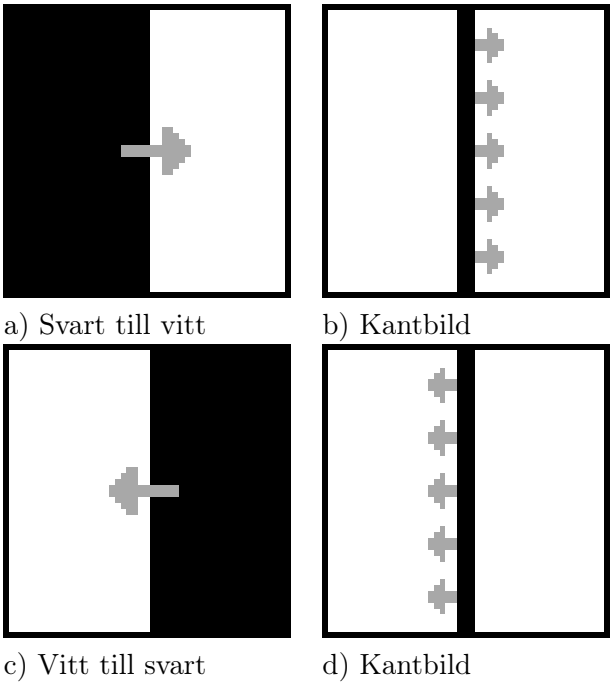
2.1 Teorin bakom detektion av kanter

Detektion av objekt sker oftast på relativt lika sätt oavsett vad syftet med datorseendet är och kan liknas vid hur även människor ser dem. Låt oss föreställa oss ett vitt rum, likt en fotostudio. Placera sedan en lika vit kub i rummet och det kommer vara extremt svårt att skilja denna från väggarna. Om kubens kanter vore svartmålade skulle det däremot vara mycket enklare. Anledningen är att när människan ska identifiera ett objekt så är kontrasten mellan objektet och dess omgivning viktig och i en bild utan några konturer så blir det omöjligt att skilja objektet från dess bakgrund.

Figur 1: Fotografi före och efter kantdetektion. [1]

Eftersom kontrasten mellan objekt och dess omgivning är väsentlig brukar datorseende använda sig av så kallade kantdetektionsalgoritmer. Dessa algoritmer räknar ut kontrastskillnaderna i bilden och analyserar dem för att identifiera objektets konturer. Vid en stark kontrastskillnad i en punkt är det troligtvis en kant till ett objekt i den punkten. Kontrasten för en punkt kan matematiskt förklaras som bildens gradient i den punkten, där kontrastskillnaden är gradientens magnitud och kontrastriktningen är vinkeln mellan gradienten och bildens x-axel. [2]

Oberoende på syftet med datorseendet brukar det vara att föredra att kantdetektionens svar är så exakt som möjligt och gärna även så tunt som möjligt. Många implementationer offrar gärna exakthet om det resulterar i en hastighetsökning och andra kan se det som viktigare att de får så lite störningar som möjligt. För vissa implementationer är det dessutom viktigt att riktningen på kanten identifieras. Riktningen är åt vilket håll kontrasten förändras. Om man exempelvis har en bild som går från svart på vänster sida till vitt på högersidan, kommer dess kant ha motsatt riktning mot en bild där de svarta och de vita områdena byter plats. Se Figur 2.



a) Svart till vitt b) Kantbild

c) Vitt till svart d) Kantbild

Figur 2: Bilder med hög kontrast i mitten och deras kantbilder.

I bilderna a) och c) från figur 2 sker en värdeförändring (vänster till höger) från svart till vitt respektive vitt till svart. Värdeökningen sker alltså i pilens riktning och de resulterande kanterna b) och d) är därför på samma position men med motsatt riktning.

2.1.1 Malldetektion

Malldetektion är en algoritm som bygger på användandet av så kallade mallar [2] [3]. En mall kan ses som en mindre bild vars pixlar är vikter. Dessa mallar läggs över varje pixel i bilden och vikterna multipliceras då med värdet i den pixel som vikten överlappar. Resultatet summeras och läggs in i motsvarande pixel i kontrastbilden. Bilderna a) och b) i figur 3 visar på Sobels mallar som approximerar gradientens x- respektive y-komponent och består av enbart två mallar. Det är vanligast att mer än en mall används. Deras antal, utseende och hur de används beror på vad algoritmen ska uppnå. Jämförelsevis så använder Kirsch åtta mallar [3], som beskrivs mera nedan.

Mallarna kan vara associerade med en riktning och detta används olika beroende på vilken algoritm som används. Då Sobels mallar approximerar bildens gradient blir riktningen vinkeln mellan denna och x-axeln.

-1	0	1
-2	0	2
-1	0	1

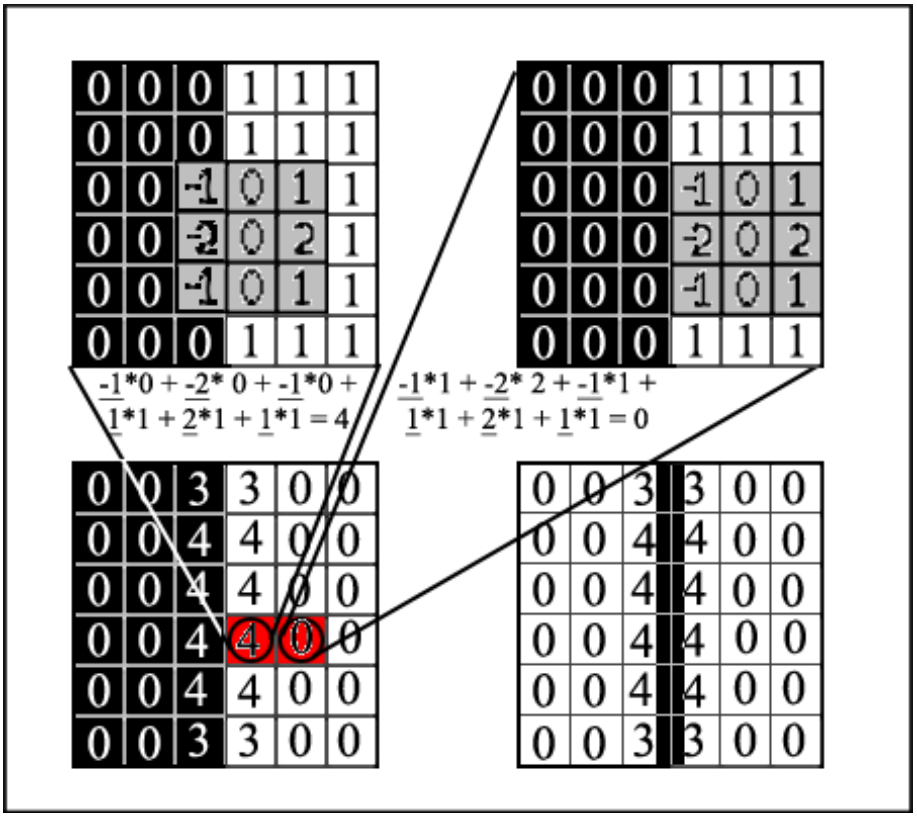
a)

-1	-2	-1
0	0	0
1	2	1

b)

Figur 3: Mallarna som används i Sobel.

I figur 4 så visas ett exempel på hur mallar används. I figuren illustreras processen att få ett kontrastvärde för två markerade pixlar. Mallen läggs över motsvarande pixel i originalbilden (de övre bilderna) och summan blir värdet för den resulterande pixel. Sista bilden visar resulterande kantbild från de uträknade värdena. Kanten ligger egentligen mellan två kolumner och kommer därför räknas som att den är två pixlar tjock. Detta exempel visar bara en av mallarna i Sobel då den andra resulterar i en summa av noll för alla pixlar, då bilden saknar vertikal kontrastförändring.

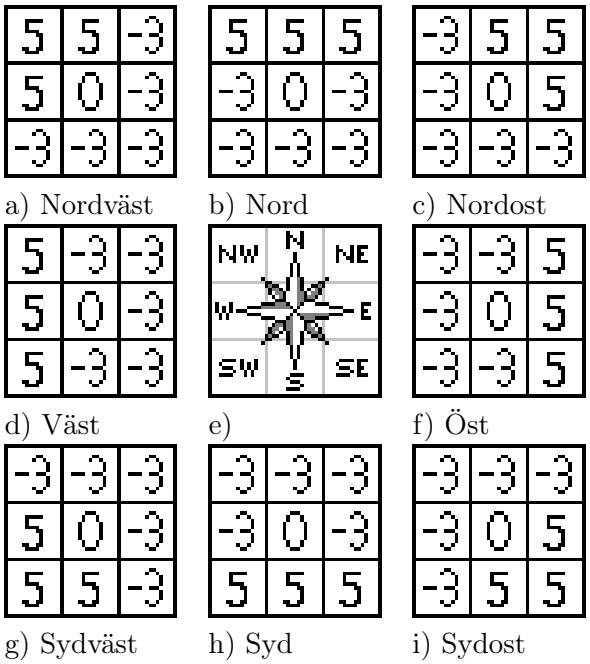


Figur 4: Exempel på malldetektion.

Den resulterande kontrastbilden analyseras med tröskelvärden [2]. Alla kontraster över en viss tröskel accepteras som kanter och förs vidare över till kantbilden, men man kan även använda en lägre tröskel för så kallade svaga kanter. Dessa accepteras som kanter om de är granne med en riktig kant och algoritmen arbetar rekursivt så att deras grannar som är svaga kanter blir accepterade. Tröskelvärdena kan vara absoluta värden som används direkt på kontrastbilden, men det kan även vara fördelaktigt att ha dem i form av procent. Kontrastbildens högsta och lägsta värden används då för att räkna ut hur starka kontrastpixlarna är i procent. Detta resulterar i att samma tröskelvärden kan användas i mer varierande situationer, exempelvis ett rum där ljusstyrkan kan förändras. Istället för att algoritmen inte finner några kanter för att mörker har gjort alla för svaga, kommer den alltid att hitta de starkaste bland de som finns. Dessutom kommer en ljussättning som förstärker även de minsta kontrasterna inte att resultera i felaktiga kanter överallt i kantbilden.

Kirsch är en version av malldetektion som använder åtta specifika mallar [2] [3]. Varje mall är 3*3 pixlar och motsvarar varsin riktning, jämt ut-

delade över en cirkel, likt de 8 väderstrecken. Se Figur 5. Detta för att modellera kanter i de motsvarande riktningarna snarare än att approximera bildens gradient som Sobel. Utslaget är därför hur pass mycket det undersökta området liknar en sådan kant och brukar fungera bättre över sneda kanter än Sobel [2, Fig. 2.10]. I Kirsch räknas kontrasten för pixeln som det starkaste resultatet från mallarna och riktningen brukar approximeras som den mallens vinkel [2]. Man kan även använda en genomsnittlig vinkel från alla mallar, baserad på deras genomsnittliga respons.



Figur 5: Mallarna som används i Kirsch.

2.1.2 Canny

Canny-detektorn är en kantdetektor som är utvecklad att uppfylla tre mål:

- Bra detektion - Alla kanter i bilden ska hittas samtidigt som inga falska utslag får uppstå.
- Bra lokalisering - Kantpixlarna ska vara så nära den riktiga kanten som möjligt.
- Minimal respons - Flera kanter ska inte hittas då endast en kant existerar.

Inom en dimension så ses Canny som ett filter f som faltas med bilden G. Den resulterande bilden H ges av:

$$H = \int_{-W}^{W} G(-x)f(x)dx$$

De målen som Canny ställer ser matematiskt ut så här [4]:

$$SBR = \frac{A|\int_{-W}^{0} f(x)dx|}{n_0\sqrt{\int_{-W}^{W} f(x)^2dx}}$$

$$Lokaliseringsvärde = \frac{A|f'(0)|}{n_0\sqrt{\int_{-W}^{W} f'(x)^2dx}}$$

$$x_{zc} = \pi\left(\frac{\int_{-\infty}^{\infty} f'(x)^2dx}{\int_{-\infty}^{\infty} f''(x)^2dx}\right)^{\frac{1}{2}}$$

SBR är Signal-till-Brus-Ration, det vill säga hur bra filtret är på första punkten. *SBR* ska vara så hög som möjligt vilket det fås genom att signalen ska passera filtret med så lite förändring som möjligt och bruset ska filtreras bort i så stor mån som möjligt. *Lokaliseringsvärdet* är inversen av avståndet mellan den hittade kanten och den riktiga kanten. Det betyder att *lokaliseringsvärdet* ska vara så stort som möjligt för att avståndet ska vara så litet som möjligt. Till sist så är x_{zc} en begränsning som säger att filtret inte ska få för många utslag från en och samma kant i en liten region.

En deriverad gaussfunktion är en bra approximation för att uppfylla målen där $G(x) = e^{-\frac{-x^2}{2\sigma^2}}$ är en gaussfunktion och $G'(x) = (-\frac{-x^2}{\sigma^2})e^{-\frac{-x^2}{2\sigma^2}}$ är dess derivata.

Cannys algoritm appliceras på en bild B, där första steget är att applicera ett endimensionellt gaussfilter, G, för att reducera bruset i bilden. Filtret används längs bildens rader för att få ut B_x som är x-komponenten av bilden och sedan längs bildens kolumner och får ut B_y som är y-komponenten av bilden.

Nästa steg är att applicera ett första derivatans gaussfilter på B_x och B_y som ger ut bildens gradientkomponenter, B'_x respektive B'_y. Sedan slås komponenterna ihop till gradientvektorer där deras magnitud och riktning fås ut genom [2]:

$$M = \sqrt{B_x'^2 + B_y'^2}$$

$$\Theta = \arctan\left(\frac{B_y'}{B_x'}\right)$$

Nu för att få bort mindre relevanta kanter så letas det efter lokala gradientmaximum runt omkring i bilden. Dessa hittas genom att man utgår ifrån en pixel och kollar nästa pixel som kommer i gradientens riktning och även pixeln i motsatt riktning. Om gradienten pekar någonstans mellan två pixlar så används en interpolering mellan de två pixlarnas värden. Sedan jämförs om utgångspixelns magnitud är större än de två interpolerade pixlarnas magnitud. Om så är fallet så ses utgångspunkten som en maximipunkt.

Det sista som görs i Cannys algoritm är att filtrera bort de kanter som inte är kopplade till en stark kant. Två tröskelvärden används för detta, ett högt värde för att identifiera starka kanter och ett lägre värde för svaga kanter. Är en kant stark så betyder det att kanten kommer vara en del av den slutgiltiga kantbilden. Algoritmen utgår från de starka kanterna för att sedan gå till närliggande kanter och ser om dessa kanter klarar av det låga tröskelvärdet. Om så är fallet blir även dessa kanter en del av den slutgiltiga kantbilden.

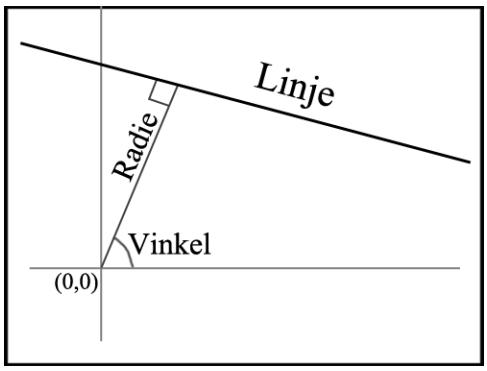
Canny kan konfigureras med olika standardavvikelser och tröskelvärden för att ge det resultat som passar bäst till det aktuella problemet.

2.2 Hough-Transformen

I en kantbild är kanterna fortfarande beskrivna i form av pixlar, vilket är svårt och framförallt dyrt att räkna med när man ska identifiera objekt och hantera dem i en dator. Istället är det oftast fördelaktigt om man kan beskriva dem geometriskt. Hough-transformen är en transform som identifierar geometriska figurer av en viss typ i en bild. Den vanligaste Hough-transformen bygger på att en kontur är uppbyggd av tangent-linjer och de figurer den transformerar till är därför just räta linjer [5]. En rät linje kan matematiskt beskrivas på ett stort antal sätt, de flesta med sina egna fördelar. Nedan följer en beskrivning av olika former och deras fördelar samt nackdelar när de används för att beskriva alla linjer som korsar en pixel i en bild, vilket krävs i en Hough-transform [5].

2.2.1 Räta linjer på normalform

Normalform beskriver linjer med hjälp av vinkeln mellan deras normal och x-axeln samt avståndet längs normalen till en referenspunkt [5]. Se Figur 6.

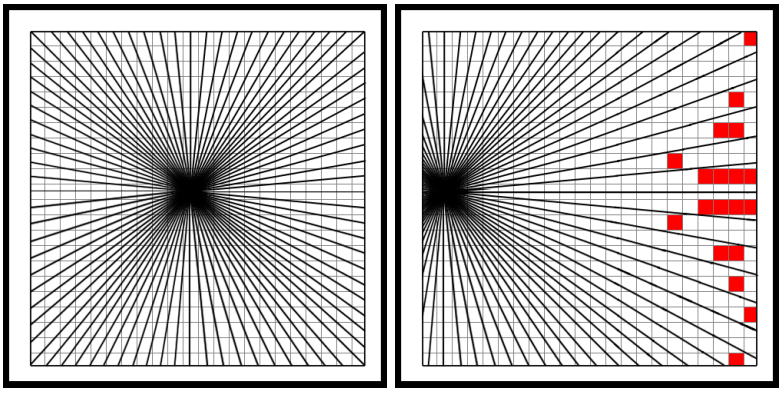


Figur 6: En linje i normalform

Förutsatt att linjens vinkel och positionen för en punkt på linjen är kända, kan man räkna ut avstånd till referenspunkten enligt formeln

$$radie = x * cos(\alpha) + y * sin(\alpha)$$

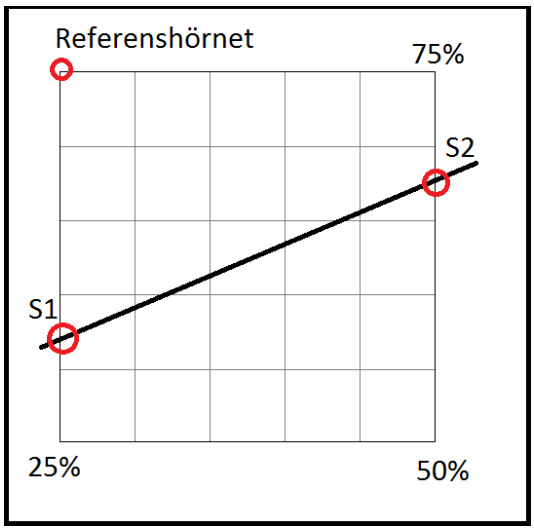
Detta är en relativt enkel ekvation att använda om man vill ha alla linjer som går genom en punkt då man enbart behöver iterera över alla tänkbara vinklar i bilden. Om det är mindre viktigt att ta hänsyn till alla möjliga linjer kan det dessutom vara värt att sänka vinkelantalet i förmån för transformens hastighet. En nackdel är däremot att pixelns position är avgörande för hur många av de skapade linjerna som är överflödiga eller saknas. Exempelvis skulle tillräckligt många vinklar för att skapa alla möjliga linjer genom en pixel i mitten inte vara tillräckligt för en pixel närmare en kant. Notera i figur 7 hur ingen linje traverserar de ifyllda pixlarna trots att vinkeln mellan alla linjer är identisk mellan bilderna och att antalet linjer räcker i första bilden.



Figur 7: Linjerna som behövs är olika beroende på pixelns position.

2.2.2 Räta linjer genom Muffparametrisering

Muffparametrisering är relativt speciell då den använder det faktum att man känner till storleken på ytan linjen går över. Linjen beskrivs i form av skärningspunkterna mellan utgångslinjen och ytans kant relativt till ett referenshörn. Varje punkt beskrivs av varsin skalär som har värdet av avståndet mellan referenshörnet och punkten om man följer ytans kant motsols [5]. Se figur 8. Det finns tydliga fördelar med denna parametrisering när ytan är en bild, det vill säga uppbyggd av pixlar. Då kommer parametriseringen nämligen av sin natur att begränsas till alla linjer som faktiskt kan renderas över bildens pixlar [5].



Figur 8: En Muffparametriserad linje.

Om man jämför den med normalformen ser man att Muffparametrisering inte har problem med att pixelns position påverkar antalet överflödiga eller saknade linjer. Istället har den nackdelen att om man känner till riktningen på linjen i en punkt så är det inte ett enkelt steg för att få linjen Muffparametriserad. Man behöver räkna ut var längs bildens kanter denna linje skulle korsa och dessutom konvertera denna position till formen av avståndet till referenshörnet.

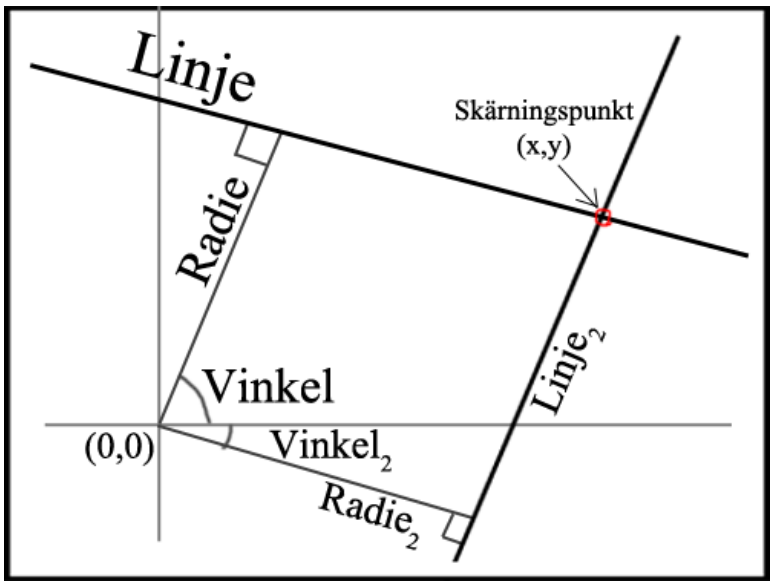
2.2.3 Beräkning av skärningspunkt mellan linjer

När man arbetar med linjer kan det vara användbart att kunna beräkna punkten där de korsar varandra, vilken kan beräknas med matematiska formler. Denna punkt brukar ges i formen av dess kartesiska koordinater i en tvådimensionell vektor längs x- och y-axlarna. Se figur 9. För linjer på nor-

malform beräknas skärningspunkten genom ekvationerna:

$$x = \frac{r_1 * sin(\alpha_2) - r_2 * sin(\alpha_1)}{cos(\alpha_1) * sin(\alpha_2) - cos(\alpha_2) * sin(\alpha_1)}$$

$$y = \frac{r_2 * cos(\alpha_1) - r_1 * cos(\alpha_2)}{sin(\alpha_2) * cos(\alpha_1) - sin(\alpha_1) * cos(\alpha_2)}$$



Figur 9: Två linjer på normalform och deras skärningspunkt.

2.2.4 Tanken bakom transformen och hur den fungerar

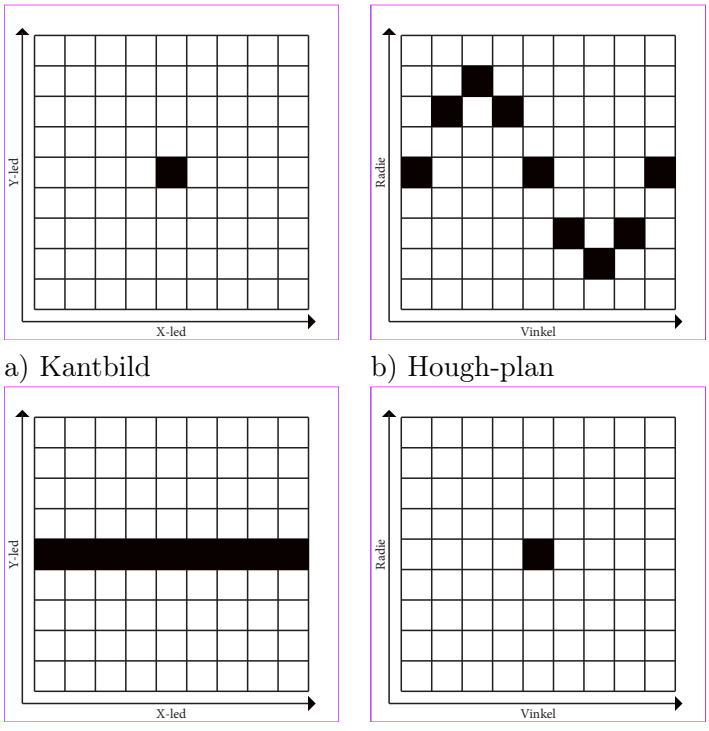
En Hough-transform bygger på ett popularitetssystem och det faktum att det i en kantpixel kan finnas ett antal olika geometriska figurer vars kanter korsar den. När en bild transformeras så itererar algoritmen över alla pixlar i bilden och för varje kantpixel i den identifieras figurerna som innehåller pixeln. Pixeln "röstar" då på varje figur och antalet röster en figur får är därför direkt relaterat till sannolikheten att den existerar.

2.2.5 Hough-planet

Hough-transformen tar en binär bild och transformerar denna till en ny bild där varje pixel istället motsvarar en figur i originalbilden. Denna bild kallas oftast Hough-plan eller parameter space på engelska [5]. Vanligast är att figurerna som Hough-planets pixlar motsvarar är linjer eller cirklar, men alla figurer som går att förklara med två parametrar kan beskrivas i ett Hough-plan. Speciellt intressant är att precis som den engelska termen tyder på

så måste resultatet inte vara ett tvådimensionellt plan, utan transformen kan mycket väl resultera i en rymd med fler dimensioner. Detta innebär att transformen även kan hantera figurer med mer än två parametrar genom att lägga till en ny dimension för varje extra parameter [5]. Problemet med en sådan rymd är att minnesanvändningen som krävs ökar dramatiskt för varje dimension samtidigt som figuren oftast redan går att beskriva som en grupp av enklare figurer.

Det vanligaste Hough-planet representerar alla linjer i bilden [5]. Förutsatt att linjerna beskrivs på normalform motsvarar en pixel i Hough-planet därför en linje (radie,vinkel) där radien är pixelns x-komponent, och vinkeln dess y-komponent. Då transformen fungerar genom att varje pixel i bilden röstar på varje linje som går igenom den, motsvarar en punkt i bilden uppenbart en pixel för varje kolumn i Hough-planet. Mindre uppenbart är att radien inte bara ändras gradvis, utan att den faktiskt ändras så att resultatet blir en sinuskurva i Hough-planet. Se figur 10. I samma figur kan man se att en linje av pixlar resulterar i ett Hough-plan där alla sinuskurvor orsakar ett tydligt maxima i den enskilda pixeln som motsvarar linjen.

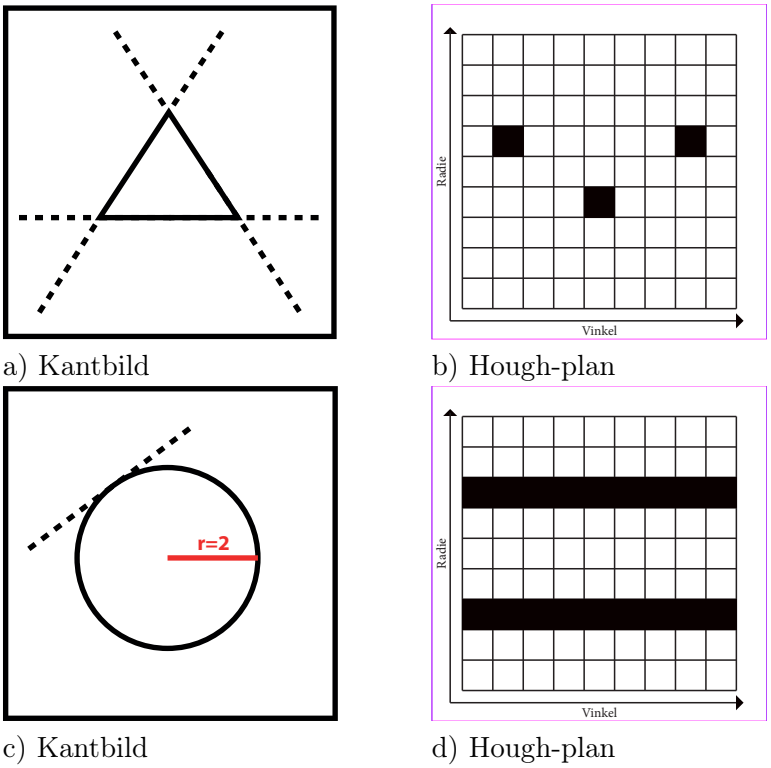


a) Kantbild b) Hough-plan

c) Kantbild d) Hough-plan

Figur 10: Simpla kantbilder och resulterande Hough-plan.

Eftersom transformen fungerar genom att varje kantpixel röstar på en linje, så motsvarar värdet på pixeln i Hough-planet antalet kantpixlar i bilden som tillhör dess linje. Om man därför har en figur i bilden som transformeras kommer de linjer vilka följer figurens kanter bäst att vara de starkaste i Hough-planet. Exempelvis i figur 11 så resulterar triangeln a) i tre maxima för b). Dessa motsvarar de streckade linjerna i a), det vill säga triangelns tangentlinjer. Den centrerade cirkeln c) är däremot lite speciell i det att det finns ett oändligt antal tangentlinjer med olika vinklar. Det resulterande Hough-planets maxima ligger därför i alla pixlar med radien $r = \pm 2$ oavsett vinkel och ser ut som två horisontella linjer. Eftersom bilderna gjordes för hand så stämmer inte de resulterande transformbilderna perfekt.



a) Kantbild b) Hough-plan

c) Kantbild d) Hough-plan

Figur 11: Geometriska figurer och resulterande maxima i deras Hough-plan.

2.2.6 Filtrering av Hough-planet

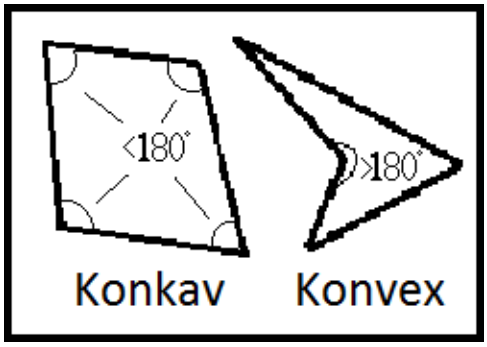
Efter transformen kommer Hough-planet att innehålla ett flertal pixlar med enbart ett fåtal röster. Dessa syftar sällan på en faktisk linje i bilden utan brukar vara ett resultat av att linjerna är oändligt långa och därför kan innehålla kantpixlar från ett flertal olika separata objekt. Det skulle vara möjligt att enbart filtrera med ett tröskelvärde, men det skulle inte filtrera bort linjer som bara delvis följer sin kant om de ändå traverserar tillräckligt många pixlar i den. Detta innebär att längre linjer efter en transform resulterar i en solfjäder av linjer [5].

Istället kan man filtrera för att få fram lokala maxima. Detta innebär att även ett område av starka linjer resulterar i en linje, vilken är den som anses passa kanten bäst. En sådan filtrering skulle däremot innebära att även linjerna som går över ett flertal objekt accepteras. Det följer då på detta att en filtrering med både tröskelvärde och på lokala maxima är att föredra.

2.3 Geometriska egenskaper hos krittavlor

För att en dator ska kunna känna igen ett objekt behöver den utgå från dess geometriska egenskaper. Detta innebär att det är viktigt att man har en god insikt av egenskaperna för objektet man vill arbeta med för att kunna skapa algoritmer som identifierar dem eller arbetar med dem. Nedan beskrivs egenskaperna hos en krittavla då detta är det objekt som är relevant för projektet.

En krittavla är egentligen ett rektangulärt objekt, men vid fotografering av den orsakar perspektivet att bilden representerar den som en skev fyrhörning. Denna fyrhörning är alltid konkav, se figur 12, och eftersom kameran bör vara placerad för att få ett så bra perspektiv som möjligt, går det att förutsätta att dess hörn är någorlunda vinkelräta. Beroende på hur mycket man litar på perspektivet kan vinkelmarginalen sänkas och med det sänka antalet identifierade fyrhörningar som är en potentiell tavla.



Figur 12: Exempel på en konkav och en konvex fyrhörning

I kantbilden kommer dessutom tavlans motstående kanter ha motsatta riktningar, eftersom om det vid ena kanten går från ram till innehåll så kommer det vid den motstående att gå från innehåll till ram [6]. Detta leder också till att riktningen på kanterna har ett samband, där om man traverserar dem motsols så roterar även riktningen motsols [6]. Se figur 13.

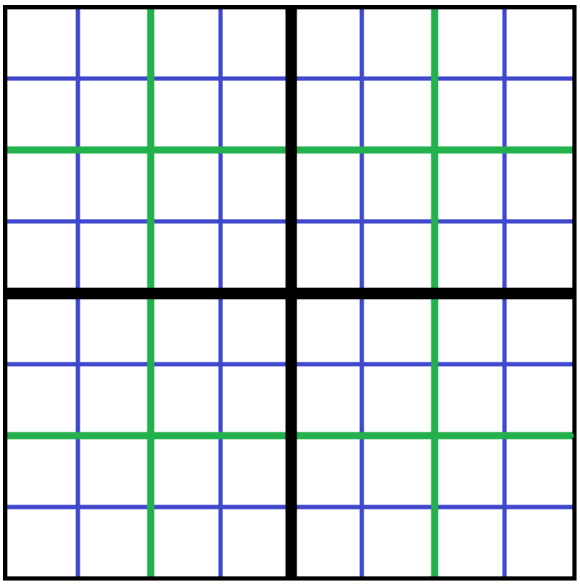
Figur 13: Riktningarna på linjerna roterar med kanten på tavlan.

2.3.1 Krittavlor bland andra rektanglar

Även om man känner till krittavlans geometriska egenskaper så är det troligt att det i en bild existerar fler rektangulära objekt än krittavlor. Detta innebär att det är viktigt att kunna specificera mer unika egenskaper för just krittavlor, vilka datorn kan använda sig av för att skilja på en godtycklig rektangel och en faktisk krittavla. Detta kan göras genom att iaktta ett typexempel av hur krittavlan skulle se ut i bilden och approximera dess storlek samt bredd- och höjdförhållande. Storleken kan bero väldigt mycket på avståndet även för en och samma tavla, men förhållandet kommer hållas relativt konstant från bild till bild. De rektanglar med tillräckligt lika värden kan då troligtvis anses vara en krittavla, om man förutsätter att bilden innehåller någon överhuvudtaget.

2.4 Rekursivt hierarkiskt rutnät

Ett rekursivt hierarkiskt rutnät är ett rumsligt symmetriskt dataträd där varje nod representerar en yta. Trädens noder innehåller sitt egna dataträd av samma typ. Hur många rutor det är per trädnivå och hur många nivåer som ska finnas bestäms vid implementation och kan variera stort. Detta kan exempelvis användas för att jämföra två olika bilders likhet. [7] Se figur 14.



Figur 14: Ett rekursivt hierarkiskt rutnät med en rutnätsstorlek på 2x2 och 3 nivåer.

3 Metod

I detta avsnitt beskrivs arbetssättet och förstudier till de olika designvalen som gjordes.

3.1 Förstudie

För att avgöra vilka lösningar som skulle passa bäst för problemen gjordes en förstudie genom att söka efter information på Chalmers bibliotek och andra källor på internet. Bildanalys krävde en mer omfattande förstudie än de andra delarna på grund av att det är ett komplicerat ämne som vi inte hade någon tidigare erfarenhet av innan projektets start.

3.2 Arbetssätt

De första veckorna diskuterade och bestämde vi hur produkten ska fungera i största möjliga utsträckning. Det beslutades om alla stora designval såsom vilken hårdvara som skulle användas och hur flödet från krittavlan till att den kan visas på slutanvändarens webbläsare skulle vara. Detta så att vi skulle kunna dela upp projektet i mindre delar så arbetet och ansvarsområden kunde fördelas mellan alla gruppmedlemmar.

Vi hade möten veckovis där vi diskuterade designval, om vi lyckades med de målen som sattes upp föregående vecka och nya mål stakades ut för nästkommande vecka. Metoden vi använde för att sätta upp veckomål är en typ av Scrum. Scrum är ett sätt att strukturera och sätta upp mål som ska ha uppnåtts till nästa milstolpe.

Vi använde oss av Git för att spara och hantera vår programkod. Git är ett versionshanteringssystem för programmering.

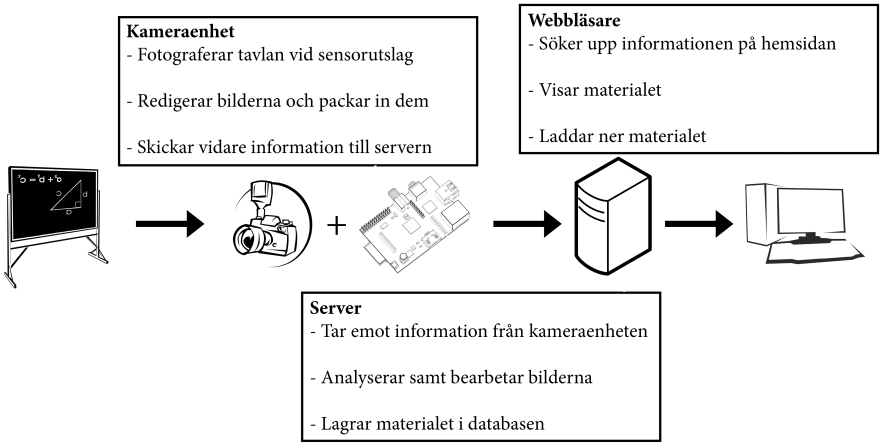
4 Design och implementation

Detta kapitel kommer att gå igenom vilka designval som gjordes, hur de olika designvalen implementerades och hur prototypen blev i slutändan. Början av detta kapitel går överskådligt igenom systemflödet och de senare delarna i kapitlet går igenom de olika programdelarna i mer detaljerat.

4.1 Grundflödet för systemet

För att lösa problemet med att ta kort på en krittavla i en föreläsningssal valde vi att placera en enkortsdator i varje föreläsningssal och ansluta en kamera per kolumn av krittavlor till datorn. En givare är placerad vid varje krittavla för att kunna känna av tavlans position som sedan används för att kunna bestämma om den är färdigskriven eller inte. En tavla anses vara färdigskriven om den är i sitt toppläge. Detta valdes på grund av att vi har uppmärksammat att det är vanligt bland föreläsare att skicka upp tavlan till sitt toppläge när de är klara med den. En enkortsdator fungerar som en vanlig stationär dator med skillnaderna att den är betydligt mindre, har sämre prestanda och förbrukar mindre mängd ström. Enkortsdatorn kommer refereras som kameraenhet från och med nu.

Efter att kameraenheten har bearbetat föreläsningsanteckningsbilden överförs bilden till en central server som alla kameraenheter är anslutna till. På servern sparas föreläsningsanteckningsbilden i en databas med all relevant information såsom kurskod, vilken tid bilden fotograferades med mera. Servern hanterar också webbsidan som slutanvändarna ansluter till för att kunna ta del av föreläsningsanteckningsbilderna. Se figur 15 för flödet.



Figur 15: Överblick på hela flödet från kameraenhet till servern

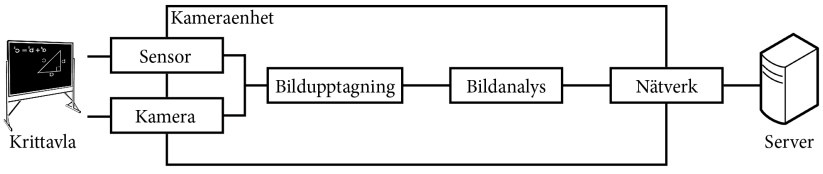
4.2 Överblick över kameraenhetens programflöde

Systemet på kameraenheten är uppdelat i tre delar. Dessa tre delar är ett program som hanterar kameror och givare, ett program som analyserar bilder och ett program som hanterar all nätverkskommunikation. Se figur 16 för kameraenhetens programflöde. För att få hela systemet så skalbart som möjligt valdes det att placera så många av uppgifterna som möjligt på kameraenheten. Detta för att om en skola skulle vilja installera exempelvis 100 kameraenheter till ska de inte behöva uppgradera sin server. På kameraenheten valdes det att placera all bildbehandling som innefattar endast en föreläsningsanteckningsbild. Bildbehandling som endast innefattar en föreläsningsanteckningsbild är skevhetskorrigering och att skära ut krittavlan ur en föreläsningsanteckning så att endast krittavlan finns med på föreläsningsanteckningsbilden.

För att kunna hitta bland annat kurskod för en föreläsningsanteckningsbild ansluts kameraenheten till skolans webbsida för schemavisning. Denna uppgift är placerad på kameraenheten för att spara på bandbredden och processorkraft åt servern.

Programmet som analyserar givarutslag och använder kameran (Kamerakontrollern) är skrivet i C++ såväl som bildbehandlingsprogrammet. Nätverksprogrammet är skrivet i Java och det som omfattas av nätverksprogrammet är att överföra filer till servern, kommunicera med skolans webbsida för schemavisning och att starta kamerakontrollern för att inte kameraenheten ska ta kort på krittavlor när en föreläsning ej är aktiv.

Programmen kommunicerar med varandra genom att skicka vidare föreläsningsanteckningsbilder till en mapp som skannas av det mottagande programmet. Exempelvis när kamerakontrollern har tagit ett kort så skickar den föreläsningsanteckningsbilden till bildbehandlingsprogrammets mapp som kontinuerligt skannas efter nya bilder.



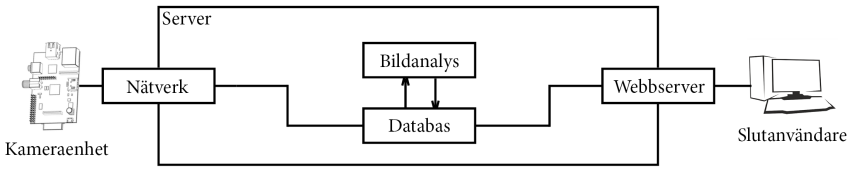
Figur 16: Överblick över programflödet på kameraenheten

4.3 Överblick över serverns programflöde

Efter att servern har mottagit en föreläsningsanteckningsbild packas bilden upp och sparas i en fördefinierad mapp. Samtidigt uppdateras databasen med den nya föreläsningsanteckningsbilden. Se figur 17 för programflödet på servern.

Bildbehandling som är placerad på servern är bildjämförelsen vilken innefattar flera föreläsningsanteckningsbilder. Det är bildjämförelsen som filtrerar ut föreläsningsanteckningsbilder som är väldigt lika varandra. Detta är på grund av att vi valde att kameraenheten ska arbeta med alla sina uppgifter kontinuerligt och så snabbt som möjligt och inte bli begränsad av att en föreläsning ska bli klar innan kameraenheten får skicka över föreläsningsanteckningsbilderna till servern.

Därutöver finns en webbplats tillgänglig för internetåtkomst, som i sin tur kommunicerar med databasen och presenterar föreläsningsanteckningsbilder och annan relevant information för slutanvändaren.



Figur 17: Överblick över programflödet på servern

4.4 Hårdvara

För att systemet ska kunna utföra sin uppgift krävs en del hårdvara. I varje sal som är ansluten till systemet ska det finnas en enkortsdator ansluten till kameror och givare för att ta kort på tavlor vid rätt tillfälle. Det här delkapitlet beskriver all den hårdvaran som systemet är uppbyggt av.

4.4.1 Kameraenheten

Kameraenheten består av en Raspberry Pi kopplad till en givare per tavla samt ett kretskort med tre kameror. En Raspberry Pi är en enkortsdator med en 700MHz ARM-processor, 512MB RAM-minne och en Broadcom VideoCore IV GPU[9]. Se figur 18. Det finns flera alternativ till en Raspberry Pi som exempelvis andra enkortsdatorer och Arduinokretsar. De största fördelarna med en Raspberry Pi är att det är en komplett dator och kan således köra ett vanligt operativsystem utan problem, samt att den är rela-

tivt billig[10][11]. Det finns dessutom 26 lättåtkomliga GPIO-pins som i det här systemet används till avläsning av givare, tända status-LEDs och skicka styrsignaler till kamerakretsen. Det finns ett antal olika operativsystem att välja på till en Raspberry Pi. Eftersom Raspberry Pi:n har begränsad prestanda var det naturligt att välja ett så lättviktigt operativsystem som möjligt. Operativsystemet som valdes blev därför Arch Linux ARM som inte har något grafiskt användargränssnitt och är resurssnålt samt direkt anpassat för Raspberry Pi:n. Dessutom är det lika enkelt att kompilera och köra program skrivna i olika språk som på vilken annan dator som helst[12].

Figur 18: Raspberry Pi[8]

Varje kameraenhet är designad att hantera tre stycken kameror då föreläsningssalar kan innehålla upp till tre kolumner av tavlor. För enkelhetens skull används kameror av typen Raspberry Pi Camera Module till varje Raspberry Pi eftersom de är byggda för att vara kompatibla och det finns färdig programvara för dessa. Det finns dock ett problem med dessa kameror: en Raspberry Pi har bara en kontakt för att ansluta en kamera. Det finns flera sätt att lösa detta problem på. En tanke var att använda en ensam kamera och montera den på en motor som då kan vinkla kameran mot rätt kolumn med tavlor. Detta alternativet valdes dock bort för att undvika så många rörliga delar som möjligt i konstruktionen och på så sätt också minska det underhåll som systemet behöver. Det slutgiltiga valet blev att vi tillverkade ett eget kretskort med tre kontakter och med hjälp av en FPGA väljs vilken kamera som ska vara aktiv.

4.4.2 Kamerakrets och multiplexer

De tre kamerorna kopplades till ett kretskort där signalerna skulle multiplexas för att få dataströmmen från rätt kamera. Med rätt signal från Raspberry Pi:n öppnar multiplexern kanalen till den kamera vars motsvarande giva-

re har gett utslag. Denna multiplexer var primärt tänkt att programmeras på en CPLD. Problem uppstod däremot när det uppdagades att Raspberry Pi-kameror använder sig av MIPI-protokollet för dataströmmarna, just i Raspberry Pi Camera används Serial Interface CSI-2 som är en av MIPI standarderna[30]. Dessa signaler har variabla spänningar vilket en CPLD inte kan hantera[31]. Därför gick fokus över till att använda en FPGA som har möjligheten att hantera just variabla signaler. Denna förändring påverkar varken pris eller storlek i någon högre utsträckning och sågs därför som ett bra alternativ.

4.4.3 Givare för kameraenheten

För att avläsa om en krittavla i en föreläsningssal är uppe eller nere diskuterades det främst om två olika alternativ. Det ena alternativet var att låta kameraenheten köra ett bildanalysprogram i realtid under hela föreläsningarna som skulle kunna följa en tavlas position. För att avgöra vilken tavla som är vilken och om den är uppe eller inte hade vi tänkt sätta ett klistermärke på varje tavla och låta programmet leta efter de klistermärkena. Vi valde dock bort den här lösningen då en Raspberry Pi har begränsad prestanda och vi anser att användning av givare skulle vara en stabilare lösning.

När vi bestämt oss för att använda givare och gjorde en undersökning i ämnet visade sig läsgafflar vara det lämpligaste valet till detta system då de är mycket billiga och enkla att använda. En läsgaffel är, som namnet antyder, gaffelformad med två "armar" och en glipa mellan. Se figur 19 På ena armen sitter en IR-diod som lyser på den andra armen som i sin tur innehåller en fototransistor. Fototransistorn släpper igenom ström när IR-dioden lyser på den men kretsen blir bruten om något täcker för IR-dioden. På så sätt kan man avläsa en hög signal när en tavla är nere och en låg signal när tavlan är i topposition. Det används en läsgaffel per krittavla.

Figur 19: Läsgaffel

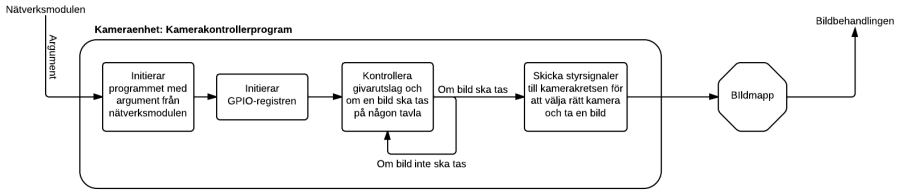
Vi hade två funderingar på hur vi skulle ansluta läsgafflarna till kameraenheten. Det första alternativet var att göra det trådlöst. Tanken var att i så fall först ansluta givarna till ett kretskort med kablar och sedan därifrån skicka signalerna trådlöst till kameraenheten. Det här alternativet valdes bort av framförallt två anledningar: dels för att priset per föreläsningssal skulle gått upp för mycket och dels för att vi anser att överföringen blir mindre stabil än med trådad anslutning hela vägen. Det andra alternativet, vilket också var det som valdes i slutändan, var att använda en trådad anslutning hela vägen från givarna till ett kretskort och sedan vidare till kameraenheten. Kretskortet består av tre lysdioder för att visa programstatus, signalpins för varje givare, pins för två olika spänningar och pins för jord.

4.4.4 Mjukvara för kamerornas styrprogram

För att kunna avläsa givarna vid krittavlorna används Raspberry Pi:ns GPIO-pins. För att kunna hantera GPIO-pinsen och deras register i programkoden behövs ett bibliotek. Det finns ett flertal olika bibliotek att välja på beroende på vilket språk man väljer att skriva programmen i. Då styrprogrammet på kameraenheten programmerades i C++ hittades ett lämpligt C-bibliotek för GPIO-hantering, bcm2835, som tillåter åtkomst till GPIO-registren[13].

Själva kamerakontrollerprogrammet har tre huvudsakliga uppgifter: att läsa av givarna för att bestämma position av tavlorna, kolla huruvida en tavla täcker en annan och ta kort på en tavla när så ska ske. Programmet kan startas med argument för att bestämma hur många kolumner med tavlor det är i den aktuella salen, hur många tavlor det är i varje kolumn, hur länge programmet ska köras, om det ska köras i debug-läge eller om det ska köras i testläge. Om inga argument anges initieras värdena till tre kolumner av tavlor med tre tavlor i varje och en körtid på 120 minuter.

Programmet börjar med att initiera GPIO-registren och alla värden utifrån argumenten och börjar därefter läsa av givarna i en kolumn. Därefter avgörs om en bild ska tas på någon av tavlorna i kolumnen. För att en bild ska tas måste vissa kriterier uppfyllas: en tavla måste vara i sitt toppläge, men en bild kommer inte tas på den om det redan gjorts utan att tavlan varit nere emellan, och ingen annan tavla får täcka över tavlan. Om en bild ska tas skickar programmet styrsignaler till multiplexern på kamerakretsen för att välja rätt kamera och skickar sedan signaler till den valda kameran för att ta kort. Efter att programmet har avgjort om en bild ska tas eller inte går det vidare till nästa kolumn med tavlor och återupprepar samma algoritm där. Programmet loggar alla större händelser i programflödet i en loggfil så att det är enkelt att kolla om någonting gått fel under en körning.



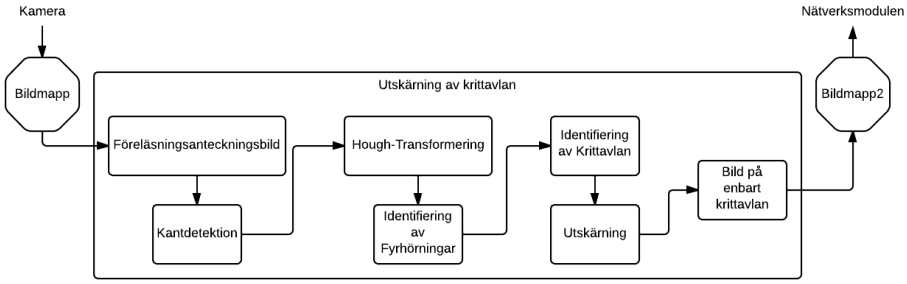
Figur 20: Programflödet för kamerakontrollern

4.4.5 Serverns hårdvara

Servern skall vara installerad med ett Linux/Unixbaserat operativsystem.

4.5 Bildbehandling

Systemet använder bildbehandling för att skära ut och skevhetskorrigera tavlan samt för att jämföra de färdiga bilderna med varandra. Bildanalysen innefattar kantdetektion, Hough-transform, identifiering av potentiella tavlor, gallring för att hitta den bästa kandidaten och utskärning av denne. Bildjämförelsen kollar om de färdiga bilderna är tomma eller ej samt jämför de färdiga bilderna med varandra. I de fallen den är tom eller tillräckligt lik en annan bild, slängs den först inkomna bilden.



Figur 21: Överblick över programflödet för utskärningen av krittavlan.

4.5.1 Utskärning av krittavlan

För att kunna identifiera och skära ut krittavlan från bilden utförs först en kantdetektion på bilden. Den kan behöva ställas in beroende på sal och ljus-

sättning, men senare algoritmer har visat sig ha förlåtelse till om onödiga kanter detekteras och det är bara deras prestanda som försämras. Den algoritm som fungerar bäst för identifiering av tavlan är Kirsch malldetektion. Tester visade på att användandet av enbart bildens gråskala fungerade bra, då det inte märktes någon skillnad på slutresultatet av utskärningen jämfört med om även de tre färgkanalerna (röd-, grön-, blå-kanalerna) användes. En färgkanal i bilden är en skalär per pixel för att beskriva magnituden av en färg och gråskalan är då en kanal som summerar de tre färgkanalerna (röd, grön och blå) och beskriver bilden i svartvitt. En fördel med att endast köra gråskalan är att det bara tar en fjärdedel av tiden det skulle ta att använda de tre färgkanalerna och gråskalan. De algoritmer som undersöktes som alternativ var Canny och en kantdetektionsalgoritm kallad ISEF från boken Algorithms for Image Processing and Computer Vision (Andra upplagan)[2]. ISEF fungerade dåligt med alldeles för mycket brus och otydliga linjer. Den tid som skulle krävas för att få denna algoritm att fungera väl ansågs för betydande på grund av dess komplexitet. Canny fungerade bra med betydligt smalare kanter än Kirsch men processen tog längre tid. Resultatet hade fler hål i kanterna men dessa var försumbara och hade försvunnit efter en Hough-transform.

En Hough-transform utförs på den resulterande kantbilden. Det är en snabb och användbar transform som används i liknande arbeten för att identifiera rektanglar[6]. Resultatet filtreras för att hitta endast de långa och tydliga linjerna. Oftast tillhör dessa linjer mestadels tavlan, tavlans räls och lamporna ovanför. Detta resultat är starkt beroende av sal och ljussättning. Systemets Hough-transform utgår från mitten av bilden som (0,0) och transformerar detta till ett hough-plan som representerar linjer i normalform inom intervallet $(-\pi, -r)$ till (π, r). Detta intervall används eftersom det innebär att alla linjer kan beskrivas två gånger, med motsatt riktning för varje linjepar.[6] Bildens kanter har nämligen riktningar, vilka används som bas för att de resulterande linjerna ur Hough-transformen ska ha rätt riktning.

Anledningen till att linjernas riktning har så stor betydelse är på grund av de geometriska egenskaperna hos en krittavla. Linjerna används till att identifiera alla potentiella krittavlor, vilket sker genom att hitta alla kombinationer av linjerna som följer vissa kriterier. Varje kombination ska vara en grupp om exakt fyra linjer där de motstående linjerna har motsatta riktningar med ett visst avstånd mellan varandra och intilliggande linjer är vinkelräta mot varandra. Varje kombination läggs då in i en lista i form av krittavlans fyra hörn identifierade genom att hitta skärningspunkterna mellan de intilliggande linjerna.

Listan med potentiella tavlor gås igenom för att identifiera den bästa kandidaten. Detta sker genom att först följa dess motsvarande kanter i kantbilden

och filtrera bort alla kandidater med för många hål efter varandra eller för många hål relativt mängden pixlar längs kanten. Detta är för att en linje från transformen är oändlig och kan ha skapats från kantpixlar tillhörande vitt skillda objekt. När listan har filtrerat bort de klart felaktiga kandidaterna itererar man över de kvarvarande och ger dem en vikt beroende på hur bra de matchar ett typexempel på en krittavla i bilden. Kandidaten med det bästa resultatet accepteras då som den faktiska krittavlan.

Nu när tavlans hörn har hittats i bilden kan själva utskärningen börja. Utskärningen går till så att tavlan delas upp i två trianglar med delad hypotenusa och pixlarna i hypotenusan identifieras och sparas till senare användning. Sedan samplas längs en linje från det motstående hörnet i en av trianglarna till en av punkterna i hypotenusan, vi kallar detta för en samplingslinje. Detta görs för alla punkter i hypotenusan och likadant för den andra triangeln. Sedan sätts en tom bild upp med den slutgiltiga och symmetriskt korrekta storleken som delas upp i två trianglar på samma sätt som tavelbilden. Även här hittas alla pixlar i hypotenusan. Nu tar vi samplingslinjerna från tavlan och använder dessa för att återskapa tavlan utan några skeva kanter i den tomma bilden. Vi går från motstående hörn till en pixel i hypotenusan på samma sätt som när vi samplade, fast nu använder vi en av samplingslinjerna för att istället föra in pixlar till den tomma bilden. Vilken samplingslinje som ska användas beror på hur stor tavelbildens hypotenusa är i jämförelse med slutbildens hypotenusa. Den första samplingslinjen är den första som slutbilden använder och den sista samplingslinjen är den sista som slutbilden använder, interpolering bestämmer vilka som ska användas där emellan. När en pixel stöts på vid traversering från det motstående hörnet till pixeln i hypotenusan så tar vi tre sampels från samplingslinjen och använder dessa för att ge pixeln ett färgvärde. När alla pixlar i slutbildens hypotenusa har blivit besökta från båda motstående hörnen så har nästan alla pixlar i slutbilden blivit ifyllda. Det finns ofta några pixlar som inte blivit besökta men deras fyra närmsta grannar (upp, ner, vänster och höger) har alltid fått ett värde. Så de obesökta pixlarnas värde blir en interpolering av deras grannars värde.

4.5.2 Tavelanalys

I vår realisering av uppgiften har vi två invarianter av överflödiga bilder som behöver angripas: tomma bilder och multipla bilder på samma tavla. De här bilderna är inte efterfrågade och ska inte sparas undan då de inte ger någon vidare information till studenterna. Ytterligare ett problem som kan ha uppstått är när en tavla blir förändrad under föreläsningen. Det kan vara för att en föreläsare kanske lägger till något på tavlan efter tavlan har blivit uppskjuten och fotograferad. Då vill vi spara undan den nytagna bilden istället för den gamla, eftersom den nya bilden har den senaste informationen. Ett delprogram som kan identifiera tomma tavlor och jämföra tavlor med

varandra behövs för att kunna hantera dessa överflödiga bilder.

För att kunna analysera en bild på ett bra sätt behöver bilden bli kantdetekterad. Detta ger en binär bild, ett format som kan användas till att jämföra olika tagna foton gentemot varandra. Utan kantdetektering blir det svårt att hitta kontrasterna som avgör likhet med en annan tavla då faktorer som olika ljusförhållanden kan få även de mest lika tavlor att tolkas olika. Efter att vi fått en binär bild vill vi filtrera ut datan över flera pixlar så att två liknande bilder ska kunna ha några pixlars marginal när vi jämför dem med varandra. Detta för att en och samma tavla kan bli fotograferad med lite variation i dess läge, vilket kan ge en förskjutning av tavlans linjer med några enstaka pixlar.

Vi vill nu se bilden som en procentuell jämförelsebild, Percentage Comparison Image (PCI), där en pixel som ligger på en kant har ett värde på 100% och en pixel som inte ligger i på en kant får ett värde mellan 0%-100% beroende på avståndet från närmaste kant. I vårt fall startar alla pixlar som inte är på en kant på värdet 0%. Vi komprimerar ner bildens höjd och bredd genom att halvera dem, så att fyra pixlar blir en pixel och sparar deras genomsnittsvärde i den nya pixeln. Sedan lägger vi på ett litet gaussfilter som ytterligare sprider ut pixlarnas värde.

Ett rekursivt hierarkiskt rutnät skapas nu där PCI-bildens värden läggs in där trädets löv minst har hand om 10x10 pixlars värde. Löven i trädet håller i summan av de pixlars värde som ligger inom lövens respektive områden. De andra noderna i trädet håller hand om summan av nodernas barnnoder. Trädets rot har alltså summan av alla pixlar i PCI-bilden.

Nu kan vi jämföra summan i quadträdets rot med ett tröskelvärde för tomhet och om summan inte når upp till tröskelvärdet så ser vi tavlan som tom och tar då bort bilden på tavlan.

För att jämföra om två bilder är lika varandra eller identiska så behöver varje nivå av bildernas rutnät vara tillräckligt lika. Bilderna jämförs nu med hjälp av ett tröskelvärde för att se om bilderna är tillräckligt lika varandra. Vi kollar först i trädens rötter och ser om de är tillräckligt lika där, om så är fallet så jämförs rötternas barnnoder om även de tillräckligt lika. Det här upprepas rekursivt tills båda trädens löv har blivit jämförda. Om alla nivåer har varit tillräckligt lika så säger vi att bilderna är identiska och så sparar vi den senare bilden samtidigt som den gamla bilden slängs. I annat fall är bilderna inte lika varandra och då sparar vi båda bilderna.

4.5.3 Bortvalda implementationer

En version av Kirschdetektorn kördes över bildens tre färgkanaler och gråskala. Detta innebar att exempelvis kanten mellan ett grönt område och ett rött område av samma magnitud detekteras. I gengäld tog den ungefärligen fyra gånger så lång tid på sig (se bilaga 5 och 6. När implementationen testades på bilder av krittavlor märktes däremot att resultatet inte gav bättre kanter för krittavlan, vilket innebär att den extra kostnaden för denna version var helt onödig.

Cannydetektorn implementerades enligt beskrivningen i den teoretiska och tekniska bakgrunden. Vi lyckades inte riktigt att få till den som vi ville. Det blev lite mer brus än vad vi hade hoppats. Dock så var de kanter som hittades alltid en pixel tunna vilket var väldigt bra. Det som fick oss att välja bort implementationen från prototypen var dess hastighet i jämförelse med vår valda detektor, Kirschdetektorn.

ISEF implementerades som beskriven i källkoden från boken Algorithms for Image Processing and Computer Vision (Andra upplagan)[2]. Resultatet innehöll mycket brus i kantbilden och stora hål på de riktiga kanterna. Även efter att ha testat ett flertal olika värden på dess inställningar så lyckades vi aldrig få bättre resultat och implementationen ansågs därför vara oanvändbar. Algoritmen borde ha fungerat bättre än vad den gjorde och felet låg troligtvis i hur den implementerades, men vi valde att inte arbeta vidare med den då den kändes för komplicerad och tidskrävande.

Ett försök att förenkla kantdetektionen var att skapa ett eget filter för att endast ha kvar de färgerna som tavlan innehåller. Det skulle gjort så att tavlans kanter skulle vara bland de enda kanterna som hittades. Implementationen av det här misslyckades då de kanter som hittades efter filtret var för fransiga, vilket skulle resulterat i att tavlans kanter enligt Hough-transformen skulle blivit fler och svagare.

4.6 Nätverk

Det som innefattas i nätverksmodulen är delarna som har hand om överföring av föreläsningsanteckningsbilder från kameraenheten till servern, samt insättning av föreläsningsanteckningar i databasen. Nätverksmodulen hanterar även att hämta information från skolans webbsida för schemavisning och att starta kamerakontrollprogrammet 15 minuter innan föreläsningsstart. Nätverksmodulen skriver även allt den gör till en loggfil och om debugläget är aktivt skrivs mer detaljerad och mer information till loggfilen.

4.6.1 Nätverksdesign

För att kunna skicka filerna mellan kameraenheten och den centrala servern används protokollet SCP. SCP är baserat på "Secure shell" (SSH) protokollet. SCP är ett säkert och enkelt sätt att skicka filer från en dator till en annan dator[14].

De olika sätten att skicka filer som har jämförts är SCP, FTP och att skriva koden själv med hjälp av programmeringsspråken Java eller C++. Fördelen med SCP jämfört med FTP är att SCP är enkelt att installera, det behövs inte göras några inställningar vid installationen, det är väldigt enkelt att använda och det behövs inte öppnas några portar i en eventuell brandvägg. För att servern inte ska bli överbelastad med allt för många överföringar samtidigt från kameraenheterna, kan en kameraenhet maximalt skicka en föreläsningsanteckningsfil till servern i taget. Det finns en möjlighet att kameraenheten kan skicka en föreläsningsanteckningsfil och en inställningsfil samtidigt till servern.

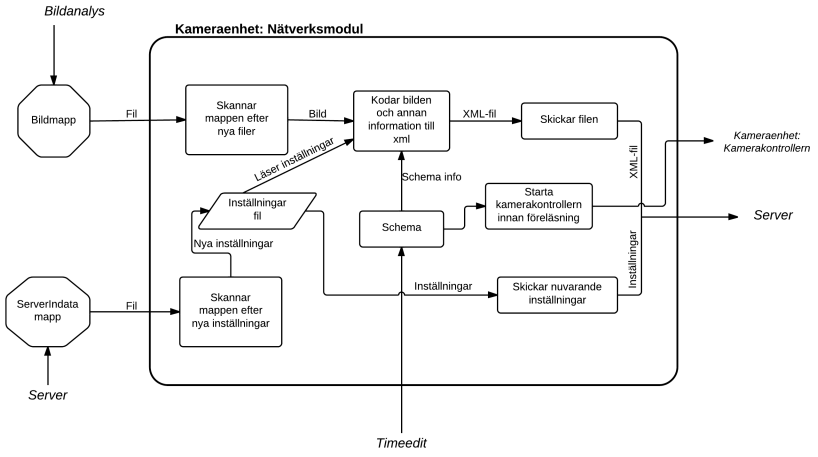
Från början var tanken att skriva koden som skickar filer själv. Det stod mellan att skriva nätverkskoden i Java eller C++. Vi valde att skriva i Java på grund av att det är smidigare än att skriva i C++. I C++ var man tvungen att själv dela upp filen som ska skickas i små paket, medan det i Java fanns färdiga lösningar för detta. Nackdelen med Java är att det exekveras ofta långsammare än C++, men under projektets gång ändrades valet att skicka filer till servern med hjälp av SCP [15]. Detta var på grund av att SCP är stabilare och säkrare än att skriva koden själv. Som en följd av detta försvann argumenten för att välja Java. Dock användes Java i alla fall för att vi redan hade börjat skriva kod i Java. Fördelen med att skriva den själv är att det hade varit mer lärorikt.

Det bestämdes för enkelhets skull att spara all data så som föreläsningsanteckningsbild, kurskod och annan relevant information i en och samma fil som sedan skickas till servern. Till detta valdes det att använda strukturspråket XML på grund av att det är lättläst och har en enkel uppbyggnad. Nackdelen med XML är att den innehåller mycket metadata och att föreläsningsanteckningsbilderna allokerade cirka 33% mer utrymme när bilden ska sparas i XML. För att omkoda en föreläsningsanteckningsbild till XML-format så läses små paket från en föreläsningsanteckningsbild för att sedan skrivas i små paket till XML-filen en i taget, för att spara på kameraenhetens RAM-minne. Nackdelen är att processorn och hårddisken tilldelas mer av arbetet. För att läsa från XML till föreläsningsanteckningsbild läses hela bilddatan från XML in till RAM-minnet för att sedan avkodas till en föreläsningsanteckningsbild. Detta är på grund av att servern inte har samma begränsning av RAM-minne. Servern är istället mer begränsad av processorn och hårddiskarnas snabbhet.

För att kunna hitta information om varje föreläsningsanteckningsbild, såsom kurskod och kurstitel, används TimeEdit. TimeEdit är Chalmers schemawebbsida för att visa visa scheman för de olika föreläsningssalarna. För att hitta informationen laddas HTML-filen för en viss föreläsningssal ned från TimeEdit. En HTML-fil är själva filen vilken innehåller all information som behövs för att visa en webbsida i en webbläsare. Denna HTML-fil söks igenom för att hitta relevant information och skriver informationen till XML-filen som ska skickas till servern. Detta är inte en optimal lösning på grund av att TimeEdits hemsida kan förändras över tid på grund av exempelvis uppdateringar. TimeEdit har ett API, dock behövs en licens för att få tillgång till det. Vi bestämde att detta inte var vårt huvudmål med projektet då det kostade för mycket. Programdelen för att hitta informationen via TimeEdit valdes att läggas på kameraenheten på grund av att minska belastningen av den centrala servern.

4.6.2 Nätverket på kameraenheten

Flödet för nätverket ses i bilden nedan:



Figur 22: Nätverksmodulen på kameraenheten

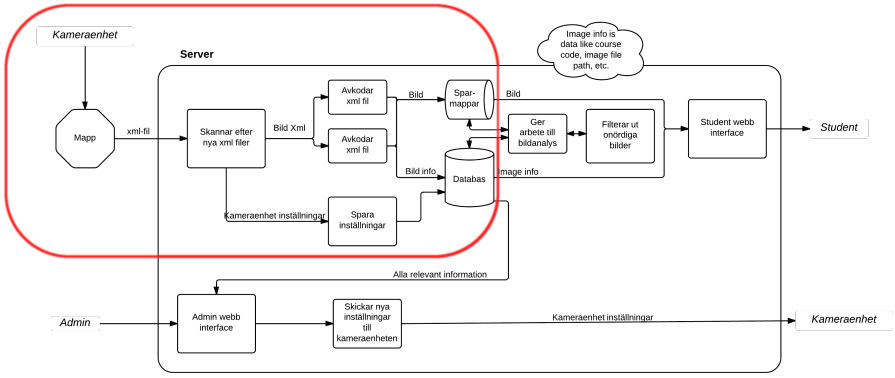
Grundflödet för en föreläsningsanteckningsbild är följande: först söks mappen som nya föreläsningsanteckningsbilder anländer till igenom. När sedan en föreläsningsanteckningsbild har hittats börjas XML-filen att kodas. Först söks inställningsfilen igenom för att hitta kameraenhetens ID och vilken fö-

reläsningssal som kameraenheten är stationerad i. Sedan hämtas kurskod, kursens namn och vilken tid föreläsningen startar och slutar från TimeEdit. Om inte någon kurs hittas för en föreläsningsanteckningsbild så raderas bilden för att den togs vid ett tillfälle när ingen officiell föreläsning hölls. Sist kodas föreläsningsanteckningsbilden in till XML-filen. När all information har sparats i XML-filen så skickas den till den centrala servern. Kameraenheten försöker skicka XML-filen till servern tills den lyckas. Efter ett antal försök läser kameraenheten in inställningsfilen igen för att se om några inställningar så som serverns IP-adress har ändrats och provar att skicka igen till servern med de nya inställningarna.

Kameraenheten kan ta emot nya inställningar från servern såsom serverns IP-adress och lösenord. Det fungerar på så sätt att servern kan skicka en XML-fil som innehåller nya inställningar till kameraenhetens mapp som söks igenom efter nya XML filer. Efter att kameraenheten har hittat den nya inställnings XML-filen från servern uppdateras kameraenhetens inställningsfil. Efter att kameraenhetens inställningsfil har uppdaterats skickar kameraenheten sina nya inställningar till servern. Kameraenheten skickar också sina inställningar vid uppstart och vid klockan tre på natten. Detta för att hålla servern uppdaterad med vilka inställningar kameraenheterna har. Inställningar som skickas är bland annat vad kameraenheten har för IP-adress och lösenord. Det är relativt enkelt att bygga ut systemet för att kunna ta emot andra typer av XML-filer som innehåller annan information för att utföra andra uppgifter på grund av enkelheten i XML.

4.6.3 Nätverket på servern

På samma sätt som tidigare skickas filerna till en mapp som kontinuerligt genomsöks av servern efter nya XML-filer från kameraenheterna. Om den inkommande filen är en föreläsningsanteckning så avkodas föreläsningsanteckningsbilden och sparas på korrekt position i serverns filsystem. Databasen uppdateras med informationen som är levererad från föreläsningsanteckningsfilen. Den informationen som sparas i databasen är föreläsningsanteckningsbilden, vilken kurs samt föreläsningen den tillhör med mera. Om den inkommande filen är en inställningsfil så uppdateras databasen med informationen för den korrekta kameraenheten. Information som uppdateras är exempelvis kameraenhetens IP-adress och lösenord.



Figur 23: Nätverks delen på servern

Avkodningsdelen för inkommande filer på servern är flertrådad och grundinställningarna är att skapa fyra avkodningstrådar. Det rekommenderas att minst ha dubbelt så många avkodningstrådar som servern har processorkärnor. Detta för att minska tiden som trådarna väntar på olika delar såsom att filen ska ha överförts från kameraenheten till servern eller att få tillgång att sätta in information i databasen.

4.7 Webb och Databas

Databasen skapades för att lagra information om bilderna och dess relation med kurser, föreläsningar och kameraenheter. Webbservern hanterar inkommande anslutningar från en HTTP-adress som skulle kunna vara till exempel http://www.tavlan.se via en webbläsare.

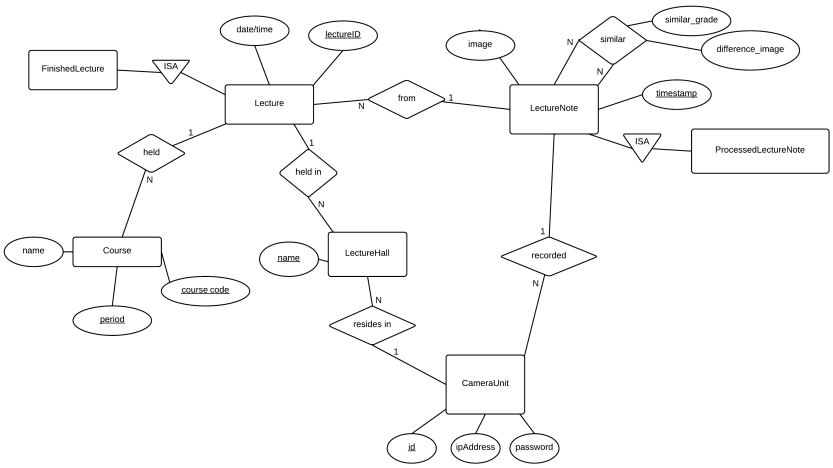
Genom detta gränssnitt kan en student logga in för att visa och ladda ned föreläsningsanteckningsbilderna som residerar i databasen. Dessutom kan en administratör logga in på samma sätt men då även ha åtkomst till att administrera kameraenheter, användare och föreläsningsanteckningsbilder.

4.7.1 Databasmodellering

För att lagra alla bilder från kameraenheten och dess metadata valde vi att använda en relationell databas placerad på servern. Kriterierna var att datan skall vara lätt och snabb att läsa, skriva från och fungera väl med en webbserver, samt i Java-implementationen av nätverksmodulen. Valet föll på MySQL för dess popularitet bland webblösningar och av anledning att

webbansvarig hade tidigare erfarenhet av denna.

Databasens struktur byggdes upp genom att rita upp möjliga entitetstyper och dess relationer i form av ett ER-diagram. Denna reviderades fyra gånger innan den ansågs färdig för att implementeras. Ett antal ytterligare förändringar lades till även efter att databasen implementerats. Det slutgiltiga ER-diagrammet visas i figur 24.



Figur 24: Databasens upplägg

Informationen separerades i entiteter med relationer enligt diagrammet. Detta för att eliminera redundant data i databasen, för att ge möjligheten att söka efter föreläsningsanteckningsbilder via en kurskod och för att kunna gruppera bilderna i enskilda föreläsningar. Vi ville också ha möjligheten att spåra den ansvariga kameraenheten för en bild och vilken föreläsningssal som denna var i, så denna information lades till som två entiteter.

Dessa entiteter översattes till databastabeller för MySQL. Tabellerna byggdes upp genom insättningsoperationer i SQL-format.

4.7.2 Val av ramverk

För att underlätta programmeringen av webbplatsen gjordes en undersökning för att hitta ett så kallat ramverk som skulle kunna stå som bas för den serverbaserade koden för webbplatsen. Ett ramverk är en färdig bas för att

utveckla en produkt på.

Följande ramverk jämfördes för att avgöra vilket alternativ som skulle användas till detta projekt:

Ruby on Rails (RoR) [16] är ett ramverk skrivet i programmeringsspråket Ruby. Ruby on Rails använder en MVC-modell och hanterar databasåtkomst via så kallade "Active Record" klasser som länkas automatiskt till databasvärden. Detta innebär att databasen genereras av ramverket och att det inte går att använda Active Record klasserna för en egenutvecklad databas.

Wordpress [17] är ett CMS (content management system) skrivet i PHP. Med wordpress medföljer ett färdigt gränssnitt för att hantera innehållet i databasen och databastabellerna har en låst struktur och namngivning. Wordpress har fördelen att det finns en stor användarkrets som erhåller många färdiga och gratis tillägg.

CodeIgniter [18] (CI) är skrivet i PHP och har en färdig MVC-struktur i likhet med Ruby on Rails. I grundutförande kopplas en URL-adress till respektive kontroller med samma namn. Till exempel adressen /foo/bar vidarebefordras till kontrollen foo och metoden bar. Detta gör det snabbt att komma igång med en enkel struktur på webbplatsen. CI erhåller varken något grafiskt gränssnitt eller någon fördefinierad databasstruktur. Istället erhålls ett antal hjälpklasser för att utföra alla de vanligaste operationer för en webbplats. Till exempel erhålls en klass för att abstrahera SQL-kommandon till funktionsanrop.

Java Enterprise Edition (Java EE) är ett Java-paket designat för att utveckla webbplatser [19]. Dessa webbplatser kan drivas av en Glassfish[20] HTTP-server som hanterar HTTP-förfrågningar och vidarebefordrar dessa till Java-klasser. Java EE i grundutförande har likt CI en föreslagen struktur men inget färdigt gränssnitt eller databas. Java EE är mer komplext än de andra systemen och hela applikationen måste kompileras varje gång en ny ändring skall utföras på webbservern, vilket inte är fallet i ovanstående ramverk eftersom dessa består av dynamiskt exekverade skriptspråk.

Nedan sammanställs några relevanta egenskaper tillsammans med för- och nackdelar i relation till webbplatsens syfte.

Det slutliga valet föll på CodeIgniter på grund av dess öppna struktur, det går att använda en ren MySQL-databas, det är väl beprövat och snabbt, samt på grund av att projektets ansvarig för utvecklingen av webbplatsen hade tidigare erfarenhet av programmeringsspråket PHP.

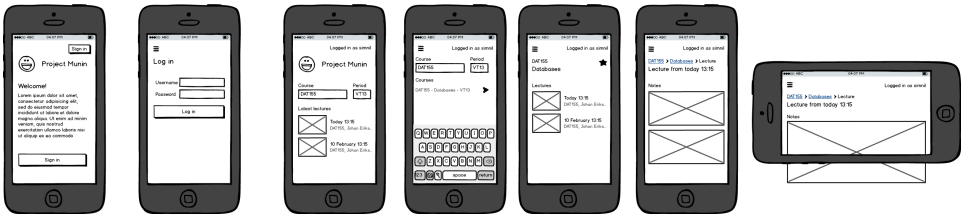
Ramverk	Programmeringsspråk	Grafiskt gränssnitt	Databasstruktur	Fördelar	Nackdelar
PHP/Inget ramverk	PHP	Nej	Öppen	Fullt anpassningsbar	Lång utvecklingstid
Ruby on Rails	Ruby	Nej	Fördefinierad	Bra struktur	Tvingad struktur
Wordpress	PHP	Ja	Fördefinierad	Många färdiga moduler, färdig administrationsvy	Överflödiga funktioner
Java EE	Java	Nej	Öppen	Snabb	Komplext, omständig installation
CodeIgniter	PHP	Nej	Öppen	Enkel, skalbar, snabb utveckling	Inget färdigt

Tabell 1: För och nackdelar med alternativa ramverk för webbsidan

4.7.3 Visuell design

Webbplatsen är menad att kunna användas av två målgrupper; en student som snabbt vill hitta en kurs som den är registrerad på, samt en administratör som vill hantera innehållet i databasen och konfigurera nya och existerande kameraenheter.

Några skisser för gränssnittet skapades i programmet Balsamiq Mockups. Dessa var till för att snabbt få fram en grund för att brainstorma och diskutera designvalen.



Figur 25: Skiss över flödet för en student på en mobiltelefon.

De viktigaste faktorerna för studenternas användarupplevelse ansågs vara att så snabbt som möjligt hitta deras kurs och de senaste föreläsningarna. Därför lades sökfunktionen så högt upp som möjligt på den första sidan som presenteras för användaren (se figur 25). Sökrutans textfält aktiveras automatiskt och sökresultaten visas automatiskt så fort någon text har skrivits i sökfältet. På detta sätt är det enda användaren behöva göra att skriva in några av de första bokstäverna av kurskoden tills kursen syns i sökresultatet. Sedan klickar användaren på en rad i sökresultatet och de senaste föreläsningarna visas i en lista med den senaste föreläsningen högst upp. Från denna kan man ta sig vidare till de sparade föreläsningsanteckningarna.

För gränssnittet för en administratör ansågs det viktigare att innehållet återberättar den systematiska strukturen. Denna delades in i tre vyer: kurser (bilaga 45), kameraenheter (bilaga 46) och användare (bilaga 47). Varje vy visar utförlig data för entiteterna i tabellform så som de är registrerade i databasen. Genom tabellen för kurser kan man bläddra sig vidare till föreläsningar och därefter föreläsningsanteckningar. Detta på grund av att relationerna mellan dessa i databasen är tvingade; det går inte att radera en kurs utan att även radera alla föreläsningar som har en relation med denna och en föreläsning går inte att radera utan att radera alla dess föreläsningsanteckningsbilder.

För att användaren lätt skall kunna följa hierarkin på de olika delarna av webbplatsen visas "breadcrumbs" över innehållet på varje vy på webbplatsen. Varje tidigare steg i hierarkin presenteras som en horisontell lista med den aktuella sidan längst till höger.

4.7.4 Responsiv design

Responsiv design innebär att samma webbplats används för alla skärmstorlekar, men att layouten anpassas dynamiskt för att passa de olika storlekarna. Detta möjliggörs med hjälp av CSS media-queries som laddar olika stilregler för elementen på webbsidan beroende på bredden av det aktuella webbläsarfönstret. Vi ville använda denna teknik eftersom det gör webbplatsen lättillgänglig för smartphones och läsplattor. Det är stor sannolikhet att studenter använder dessa enheter för deras studier. Genom att använda responsiv design bevaras all funktionalitet för dessa mindre enheter samtidigt som utvecklingstiden blir mycket kortare än alternativet att skapa en separat webbplats exklusivt för mindre enheter.

4.7.5 Ramverk för layout och visuell design

För implementation av designen med CSS, HTML och Javascript valdes ytterligare ramverk för att underlätta utvecklingsarbetet samt för att göra

det lätt att underhålla för framtida bruk.

Ett ramverk kallat Bootstrap[21], utvecklat av Twitter, användes för den grundläggande strukturen för layout och stiliseringar av bland annat knappar och formulär. Bootstrap har stilregler som hjälper för att hantera enheter av flera storlekar enligt principen "responsiv design" som förklarades i avsnittet innan.

För att underlätta en konsekvent design användes språket LESS[22]. LESS utökar funktionaliteten av CSS genom att möjliggöra variabler och operationer i koden. LESS kompileras till CSS innan det kan användas av en webbläsare.

För att hantera kompileringen av LESS och Javascript konfigurerades pakethanteraren NPM[23] från servermjukvaran Node.js[24] och hjälpverktyget Grunt[25]. Med hjälp av Grunt kunde LESS och Javascript-filerna övervakas för förändringar och så fort en ändring detekteras så kompileras dessa automatiskt, vilket möjliggjorde en snabbare utveckling.

Utöver detta användes ett paket kallat HTML5 Boilerplate[26] med Modernizr[27] som ger ytterligare grund med praxis för bland annat filstruktur och kompatibilitet med webbläsare.

Till sist användes även ramverket jQuery[28] för att abstrahera Javascript till ett enklare och mer webbläsarkompatibelt sätt.

4.7.6 Implementation

På servern installerades serverprogrammet Apache för HTTP-protokollet, tillsammans med moduler för PHP samt MySQL som databasmotor. Dessa fanns tillgängliga som ett kombinerat paket som kunde enkelt installeras på Ubuntu-installationen på servern[29]. Därefter installerades de valda ramverken för back-end och front-end på respektive rekommenderade beskrivning.

CodeIgniter har inget färdigt grafiskt gränssnitt för att varken visa eller hantera innehållet i en databas utan detta var tvunget att skapas från grunden. Det som CodeIgniter däremot tillhandahåller är ett antal hjälpklasser som kan hantera databasåtkomst, bildhantering, översättning och så vidare. Dessutom kommer det paketerat med en färdig mappstruktur och klasstruktur enligt MVC modellen.

För varje aktivitet skapades en ny kontroller i mappen för kontroller. Till exempel kontrollern för att hantera sökningen kallades browse.php och denna kopplades för att anropas när slutanvändaren öppnar rot-adressen (t.ex. tavlan.se/). Denna hämtar som standard de 10 senaste föreläsningarna från

ett så kallat "Data Accessor Object" (DAO). DAO innehåller alla metoder för att hämta och skriva till databasen. Denna laddades in som ett modellobjekt. DAO använder hjälpklassen från CodeIgniter för att hämta data från MySQL-databasen. Vid en sökning anropas samma kontroller men istället för att visa vyn med de 10 senaste föreläsningarna visas en vy för resultatet av söksträngen istället.

För att minska antalet steg en användare behöver utföra för en sökning används Javascript som kan köras i användarens webbläsare utan att sidan behöver laddas om. Så fort en användare skriver in en bokstav i sökfältet anropas en funktion i Javascript som skickar en HTTP förfrågan till ovanstående kontroller som då returnerar HTML med endast delen med sökresultatet som då kan ersätta det tidigare innehållet där de senaste föreläsningarna tidigare låg. Javascript används också för att hantera ett antal grafiska element såsom animerade laddningssymboler och även för att hantera vissa händelser i administrationen.

4.7.7 Kommunikation med kameraenheten

I gränssnittet för en administratör finns alternativet att redigera inställningarna för en kameraenhet. När användaren klickar på knappen för att spara inställningarna anropas ett Java-program som skickar de nya inställningarna till kameraenheten. Gränssnittet ger feedback om ändringen kunde slutföras eller inte. Informationen om kameraenheten i databasen ändras av Javaprogrammet när denna är säker på att informationen har sparats fullständigt på kameraenheten, och därefter uppdateras informationen på webbplatsen efter det som då är registrerat i databasen.

4.8 Verifiering

För att testa hela systemet sattes det upp givare och en kameraenhet i en föreläsningssal. Sedan kopplades kameraenheten och servern ihop till samma nätverk. Efter det provades det att skicka upp en krittavla till läget som tavlan är färdigskriven och sedan verifierades det om föreläsningsanteckningsbilden kunde ses via webbsidan.

För att se hur användarvänligt systemet är gjordes det en undersökning genom att ett antal personer fick prova att använda webbsidan. Efter att de har provat webbsidan ställdes det två frågor till dem och dessa två frågor var "Hur lättanvänd är webbsidan?" och "Hur gärna skulle du vilja se produkten i salar på Chalmers?". Svaren var i en skala från 1 till 10 där 1 står för instämmer inte alls och 10 instämmer helt.

5 Resultat

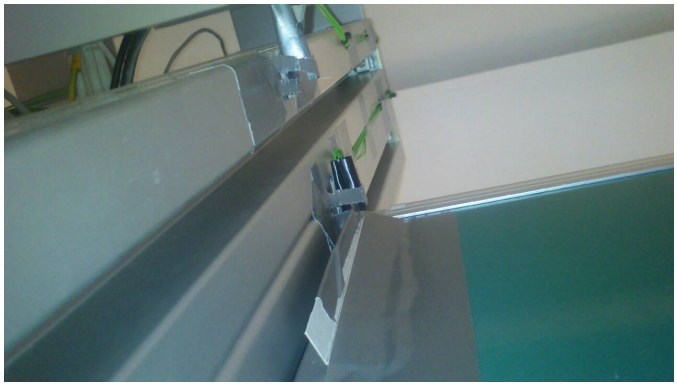
Det som kommer beskrivas i detta kapitel är hur prototypen är installerat i en föreläsningssal, prestandan på kameraenheten och servern, hur webbsidan designad och resultatet från bildbehandlingen.

5.1 Prototypen installerad i en föreläsningssal

Se i figurerna 26 27 på vilket sätt givarna är installerade i en föreläsningssal och se även figur 28 för delen som bryter givaren. Enligt figurerna 29 30 ses det hur kameraenheten är placerade i en föreläsningssal. Det gjordes även ett kopplingskretskort för att förenkla kabeldragningen och kretskortet kan ses i figur 31. Systemet sattes ihop och dess delmoduler testades enligt det som beskrevs i verifieringsstycket.

Figur 26: En bild på hur givarna är installerade med ett längre fotograferingsavstånd.

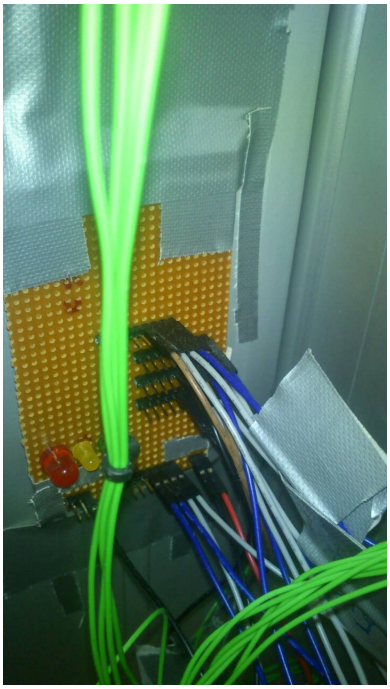
Figur 27: En bild på hur givarna är installerade med ett kortare fotograferingsavstånd.



Figur 28: En bild på hur givaren bryts när en tavla är vid toppläget.

Figur 29: En bild på hur kameraenheten är placerad i en föreläsningssal.

Figur 30: En bild på hur kameraenheten är placerad i en föreläsningssal.



Figur 31: En bild på hur kopplingskretskortet ser ut.

5.2 Flödets prestanda

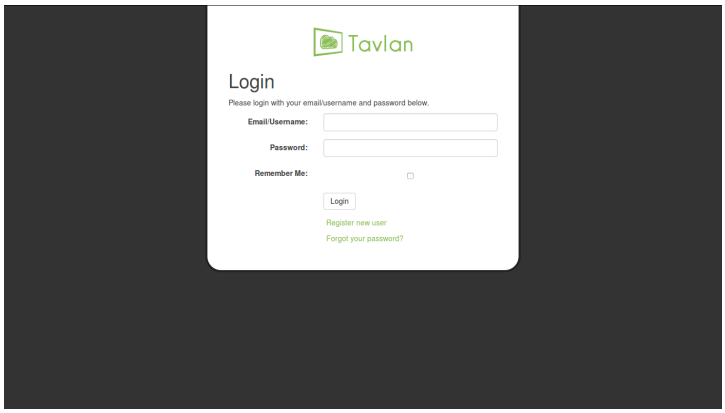
Efter tester med hela flödet exklusive bildanalysen och bildjämförelsen tar det i genomsnitt 1.5 till 2 minuter på kameraenheten att konvertera föreläsningsanteckningsbilden till XML och tiden det tar för servern att bearbeta en föreläsningsanteckningsbild är obefintlig jämfört med den tiden det tar för kameraenheten, se bilaga 3 för testdata. Servern spenderar i genomsnitt 1.4 sekunder per föreläsningsanteckningsbild och denna tiden är exklusive den tiden det tar att överföra bilden, se bilaga 4 för testdata. Alla dessa tider är tagna utan bildanalys på kameraenheten och servern.

5.3 Webbsidans slutliga design

Webbsidan är uppbyggd på ett enkelt, stilrent och användarvänligt sätt. Det finns två vyer på webbsidan och ena delen är den som slutanvändaren kan se som visar alla kursen med sina föreläsningar. Den andra delen är administrationsvyn där administratören kan hantera alla kameraenheter, användare och kurser. När användaren navigerar i nivåerna på webbsidan visas "breadcrumbs" i övre delen av sidan, som gör det enkelt att återvända till tidigare sidor.

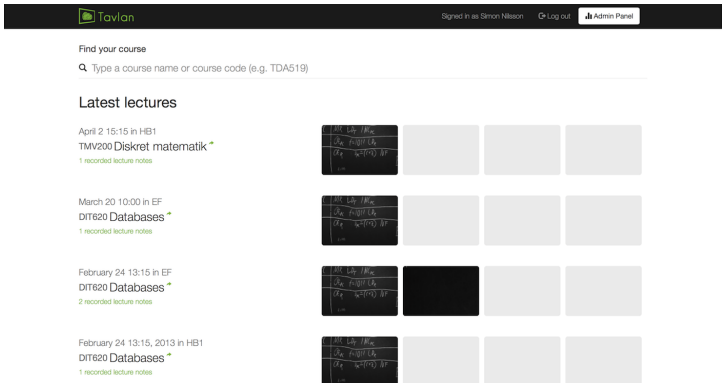
5.3.1 Slutanvändarens gränssnitt

Första sidan som slutanvändaren ser vid ansluting till webbsidan är inloggningssidan. På denna sida kan användaren välja att logga in eller att skapa ett nytt konto. När ett nytt konto ska skapas krävs det att användaren anger en e-mailadress, och för att säkerställa att användaren är en student på Chalmers accepteras endast mailadresser som slutar på @student.chalmers.se. Om någon annan e-mailadress anges får användaren ett felmeddelande om att det måste vara en Chalmersadress. Ett verifikationsmail kommer skickas till den angivna e-mailadressen för att bekräfta att den nya användaren verkligen äger den e-mailadressen. Det finns också en funktion för att återställa lösenordet om användaren har glömt detta.



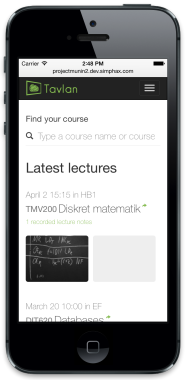
Figur 32: Webbsidans slutresultat av inloggningssidan.

Väl inloggad möts användaren av ett smalt navigationsfält i övre delen av gränssnittet. Är användaren inloggad som administratör visas en knapp för att ta sig till administrationsvyn längst till höger. Precis under menyn finns ett sökfält för att söka efter en viss kurs genom att skriva kurskoden eller kursnamnet. Nedanför sökfältet finns en lista med de 10 senaste föreläsningarna som registrerats i systemet. Varje rad listar information om vilken tid föreläsningen hälls, vilken kurs det gällde och vilken föreläsningssal den ägde rum. Det visas även de fyra senaste föreläsningsanteckningsbilderna i miniatyr för en viss föreläsning. Om den föreläsningen som användaren söker efter finns på startsidan krävs det inte mer än ett musklick för att se föreläsningen från denna vy. Det alternativa flödet är att användaren söker kursen via sökrutan. Sökrutan aktiveras automatiskt och resultaten visas så fort användaren skriver en bokstav, så användaren kan skriva de första bokstäverna i kurskoden sedan välja kursen med ett musklick för att se alla föreläsningar för en viss kurs och sedan välja den föreläsningen som användaren är ute efter. För att hitta en viss föreläsning tar det alltså minst ett musklick och maximalt två musklick.



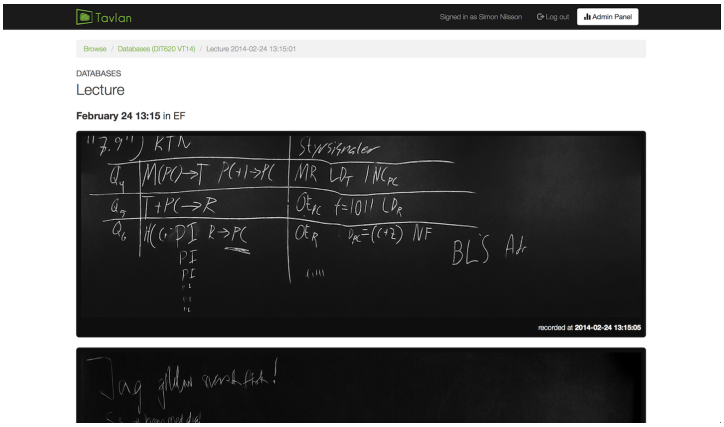
Figur 33: Skärmdump av startsidan.

Loggar en användare in med en enhet med mindre skärmstorlek anpassas layouten automatiskt för att passa denna enhet. Bilden visar startsidan på en smarttelefon och här har navigationen gömts i navigationsraden och visas först när användaren klickar på knappen med tre streck. Istället för att kursbeskrivningarna är i vänsterkolumn och föreläsningsanteckningsbilderna i högerkolumn visas de nu staplade på varandra för att behålla en läsbar storlek på typsnittet.



Figur 34: Startsidan visad med en mobiltelefon.

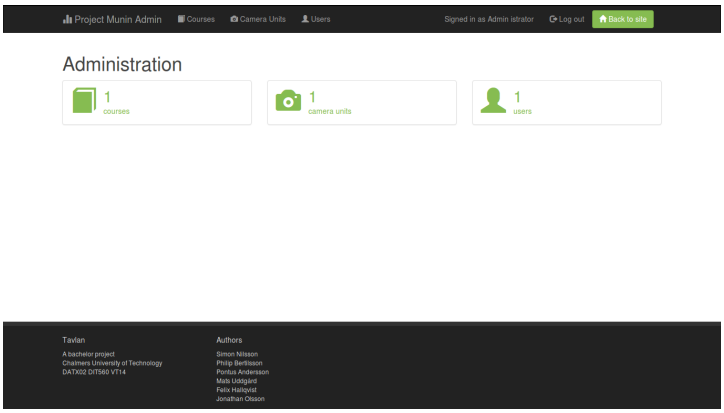
Efter att användaren har valt en föreläsning listas alla föreläsningsanteckningsbilder med den senast tagna bilden högts upp.



Figur 35: Hur en föreläsning presenteras på webbsidan.

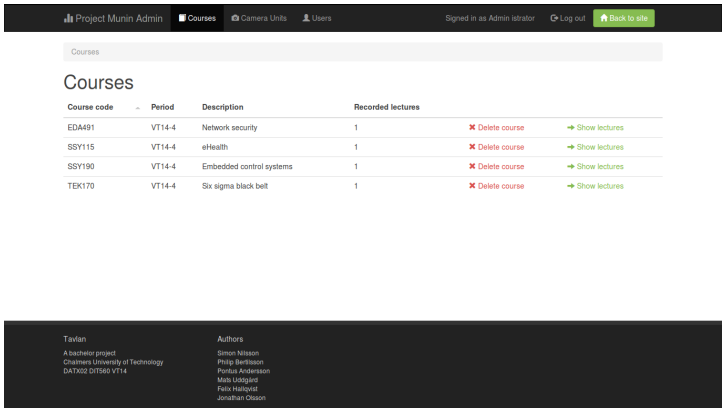
5.3.2 Administratörens gränssnitt

Administrationsvyn är enkelt uppbyggt med tre tydliga huvuddelar där administratören kan välja att hantera kurser, kameraenheter eller användare. Högst upp vid inloggningsknappen är dessa delar även lokaliserade för att enkelt kunna byta till en annan huvuddel.



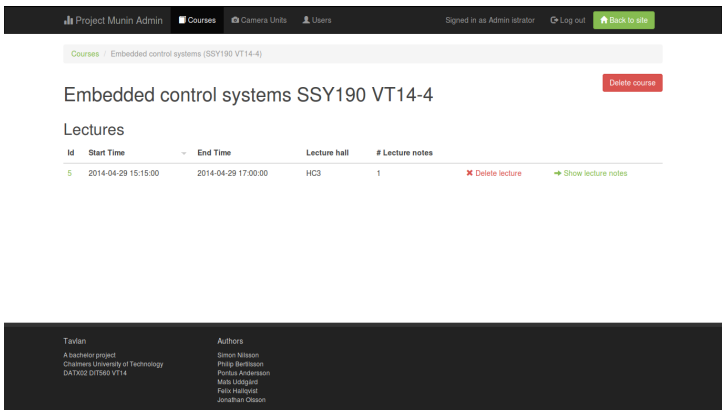
Figur 36: Slutresultatet av administrationsvyn.

I administrationsvyn för kurser radas alla kurser upp efter varandra sorterat efter kurskod, period, beskrivning eller efter antal registrerade föreläsningar. Alla dessa fyra sorteringsinställningarna är enkla att ändra genom att trycka på någon av rubrikerna. Det går även att ta bort en kurs eller att klicka vidare in till att se alla föreläsningar för en viss kurs.



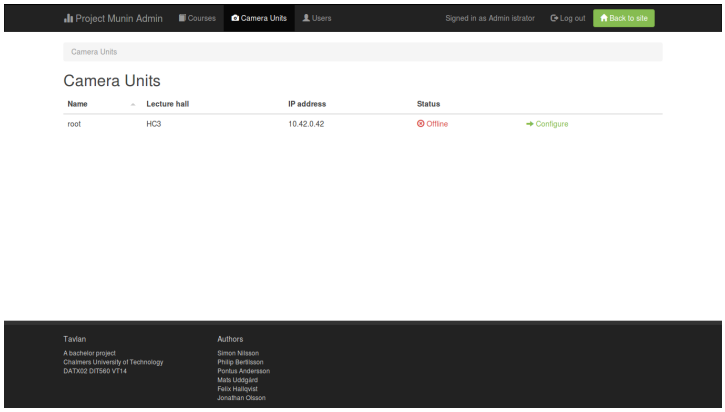
Figur 37: Hur kursadministrationsvyn ser ut.

Väl inne i en kurs radas alla föreläsningar för den kursen upp efter olika sorteringsparametrar. Dessa parametrar är föreläsningens ID-nummer, starttid, sluttid, föreläsningssal eller antal föreläsningsanteckningsbilder i föreläsningen. Även här går det att ta bort kursen eller ta bort en enskild föreläsning. Vidare härifrån går det att klicka sig in till en föreläsning och se alla föreläsningsanteckningsbilder för en föreläsning. På föreläsningens sida går det att ta bort enskilda föreläsningsanteckningsbilder om det behövs.



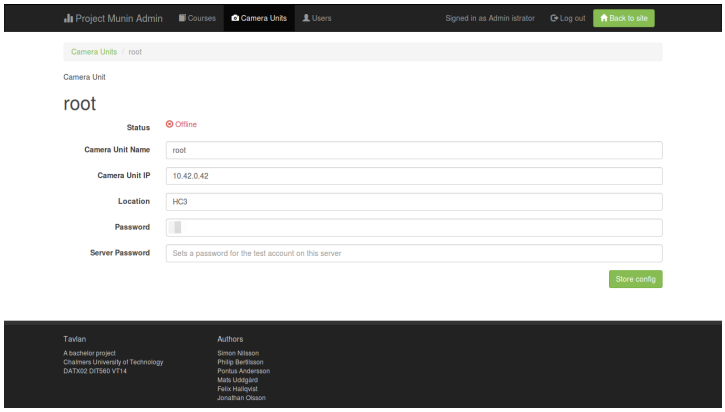
Figur 38: Hur det är utformat inuti en kurs med föreläsningar.

Kameraenhetsadministrationsvyn är uppbyggd som resten av webbsidan med ett par olika sorteringsparametrar. De sorteringsparametrar och information som visas på kameraenhetsvyn är kameraenhetens unika namn, vilken föreläsningssal den är placerad i, vilken IP-adress den har eller om kameraenheten är online. Det går även att konfigurera kameraenheten genom att klicka på knappen configure.



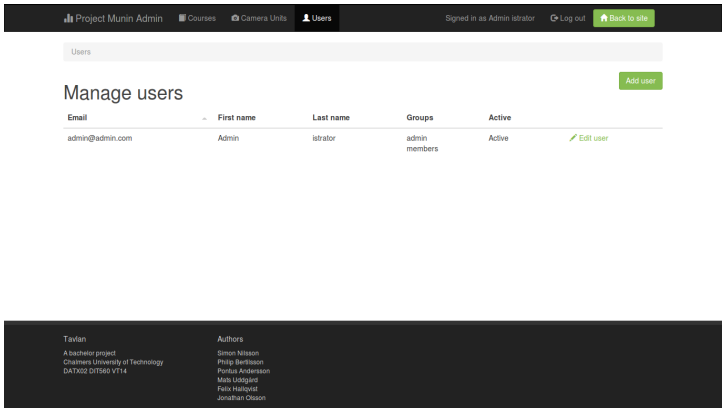
Figur 39: Slutresultatet för hur kameraenhetsadministrationsvyn är utformat.

I inställningsmenyn för kameraenheter finns det möjlighet att enkelt skicka nya inställningar till kameraenheter. Den nya informationen som går att skicka är föreläsningssal och det nya lösenordet för servern. Annan information som visas är kameraenhetens namn, IP-adress och lösenord. För att skicka de nya inställningarna är det bara att klicka på knappen store config.



Figur 40: Hur inställningarna presenteras för en kameraenhet.

Vid vyn för att hantera användare visas alla användare som är registrerade på webbsidan. Här visas informationen e-mail, tilltalsnamn, efternamn, vilken grupp användaren tillhör och om användaren är aktiv. Här går det att klicka sig vidare för att editera informationen för de olika användarna. Det finns även en funktion för att lägga till en helt ny användare genom att klicka på knappen add user.



Figur 41: Resultatet för användaradministrationsvyn på webbsidan.

5.4 Användarundersökning

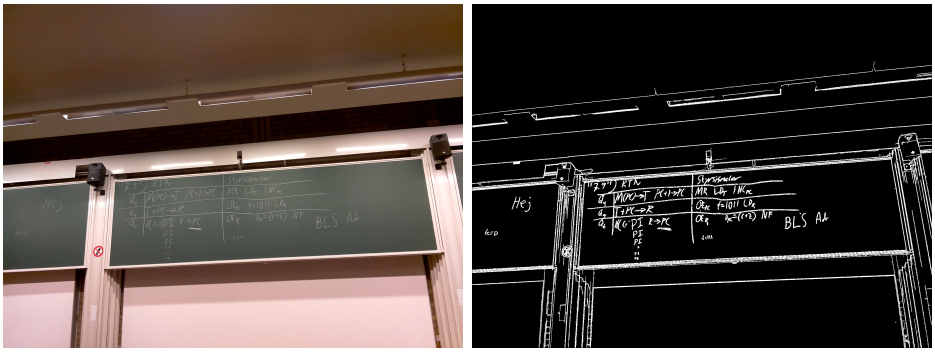
Undersökningen gav ett genomsnitt av 8 av 10 på användarvänligheten och 8.5 av 10 på hur gärna de skulle vilja se produkten i Chalmers föreläsningssalar. Se bilaga2 för undersökningen.

5.5 Bildanalys

Bildbehandlingen kunde identifiera och skära ut krittavlan i bilder när den kördes i merparten av våra utvecklingsmiljöer, men erfarde problem när den skulle köras på kameraenheten.

5.5.1 Kantdetekteringen

Den resulterande kantbilden innehöll inte särskilt mycket brus och kanterna var tydliga utan hål, men också mycket tjockare än nödvändigt. Riktningen på kanterna var relativt kontinuerliga längs en kantlinje.

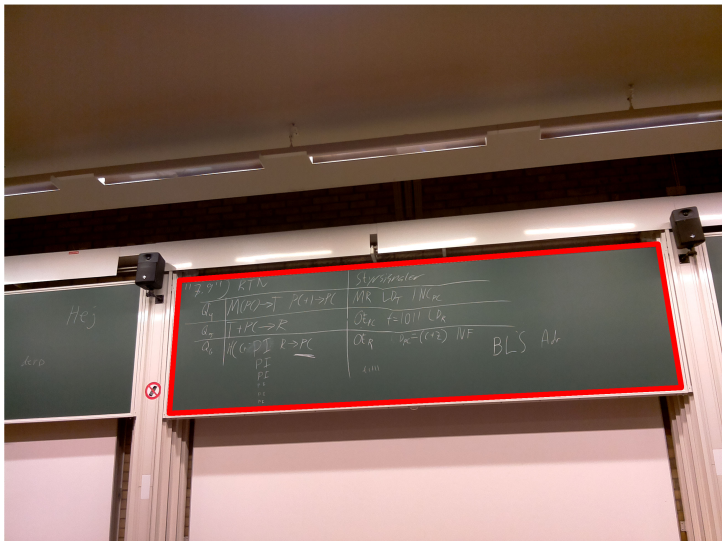


Figur 42: En bild från en föreläsningssal och dess resulterande kantbild skapad med Kirsch.

5.5.2 Identifieringen av krittavlan

Identifikationen av fyrhörningar i bilden från hough-transformen resulterade som tänkt i de olika fyrhörningarna som kan bildas av linjerna. Dessutom lyckas den filtrera bort fyrhörningar som inte matchar mot kantbildens kanter eller där linjerna resulterat i hörn utanför bilden.

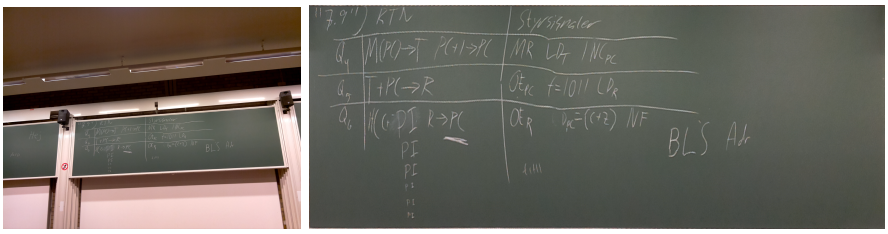
Som synes på figur 43 så överensstämmer den identifierade krittavlans kanter ungefärligt med hörnen på krittavlan i originalbilden.



Figur 43: Den identifierade krittavlan markerad i rött över dess originalbild.

5.5.3 Utskärningen

Tavlan skärs ut så att allt inom de identifierade hörnen tas med. Dock så blir det en viss förvrängning av tavlan men man ser fortfarande tydligt allt som står. Vi har testat bilder tagna från flera olika vinklar men tavlorna ser fortfarande tydliga ut efter utskärningen.



Figur 44: Bild före och efter utskärning.

5.5.4 Jämförelse

Vi blev delvis klara med bildjämförelsen. Den kan identifiera tomma bilder och även märka om två bilder är väldigt lika varandra. Jämförelsen kan i princip bara kompensera för små fel från utskärningen. Den kan inte se om en tavla har fått ett extra ord skrivet på sig.

6 Diskussion

Genom projektets gång har var person som hört om vårt projekt ifrågasatt och undrat varför vi inte helt enkelt antingen filmar allt eller spelar in ljudfiler i samband med bilderna vi skall ta med vårt projekt. Det skulle självklart vara en möjlighet att lägga till dessa funktioner men de var utanför detta projektets ramar. Vi anser dock inte att video eller ljud kan ersätta detta system. Systemet är till för att komplettera handskrivna anteckningar en student må göra på en föreläsning, inte en ersättning för att närvara på en föreläsning. Det är skillnad på att kolla igenom en video och att leta genom anteckningar. Vi anser detta system lämpar sig bättre för att snabbt kolla upp en anteckning för att komma ihåg vad en föreläsare sagt.

I tre delar av bildbehandlingen hittade vi på egna algoritmer. Detta för att vi inte hittade redan existerande varianter som kunde förstås och implementeras inom rimlig tid. Det här betyder att det finns stort utrymme för förbättring, både genom att använda nya algoritmer och ren optimering av våra egna algoritmer. Algoritmerna för jämförelse av bilder, för utskärning av ett område och för identifikation av vilken fyrhörning som representerar krittavlan är de algoritmer som vi själva har kommit på från grunden.

Utifrån resultatet från hur lång tid servern behöver spendera per föreläsningsanteckningsbild, ser vi att servern kan hantera 2400 bilder per timme om endast avkodningen och insättning av XML-filen är med i beräkningen. En kameraenhet klarar av att producera en bild på strax under två minuter och detta medför att kameraenheten kan producera 30 bilder per timme. Detta medför att servern skulle klara av att hantera 80 kameraenheter utan att hamna efter. Servern som användes är mer än 10 år gammal. Med bättre hårdvara skulle det medföra att att den skulle klara av betydligt fler kameraenheter. Detta resultat är dock utan bildbehandling i någon form. Om bildbehandlingen skulle vara implementerad på kameraenheten skulle det medföra att den spenderar betydligt mer tid per bild.

6.1 Etik- och miljöaspekter

Vårt mål med projektet var att underlätta samt hjälpa elever i studierelaterade situationer. Att ge denna hjälp kan ses som att ge studenterna en chans att slippa stressa upp sig och faktiskt ta sig tid att lära sig något under lektionerna. En negativ bieffekt kan vara att folk blir lata under lektionerna istället för att koncentrera sig på föreläsningen börjar fokusera på annat, vilket skulle ge motsatt effekt. Detta kan i värsta fall bli ett störande moment för resterande elever. Hur studenter väljer att använda sin tid ligger på en individuell basis och väldigt svårt att påverka inom projektets ramar

Om detta system sätts igång kommer åtgången av anteckningsblock minska

och därmed en väldigt stor och onödig åtgång på pappersmassa då det kan antas att studenter på eftergymnasiala utbildningar har tillgång till dator och internet. Värt att nämna att de som fortfarande vill anteckna inte hindras till detta.

6.2 Diskussion om målen och utmaningar

Att hålla priset på prototypen lågt anser vi vara viktigt för att produkten ska kunna vara attraktiv på marknaden. Vi har räknat ut ett ungefärligt pris på den nuvarande prototypen som hamnar på ungefär 800 kr, bortsett från priset för FPGA-kortet vi använder till kamerakretsen. I en slutgiltig produkt skulle det vara bättre att använda ett annat alternativ istället för en FPGA, exempelvis någon form av analog multiplexer, som då antagligen skulle få ner priset till runt 1000 kr per kameraenhet. Vi räknar inte med kostnad för servern eftersom hårdvaran för denna inte är del av produkten som utvecklats.

Exakt vad användarvänlighet handlar om finns det flera teorier om vilket gör begreppet väldigt abstrakt. I enkla drag kan man säga att det handlar om hur enkel en webbplats är att använda samtidigt som den fyller sitt syfte hos besökaren. Ett annat problem är att detta är en subjektiv upplevelse vilket kan vara väldigt svårt att mäta. Värden som säkerhet och tillgänglighet kan man ha en klar uppfattning medan värden likt utseende och interaktion lättast kan verifieras genom en omfattande undersökning. Ur vår mindre användarvänlighetsundersökning kan vi se att de medverkande tyckte hemsidan var relativt användarvänlig. Dock är undersökningen inte vetenskapligt genomförd och skulle behöva göras om genom att ha en större undersökningsgrupp och välja personer slumpmässigt.

6.3 Tidsplanering

Den övergripande tidsplanen hölls och vi låg i fas genom hela projektet. Deltidsplanerna förändrades och blev inkorrekta snabbt. Planen för nätverket skrotades tidigt på grund av att vi bestämde oss för att använda SCP istället för att skriva koden själv. Bildbehandlingen, specifikt kantdetektionen och utskärningen, var svårare och tog längre tid än vad som planerades och är inte fullständigt färdig. Bildjämförelsen tog istället mycket kortare tid än vad som förväntades. Vissa delar som kantdetekteringen blev klar en vecka innan rapportens inlämningsdatum. Tidsplanen för hårdvarudelen följdes förutom att det tog längre tid att skapa kretsen för tre kameror som därför inte blev klar. Resterande områden i hårdvarudelen av projektet gjordes i tid och fungerar. Tidsplanen för webbdelen följdes också förutom att vi inte integrerade webbsidan i pingpong och att inloggningssystemet inte är ihopkopplat med Chalmers CID.

6.4 Skillnader gentemot planeringsrapporten

Från början bestämde vi oss för att använda VPN för att sammankoppla alla enheter och för att kameraenheterna automatiskt skulle kunna koppla upp sig till servern. VPN skulle också tillåta kameraenheterna att själv hitta serverns IP-adress samt att nätverket mellan enheterna skulle bli säkert genom VPN-kryptering. VPN valdes bort på grund av att det blev enklare att implementera systemet med andra medel. Datan skickas och krypteras istället med hjälp av SCP och kameraenheterna är nu istället förinställda med serverns IP-adress.

I projektets planeringsfas bestämdes det att bildbehandlingen skulle innefatta utskärning av en tavla på en bild, skevhetskorrigering av bilden och filter för att ta bort eventuellt brus och för att förbättra kontrasten mellan texten och tavlan. Filtren blev aldrig implementerade i slutprototypen då de ansågs vara lägre prioriterade än de andra funktionerna.

6.5 Möjliga förbättringar och tillägg

En förbättring som kan göras är att minska användningen av XML i form av att inte koda föreläsningsanteckningsbild till XML-format. Detta är på grund av att det tar i genomsnitt 1.5 till 2 minuter att konvertera den. Det som också skulle förbättras är att filerna som skickas till servern inte blir 33% större jämfört med originalstorleken på föreläsningsanteckningsbild. Detta är för att XML konvertering ökar föreläsningsanteckningsbildens storlek med 33%. Varför denna förbättring skulle vara önskvärd är om föreläsaren skriver snabbt för att det skulle medför att många tavlor blir fotade. Vilket kan leda till att många föreläsningsanteckningsbilder blir köade på kameraenheten och det kan leda till att det tar timmar för en föreläsningsanteckningsbild att visas på webbsidan. Detta behövs testas mer om detta kommer ske eller inte. En föreläsare skulle behöva skriva ungefär 30 tavlor i timmen för att detta skulle blir ett problem.

För att få ut extra prestanda ur kameraenheten skulle kantdetektionsalgoritmen kunna köras på dess GPU. Ett program skulle kunna skrivas där det skulle använda GPU:n för att utnyttja parallellismen som finns i bilden då den har så många pixlar som man kan arbeta med samtidigt. Det skulle även medföra att andra program får mer CPU-tid på kameraenheten.

En möjlig funktion som kan implementeras till webbsidan är att göra det möjligt för användare att kunna kommentera föreläsningsanteckningsbilder eller föreläsningar så studenterna kan diskutera dessa. Två funktioner som skulle kunna läggas till är funktionerna att kunna välja en kurs som favorit så den kursen alltid finns på studentens framsida och lägga till så att studenterna skulle kunna bläddra efter kurser utan att använda sökfunktionen.

En annan funktion som skulle vara bra att lägga till hade varit att låta föreläsaren ladda upp egna dokument direkt på hemsidan, exempelvis PDF-filer och föreläsarens presentationsfiler, för att ge möjlighet att samla ihop dokument och föreläsningsanteckningar.

För tillfället använder Chalmers pingpong för kurser och Chalmers CID (Chalmers inloggningssystem) för att logga in på alla av skolans system. För att utvidga detta projekt till att nå fler studenter skulle det vara en möjlighet att integrera pingpong till systemet så föreläsningsanteckningsbilder skickas till pingpong och sätts in på rätt kurs. Det kan också vara önskvärt att integrera projektets webbsida med Chalmers CID inloggning så det blir lättare att använda projektets webbsida.

För att förenkla konstruktionen av kamerakretsen med multiplexern skulle någon annan typ av kameror kunna användas. Eftersom Raspberry Pi Camera Module har signaler som är differentiella spänningar kunde vi inte använda en enkel multiplexer eller programmera en CPLD-krets för att sköta de tre kamerorna utan fick använda en FPGA istället. Om vi hade använt andra kameror som inte har differentiella signaler kunde vi byggt kretsen av enklare och billigare delar.

Det skulle kunna tänkas att normalform inte är den bästa parametriseringen av räta linjer för vårt ändamål. Muffparametrisering är en bra kandidat och skulle tillföra ett antal fördelar. Dock finns det en del nackdelar varav den största är när vi utnyttjar riktningen på en kantpixel genom att enbart hitta någorlunda relevanta linjer så behöver intervallet av vinklar konverteras till punkter på bildens ram. En approximering som skulle kunna vara intressant att analysera är att ha ett litet antal förutbestämda punkter längs ramen vilka används för att hitta vinkeln på en linje mellan dem och pixeln. Den närmsta punkten och dess två grannar, samt de motsvarande punkterna på den motstående sidan, är då intervallet som linjerna ska gå över. Men detta implementerades aldrig och är således fortfarande oprövat.

7 Slutsatser och framtida arbete

Prototypen uppnådde målet med att automatiskt kunna ta bilder av en tavla under föreläsning för att efter bildbehandling skicka de redigerade föreläsningsanteckningsbilden till en server för att distribueras på en hemsida som studenter lätt kan komma åt och använda. Vi kan konstatera att det är möjligt att genomföra denna typ av system i en föreläsningssal. Vi kan mer specifikt konstatera att uppdelningen i en server och flera kameraenheter var lyckad och ger oss en centraliserad lösning som samtidigt kan hantera flera salar.

Vi har identifierat algoritmer för bildbehandling som krävs för att automatiskt identifiera och skära ut krittavlor. Vi valde ut algoritmer som i hänsyn till projektets tidsram var tillräckligt lätta att implementera men ändå innehållit funktionaliteten som eftersträvades.

Raspberry Pi, som var valet av enkortsdator, klarar att prestandamässigt köra all bildbehandling.

Utifrån användarundersökningen som utfördes och samtal med elever och lärare på Chalmers kan vi ana att det finns ett starkt intresse för en färdig produkt utifrån vår prototyp.

7.1 Färdigställande och fortsatt arbete av produkten

Programdelen för TimeEdit skulle definitivt behövas skrivas om och använda ett API från TimeEdit. Detta är på grund av att använda HTML-sidorna för att söka upp informationen inte är bra för att HTML-sidor kan förändras över tid medan ett API är konstant och ändras inte. Det funkar som vi har gjort men vi vet inte hur länge det kommer funka och om det ändras är det mycket arbete som behövs läggas ner för att uppdatera alla kameraenheter.

Vid en slutprodukt skulle det även behövs läggas till fler funktioner till administrationsgränssnittet. Exempelvis skulle det kunna läggas till att det går att se kameraenheters loggfiler, kunna hantera fler inställningar från administrationsgränssnittet och lägga till funktionen att möjliggöra att skicka nytt serverlösenord till alla kameraenheter samtidigt.

I slutprodukt skulle servern behövs göras mer felsäker genom att exempelvis kontrollera att inkommande XML-filerna är korrekt och kontrollera att den avkodade föreläsningsanteckningsbilden inte är korrupt.

Kameraenheten skulle dessutom behövas göras mer allmänt säker mot dataintrång genom att exempelvis kryptera inställningsfilen som är placerad i kameraenheten och på något sätt förhindra att utomstående kan ansluta fysiskt till kameraenheten med en skärm, datormus och tangentbord. Även servern och webbsidan bör göras mer säker. På webbsidan har det tagits viss hänsyn till att förhindra att användare kan exekvera SQL kod eller programkod via formulären. Det har också setts till så att lösenord för användarna är säkra mot dataintrång. För tillfället finns det inga kända säkerhetshål.

8 Källförteckning

Referenser

[1] JonMcLoone, *Edge detection applied to a photograph*, Wikipedia, 8 Juni 2010. [Online]. Tillgänglig: `http://en.wikipedia.org/wiki/File:EdgeDetectionMathematica.png`. [Hämtad: 18 Maj, 2014]

[2] J. R. Parker. *Algorithms for Image Processing and Computer Vision*, Andra Upplagan, Indianapolis, IN: Wiley Pub., cop. 2011 [E-Bok]. Tillgänglig: Summon

[3] J. G. Wu, et al, Research on Edge Detection of Thin Sheet Part Dimension Inspection System Based on Machine Vision", *Applied Mechanics and Materials*, vol 103, ss.194-198, sep 2011.

[4] J. Canny, "A Computational Approach to Edge Detection," *Pattern Analysis and Machine Intelligence*, IEEE Transactions on, vol. PAMI-8, nr. 6, ss. 679-698, Nov. 1986. [Online] Tillgänglig: IEEEXplore, `http://ieeexplore.ieee.org/Xplore/home.jsp`. [Hämtad: 16 Maj, 2014].

[5] V.F Leavers, *Shape detection in computer vision using the Hough transform* London, UK: Springer London, 1992 [E-Bok]. Tillgänglig: SpringerLink

[6] Z. Zhang & L.W. He, "Whiteboard scanning and image enhancement" *Digitial Signal Processing*, vol. 17, nr. 2, ss. 414–432, Mars 2007 [Online]. Tillgänglig: ScienceDirect `http://www.sciencedirect.com/science/article/pii/S1051200406000595`. [Hämtad: 27 Mars, 2014]

[7] T. Akenine-Möller, et al., *Real-Time Rendering*, Tredge Upplagan. Wellesley, MA: A K Peters, Ltd., 2008.

[8] Jwrodgers,"Wikipedia," (2014), *Wikipedia*, 2014. [Online]. Tillgänglig: `http://en.wikipedia.org/wiki/File:RaspberryPi.jpg`. [Hämtad: 16 Maj, 2014].

[9] Wikipedia, "Raspberry Pi," *Wikipedia*, 14 Maj 2014. [Online]. Tillgänglig: `http://en.wikipedia.org/wiki/Raspberry_Pi`. [Hämtad: 16 Maj, 2014].

[10] Farnell, "Raspberry Pi," *Farnell*, 2014. [Online]. Tillgänglig: `http://se.farnell.com/jsp/displayProduct.jsp?sku=2191863&isRedirect=true#`. [Hämtad: 16 Maj 2014].

[11] Farnell, "Arduino," *Farnell*, 2014. [Online]. Tillgänglig: http://se.farnell.com/arduino/a000056/w5100-enet-shield-arduino-board/dp/2061037. [Hämtad: 16 Maj 2014].

[12] archlinux,"Raspberry Pi," *archlinux*, 2014. [Online]. Tillgänglig: http://archlinuxarm.org/platforms/armv6/raspberry-pi. [Hämtad: 16 Maj 2014].

[13] M, McCauley, "bcm2835," *bcm2835*, 18 Jan 2014. [Online]. Tillgänglig: http://www.airspayce.com/mikem/bcm2835/. [Hämtad: 16 Maj 2014].

[14] Wikipedia, "Secure copy," *Wikipedia*, 16 Maj 2014. [Online]. Tillgänglig: http://en.wikipedia.org/wiki/Secure_copy. [Hämtad: 17 Maj 2014].

[15] D, Jelovic "Why Java Will Always Be Slower than C++," *Jelovic*, 2014. [Online]. Tillgänglig: http://www.jelovic.com/articles/why_java_is_slow.htm. [Hämtad: 17 Maj 2014].

[16] Ruby on Rails, "Ruby on Rails web framework," *Ruby on Rails*, 2014. [Online]. Tillgänglig: http://www.rubyonrails.org. [Hämtad: 19 Maj 2014].

[17] Wordpress, "Wordpress CMS," *Wordpress*, 2014. [Online]. Tillgänglig: http://www.wordpress.org. [Hämtad: 19 Maj 2014].

[18] EllisLab. Inc, "CodeIgniter," *EllisLab, Inc*, 2014. [Online]. Tillgänglig: http://ellislab.com/codeigniter. [Hämtad: 19 Maj 2014].

[19] Oracle, "Java Enterprise Edition," *Oracle*, 2014. [Online]. Tillgänglig: http://www.oracle.com/technetwork/java/javaee/overview/index.html. [Hämtad: 19 Maj 2014].

[20] Glassfish, "Glassfish Server," *Glassfish*, 2014. [Online]. Tillgänglig: http://glassfish.java.net. [Hämtad: 19 Maj 2014].

[21] Bootstrap, "Bootstrap front-end framework," *Bootstrap*, 2014. [Online]. Tillgänglig: http://getbootstrap.com. [Hämtad: 19 Maj 2014].

[22] Less.js, "LESS," *Less.js*, 2014. [Online]. Tillgänglig: http://lesscss.org. [Hämtad: 19 Maj 2014].

[23] NPM, "Node Package Manager," *NPM*, 2014. [Online]. Tillgänglig: http://www.npmjs.org. [Hämtad: 19 Maj 2014].

[24] Joyent, Inc, ”Node.js,” *Joyent, Inc*, 2014. [Online]. Tillgänglig: `http://nodejs.org`. [Hämtad: 19 Maj 2014].

[25] Grunt, ”Grunt: The JavaScript Task Runner,” *Grunt*, 2014. [Online]. Tillgänglig: `http://gruntjs.com`. [Hämtad: 19 Maj 2014].

[26] HTML5 Boilerplate, ”HTML5 Boilerplate web front-end template,” *HTML5 Boilerplate*, 2014. [Online]. Tillgänglig: `http://html5boilerplate.com`. [Hämtad: 19 Maj 2014].

[27] Modernizr, ”Modernizr feature detection library,” *Modernizr*, 2014. [Online]. Tillgänglig: `http://modernizr.com`. [Hämtad: 19 Maj 2014].

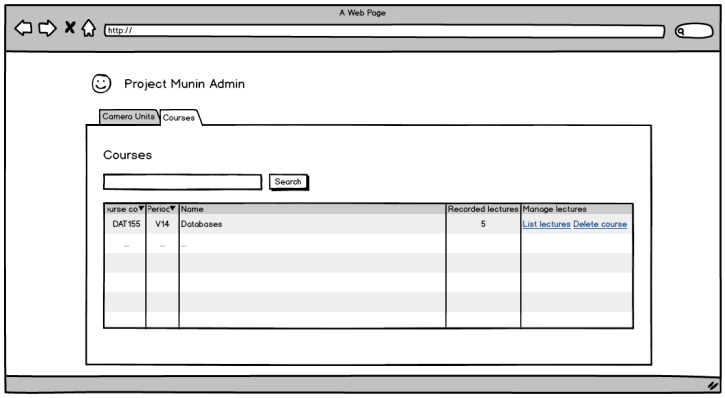
[28] jQuery, ”jQuery javascript framework,” *jQuery*, 2014. [Online]. Tillgänglig: `http://jquery.com`. [Hämtad: 19 Maj 2014].

[29] Ubuntu, ”LAMP - ApacheMySQLPHP,” *Ubuntu*, 2014. [Online]. Tillgänglig: `https://help.ubuntu.com/community/ApacheMySQLPHP`. [Hämtad: 19 Maj 2014].

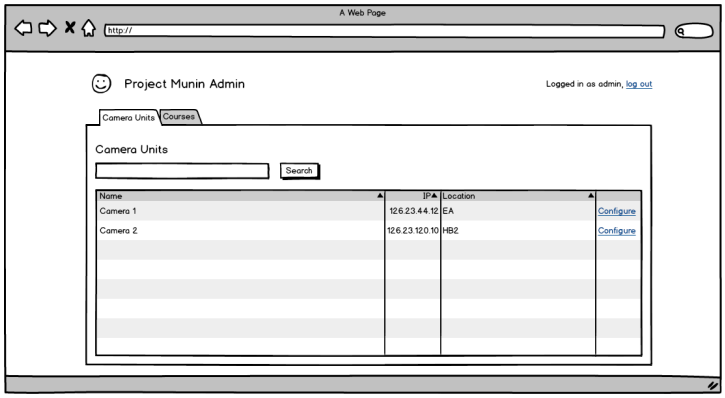
[30] MIPI, *MIPI Alliance, Inc.*, 2014. [Online]. Tillgänglig: `http://mipi.org/specifications/camera-interface#CSI2` [Hämtad: 4 Juni 2014]

[31] Wikipedia, Differential signaling", *Wikipedia*, 11 Maj 2014. [Online]. Tillgänglig: `http://en.wikipedia.org/wiki/Differential_signaling`. [Hämtad: 4 Juni 2014]

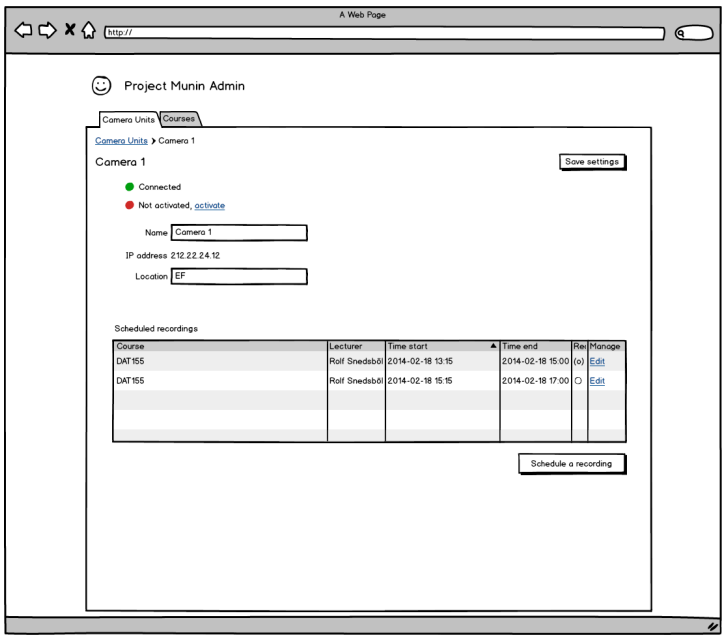
9 Bilagor



Figur 45: Mockup för webbsida före administratörkursvyn



Figur 46: Mockup för webbsida före administratörkameraenhetsvyn



Figur 47: Mockup för webbsida före administratörKameraenhetsinställningsvyn

Person	Användarvänlighet	Åtrovärdhet
Person 1	9	8
Person 2	9	8
Person 3	5	8
Person 4	9	10
Person 5	9	6
Person 6	8	7
Person 7	8	10
Person 8	8	10
Person 9	9	8
Genomsnitt	8.2	8.3

Tabell 2: Undersökning av vad personer tyckte om systemet med en skala från 1 till 10

Starttid	*Sluttid*	*Skillnad*
12.39.34:198	12.39.35:758	1:560
13.15.50:613	13.15.52:112	1:501
13.39.59:643	13.40.00:552	0:909
14.56.59:626	14.57.01:067	1:441
HH.MM.ss:SSS	HH.MM.ss:SSS	s:SSS

Tabell 3: Tider som servern tar för att avkodning av föreläsningsanteckningsbilder och insättning in i databasen.

Starttid	*Sluttid*	*Skillnad*
13.14.11:175	13.15.43:729	1.32:554
13.46.08:004	13:47:40:218	1.32:214
14:41:37:677	14:43:20:698	1.54:021
14:47:01:377	14:48:51:470	1.50:093
HH.MM.ss:SSS	HH.MM.ss:SSS	M.ss:SSS

Tabell 4: Tiden det tar för kameraenheten att skapa XMl filen som ska skickas

Algoritm	Tid1	Tid2	Tid3
Kirsch: Gråskala	36.683271	36.640032	36.549901
Kirsch: Färgkanaler	146.812939	146.622114	146.405043

Tabell 5: Tiden (i sekunder) det tar att kantdetektera en bild med olika algoritmer på en laptop i debug-läge.

Algoritm	Tid1	Tid2	Tid3
Kirsch: Gråskala	35.614894	35.538143	35.688303
Kirsch: Färgkanaler	141.574526	141.285081	140.788832

Tabell 6: Tiden (i sekunder) det tar att kantdetektera en bild med olika algoritmer på en laptop.