CHALMERS

Studier av utseendet på askan i sodapannor med hjälp av bildanalys

Studies of the ashes combustion in recovery boilers using image analysis

Examensarbete inom högskoleingenjörsprogrammet Mekatronikingenjör

Sofia Samuelsson
Carin Collinder

Institutionen för Signaler och System
Avdelningen för Reglerteknik, Automation och Mekatronik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige, 2013
Examinator: Bertil Thomas
Handledare: Manne Stenberg

Studier av utseendet på askan i sodapannor med hjälp av bildanalys

Examensarbete inom högskoleingenjörsprogrammet
Mekatronikingenjör

Sofia Samuelsson
Carin Collinder

Institutionen för Signaler och System
CHALMERS TEKNISKA HÖGSKOLA
Göteborg 2013

Förord

Detta examensarbete utfördes under våren 2013 och motsvarar 15 hp. Examensarbetet är ett moment som ingår i Mekatronikingenjörsprogrammet vid Chalmers tekniska högskola.
Största delen av arbetet uträttades på företaget SootTech AB i Göteborg. Några besök gjordes även till Aspa bruk i Munksjö.

Vi vill tacka Erik Dahlén, Chef på SootTech AB, för att ha gett oss möjligheten till detta intressanta examensarbete.

Vi vill även tacka Alexander Hentschel, student på masterprogrammet ”Computer Systems and Networks” på Chalmers Tekniska Högskola med bidragande kunskap både inom datakommunikation och Visual Basic.

Vi vill också tacka Joakim Karlsson, IT-tekniker på Aspa bruk, för att ha ställt upp med att återansluta internet till vår dator i kontrollrummet och dessutom sparat undan de bilder som tagits på sotblåsarlansen uppe i Aspa.

Slutligen vill vi även tacka vår handledare Manne Stenberg, Institutionen för signaler & system på Chalmers, för en guidning i bildanalysvärlden.

Göteborg maj 2013

Sofia Samuelsson
Carin Collinder

Sammanfattning

Sodapannor används i pappersbruk främst för att ta hand om biprodukter från pappersmassatillverkningen. Sodapannans effektivitet är starkt beroende av beläggningar från aska som sker på pannans värmeytor vid förbränning. Företaget SootTech AB i Göteborg, har utvecklat en patenterad sotningsteknologi. I samband med detta identifierade de en möjlighet, att kunna mäta förbränningsegenskaper utifrån fotografering och bildanalys av utseendet på en sotblåsarlans. Vid sotning körs lansen in i pannan och blåser rent med ånga. Lansens yta utsätts då för aska. Askbeläggningarnas utseende i form av mängden rödaktiga fläckar kan avgöra hur effektivt sodapannan körs. Syftet var att ta fram en algoritm som kommer lägga grunden för utveckling av ett analysprogram som skall mäta mängden rödaktiga fläckar på fotografier som tas tidslöpande på en sotblåsarlans. Ett analysprogram togs fram på SootTech AB och på Aspa bruk hämtades de tagna fotografierna. Analysprogrammet skapades i programspråket Visual Basic. Programmet analyserar och räknar mängden rödaktiga pixlar som representerar de rödaktiga fläckarna på den tagna bilden. Därför visar resultatet, i form av en graf, procentuell andel rödaktiga pixlar per bild. En möjlighet till verifiering av grafens värden finns i analysprogrammet. Detta resultat har stor betydelse för utveckling av ett verktyg där operatörspersonalen vid en sodapanna kan se en graf som speglar förbränningen. När grafen visar sämre värden kan operatörspersonalen välja att sota oftare eller ändra förbränningsinställningarna. Detta medför att det aldrig sotas i onödan.

Abstract

Recovery boilers are used to take care of byproducts after paper manufacturing. Recovery boilers efficiency is strongly dependent by coatings from ash which occurs on the boiler heat surfaces. The company SootTech AB in Gothenburg wants to measure combustion characteristics through image analysis of ashes in the recovery boilers. Photographs on a lance are taken and analyzed. The lance is used for cleaning the surfaces inside the pan. The lance enters the pan and blows with steam on the boiler surfaces. The lance surface is then exposed to ashes. The coatings appearance in terms of the amount of red spots can determine how effective the recovery boiler runs. The purpose was to create an algorithm which would be the base for an analysis of red spots on pictures of a recovery boiler's lance. Because of the high temperatures, red spots occur on the ashes and then the ashes could harden on the surfaces inside the pan and therefore decrease the energy efficiency on the recovery boiler. The purpose was also to make a file transfer possible, from the craft recovery company, Aspa in Munksjö where the pictures are taken, to the office in Gothenburg. When an FTP-server was installed and when the internet was working a file transfer was possible. The algorithm was created in Matlab by studying the image properties. The result, which was a prototype program, was created in Visual Basic. The prototype program manages to accomplish an analysis of the red spots and show a diagram which shows the percentage of redness per analyzed image.

Innehållsförteckning

Beteckningar

HISS= High Impact Soot System

IP= Internet Protocoll

DNS= Domain Name System

DUC= Dynamic Update Client

USB= Universal Serial Bus

RGB= Red, Green, Blue

HSI= Hue, Saturation, Intensity

PNG = Portable Network Graphics

BMP =Bitmap

1. Inledning

1.1 Bakgrund

Företaget SootTech AB är en start-up inom Chalmers Innovation och arbetar för att öka energieffektiviteten för massindustrins sodapannor. Sodapannans effektivitet är starkt beroende av igensättningar från aska som sker inuti pannan och förmågan att kunna rengöra pannans innanmäte från askan under drift.
SootTech's huvudprodukt effektiviserar den automatiska rengöringen av pannors innamäte från aska (s.k. sotblåsning). Metoden utgår från bolagets patenterade teknologi s.k. HISS-sotning (High Impact Soot System). Vid sotning körs en lans in i pannan och blåser rent med ånga. Lansens yta utsätts för rökgaser och beläggs med aska. Vid utvecklingen av HISS-teknologin identifierades en möjlighet, att troligtvis kunna mäta förbränningsegenskaper utifrån fotografering och bildanalys av utseendet på den lans som används för rengöringen av pannans innanmäte. Beläggningarnas utseende i form av mängden röda prickar och ”grovhet” på ytan kan avgöra hur effektiv förbränningen har varit.

1.2 Syfte

Att ta fram en algoritm som kommer vara grunden för en utveckling av ett automatiskt analysprogram. Programmet skall mäta mängden rödaktiga fläckar utifrån en löpande automatisk fotografering av en sotblåsarlans. Detta kommer lägga grunden för företagets idé, Voodoo. Dessutom skall en möjlighet för bildöverföring från Aspa Bruk till Göteborg finnas.

Ett utvärderingsverktyg skall utvecklas för möjligheten att verifiera analysen. Detta verktyg skall visualisera hur förbränningen har gått med en graf. Möjligheten skall också finnas att verifiera grafens värde genom att kunna se fotografiet som representerar värdet.

1.3 Avgränsningar

På Aspa Bruk finns endast en kamera som har möjlighet att fotografera en lans. Fotograferingen sker framifrån, vinkelrät sett från lansen. Det finns bara en ljussättning på lansen i form av en lampa.
Det kommer endast att studeras de rödaktiga prickarna på lansen och inte ”grovheten”.

1.4 Precisering av arbetsuppgiften

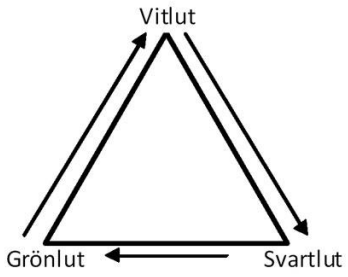
- Att få den befintliga kameran att ta kort kontinuerligt.
- Bilderna som sparas på en dator på Aspa Bruk skall kunna kommas åt på kontoret i Göteborg.
- Hur skall algoritmen skapas som analyserar mängden röda prickar på den tagna bilden?
- I vilken programmiljö skall algoritmen skapas?
- Hur skall värdena sammanställas till en analyserbar graf?
- Hur skall ett visuellt utvärderingsverktyg utvecklas?

2 Teknisk bakgrund

2.1 Vikten av sotning

2.1.1 Sodapanna

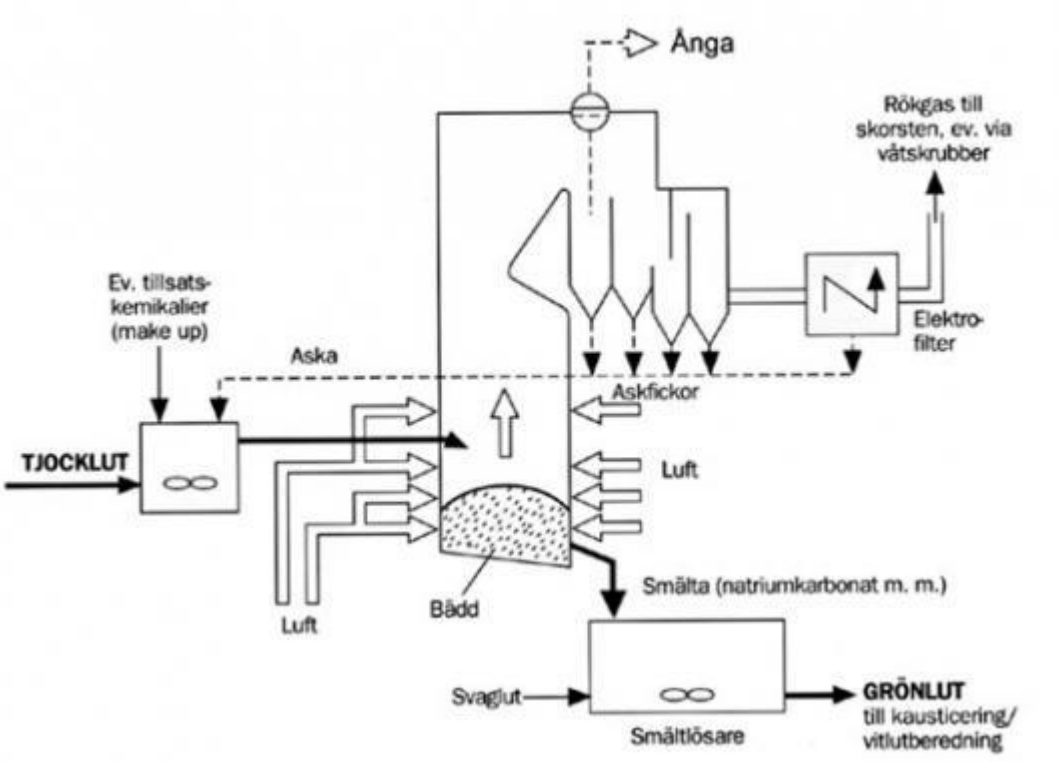
I ett pappersbruk används en så kallad sodapanna för att ta hand om biprodukter från de kokade träflisorna, som sedan skall bli till papper. Sodapannan genererar också el och värme till hela anläggningen och lite till. Allt går ihop i ett näst intill slutet kretslopp. Biprodukterna från de kokade träflisorna innehåller många icke miljövänliga kemikalier. Dessa kemikalier fås då bark och träflis kokas i vitlut tillsammans med vatten. Vitlut är en vitaktig vätskeblandning bestående av natriumsulfid och natriumhydroxid [1]. I trä finns det också lignin, det är det som binder ihop cellulosa och hemicellulosa till fibrer i träet och dessutom gör den karakteristiska trädstrukturen. När träflisorna kokas med vitlut och vatten löses ligninet upp utan att förstöra träfibrerna, vilket är en betydande faktor när man tillverkar papper. Trämassan går sedan vidare för papperstillverkning [2]. Ligninet gör att kokvätskan blir svart och när kokningen är klar kallas blandningen för svartlut. Efter indunstning sprutas svartluten in i botten av pannan som små droppar där den förbränns under syrefattigt förhållande. På grund av detta sker inte en fullständig förbränning av de organiska ämnena i svartluten. Det bildas istället vattenånga, vätgas och kolmonoxid som fullständigt förbränns högst upp i sodapannan [1].
Under förbränningen flyger aska upp i pannan och detta skapar beläggningar.
Sista steget i förbränningen är när svartlutsbädden i botten förbränns. Den höga temperaturen gör att natriumsulfat reduceras tillbaka till natriumsulfid. Det bildas en smälta i nedre delen av bädden som rinner ut ur sodapannan ner i en smältlösare. När smältan blandas med vatten blir blandningen grön och kallas då för grönlut. Grönluten skickas sedan till kausticering, vilket betyder att grönluten kommer vitberedas till vitlut i en rad reaktioner. Vitluten i sig används sedan på nytt. Förloppet visas i *Figur 2.1*.



Figur 2.1: Lutens kretslopp [1]

2.1.2 Sotning

Inuti en sodapanna går det långa rör som innehåller vatten. Vattnet i rören förångas på grund av den höga temperaturen i pannan som fås då svartluten förbränns. Vattenångan kan delvis gå via en turbin och generera el eller så kan den föras ut till ett lokalt fjärrvärmenät. I *Figur 2.2* kan man se hur förbränningsförloppet i en sodapanna går till [1].

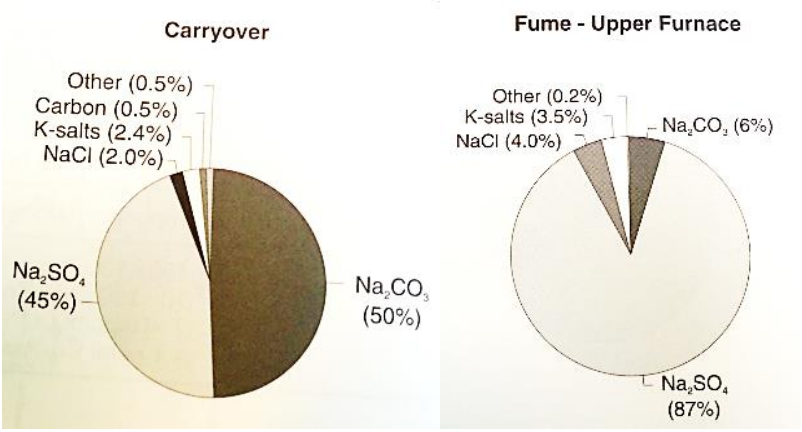


Figur 2.2: En överskådlig bild över hur en sodapanna fungerar. [3]

Då svartluten förbränns flyger det upp aska som sedan fastnar på rören. Detta minskar värmeupptagningsförmågan på rören. Följderna blir mindre el-generering och mindre värme ut på fjärrvärmenätet. Därför behövs rören rengöras med hjälp av sotning. Vid sotning förs lansar in i pannan och sprutar rent rörens ytor med strålar av ånga [4].

2.1.2.1 Beläggningar på värmeytorna i sodapannan

De finns två partiklar som bidrar till nedsmutsning och avlagringar på värmeytorna i pannan. Rökgaspartiklar och överbäringspartiklar. Överbäring innebär att delar av oförbrända svartlutsdroppar flyger med rökgaser upp i pannan och sätter sig som en beläggning på pannrören. Överbäringspartiklar bildas då svartlutsdroppar förångas efter insprutning i pannan. Dessa partiklar sväller upp på grund av att vatten avdunstas. Ju mer dessa partiklar sväller desto snabbare brinner de upp. Men ju mer de sväller desto lättare har de också att ryckas med upp i pannan tillsammans med rökgaserna [5]. Rökgaserna innehåller aska som uppstår då ett oorganiskt salt från smältbädden förångas och sedan kondenserar när det åker högre upp i pannan där temperaturen är lägre [5, 6]. Rökgasaskan har en fluffig och puderaktig konsistens när den lägger sig på pannrören. När temperaturen i pannan ökar kan detta puder sintra, d.v.s. smälta samma och bilda en hård betongliknande konsistens. Denna aska går då inte att sota bort utan kräver annan sorts rengöring. Rökgaspartiklar är också mindre än överbäringspartiklarna och de olika partiklarna har olika uppbyggnad av kemikalier, se *Figur 2.3*.



Figur 2.3: Rökgaspartiklar och överbäringspartiklarna har olika uppbyggnad av kemikalier[6]

Rökgaspartiklarna innehåller mer natriumsulfat (Na_2SO_4) och natriumsulfid (Na_2S), än överbäringspartiklarna. Natriumsulfid har en röd och gulaktig färg, rester av detta kan man se på lansarna som en beläggning. Detta kan kännas igen som rödaktiga fläckar på lansen [5, 6, 7].

2.1.3 HISS-sotning

SootTech har kommit på en patenterad idé som kallas för HISS-sotning. Med denna teknik körs flera sotblåsare samtidigt. Dessutom strålar det endast ut vattenånga på invägen och inte på utvägen. Detta gör man för att kunna sota effektivare och oftare och medför att man sparar energi och får dessutom en mer lättsotad aska [4].

2.1.4 VooDoo

För att kontrollera att pannans värmeytor inte drabbas av överbäring eller att askan har stelnat är det driftingenjörens uppgift att gå ut och se på sotarlansen. Voodoo tekniken använder sig istället av en kamera som fotograferar sotarlansen när den kommer ut från sodapannan. Fotot visas i kontrollrummet där operatörspersonalen kan titta på kamerabilden. Om överbäring har skett är askan på lansen helt skrovlig och om askan är full med rödaktiga fläckar tyder detta på att askan är på väg att stelna. Med denna vetskap kan operatörspersonalen avgöra hur de ska styra pannan för att få en så effektiv förbränning som möjligt [4].

2.2 Bilders uppbyggnad

En digital bild är uppbyggd av små kvadratiska element som kallas pixlar. Pixel i sig är en förkortning av "Picture element" [8]. Dessa pixlar kan man lätt se som helfärgade kvadrater om man förstorar en digital färgbild på datorn. Tillsammans bildar detta, vad ögat uppfattar som en sammanhängande bild. Ju fler pixlar desto bättre upplösning [9].

2.2.1 Färgmodeller

2.2.1.1 RGB

En modell som används för att bestämma en pixels färg är RGB-modellen. Denna modell bygger på att det mänskliga ögat kan upptäcka kombinationer av rött, grönt och blått vilka är det primära färgerna [10]. En vanlig datorskärm visar oftast pixlar enligt RGB-modellen. På grund av bitdjupet i en RGB-bild så kan varje pixel visa 256 nivåer av varje grundfärg. Med detta går det att nästan återskapa alla färger de mänskliga ögat kan se i verkligheten [11]. Pixlarna som visas har en egen tredimensionell vektor med olika värden av rött, grönt och blått (R, G, B). Denna vektor ger pixeln dess färgegenskaper, där (0, 0, 0) är svart, (255, 255, 255) är vit och (255,0,0) är "ren" röd [10].

2.2.1.2 HSI

En annan modell som kan användas för att definiera färg är HSI-modellen som står för Hue, Saturation och Intensity.
Denna modell bygger också på en tredimensionell vektor. Denna vektor beskriver pixels färg genom värden på nyans, mättnad och intensitet. Nyansen beskriver vilken färg som studeras, mättnaden beskriver hur färgstark färgen är. Lägsta mättnadsvärdet ger en grå färg. Intensiteten beskriver hur ljus eller mörk färgen upplevs vara, ex. mörk- eller ljusröd [10].

2.2.1.3 Val av färgmodell

HSI-metoden är en bra metod vid bildbehandling, då människan har en bra uppfattning av nyans och mättnad. Man kan därför med HSI-modellen enkelt sätta ett värde på exempelvis färgen ljusröd. Detta kan vara svårare om man använder RGB-modellen. En nackdel med HSI- modellen är att ljus-parametern ändras samtidigt som man ändrar på mättnads-parametern. De tre parametrarna i denna modell kan alltså inte ändras fristående. Detta visades tydligt efter tester i programmet Gimp. En anledning till att RGB-modellen valdes berodde på att den används vid digitala bilder och på bildskärmar. Eftersom algoritmen skulle behandla digitala bilder var det naturligt att välja denna modell [12]. Det finns också många program som har färdiga funktioner som läser av pixlarnas RGB-värden.

3 Kamera och bildöverföring

Kameran som var fastmonterad vid en av lansarna intill sodapannan på Aspa Bruk skulle ta bilder. Istället tog den bara kort varje gång kamerans IP-nummer uppdaterades i webbläsaren på en dator i kontrollrummet. Detta var förinställt från återförsäljaren. För att kunna uppdatera kamerans IP och samtidigt spara kortet som togs, användes ett tillägg webbläsaren Firefox, nämligen Timelapse [13]. Då kameran var kopplad via brukets ethernet kunde detta ske utan att vara uppkopplad på internet. Men för att komma åt bilderna från Göteborg behövdes ett mobilt bredband. Uppkopplingen var dålig men fungerade vid första besöket på Aspa bruk. För att göra bildöverföringen möjlig installerades en FTP-server, Filezilla-server [14]. Ett problem skulle uppstå varje gång det mobila bredbandet byter IP-nummer, därför utnyttjades tekniken Dynamic DNS (Dynamic Domain Name System). För att lösa detta problem installerades programmet DUC, Dynamic Update Client, från No-IP. Där valdes ett domännamn som skulle representera det mobila bredbandets IP-nummer även då det byttes.

Tester visade att det gick att komma åt mappen med bilderna.

Bara efter någon dag hade kontakten till datorn brutits. För att undvika att detta skulle hända igen gjordes ytterligare ett besök på Aspa bruk. Då monterades en signalförstärkande antenn upp. Internetsignalen blev betydligt bättre. Även här testades om det var möjligt att komma åt mappen med bilderna, vilket det också gjorde. Programmet Teamviewer installerades på datorn på Aspa bruk, då internetuppkopplingen denna gång var mycket bättre. En fjärranslutning till datorn var då möjlig ifall några inställningar skulle behövas ändras på.

Kameran hade tagit tillräckligt många kort för att kunna testa en algoritm och medtogs därför till Göteborg på ett USB-minne.

Dock visade det sig senare att kontakten till datorn ännu en gång hade brutits. En riktig felsökning till varför detta hände gjordes inte.

4 Bildanalys

Lansen utsätts för aska när den åker in i sodapannan. En studie av askan på lansen kan göras för att se hur förbränningen har gått. Om askan exempelvis visar mycket rödaktiga fläckar så betyder det att temperaturen varit för hög. Detta ökar risken för att askan ska sintra på värmesytorna. Det som har störst betydelse är mängden fläckar, inte hur mörka de är. För att kunna undvika att askan sintrar på värmeytorna gäller det att sota rent med jämna mellanrum. Man vet inte idag hur man ska kunna optimera sotningen utan att en okulär undersökning av lansen görs. Idag kan ett foto av lansen undersökas, men tanken är att det skall ske automatiskt och visa resultatet i en graf. Därför analyserades de digitala bilderna med vetskap om hur en bild är uppbyggd.

Figur 4.1: lans med rödaktiga fläckar

4.1 Algoritm i Matlab

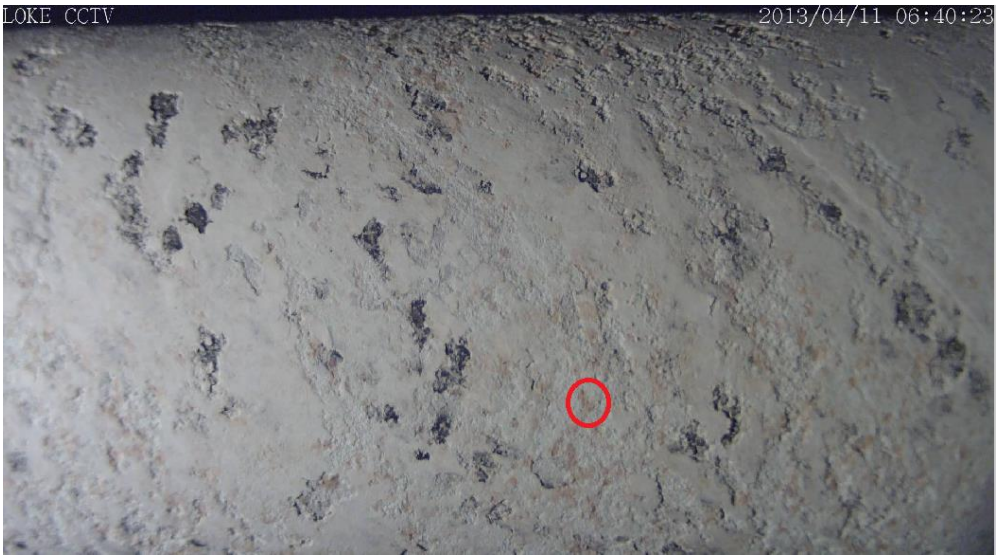
Då Matlab är ett kraftigt beräkningsprogram och dessutom kunde läsa av bilders RGB-värden användes detta för att skapa en algoritm.

4.1.1 Rödaktiga fläckar

De rödaktiga fläckarna representerar de egentligen mer brungula prickarna i Figur 4. För att undersöka vilka RGB-värden dessa rödaktiga fläckar hade, användes några funktioner i Matlab. En funktion som läste in bilder, imread, och en som tog reda på RGB-värdet för de manuellt valda punkterna i bilden, impixel, se *Figur 4.2*. Se matlab-kod i BIL. 1.

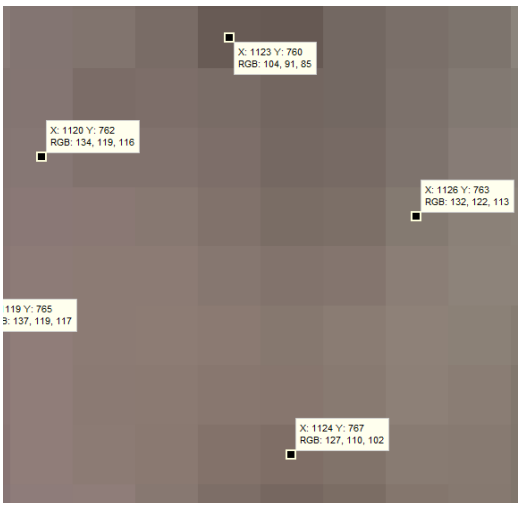
Figur 4.2: Funktionerna "imread" och "impixel" i Matlab. Efter att ha anropat "impixel" kommer den valda bilden upp och man kan klicka på ett antal punkter i bilden och sedan få utskrivet dess RGB-värden.

Efter man kört dessa två funktioner gavs möjligheten att förstora den inlästa bilden och få reda på exakt vilket RGB-värde som representerade vilken pixel, se *Figur 4.3- Figur 4.5*.



Figur 4.3: Ett utmarkerat ett rött område på sotblåsarlansen

Figur 4.4: Flera förstoringar av sotarlansen från det markerade området i Figur 4.3. Pixlarna syns tydligt.



Figur 4.5: Med hjälp av ett verktyg i Matlab kunde också en utsättning av pixlarnas RGB-värden sättas ut.

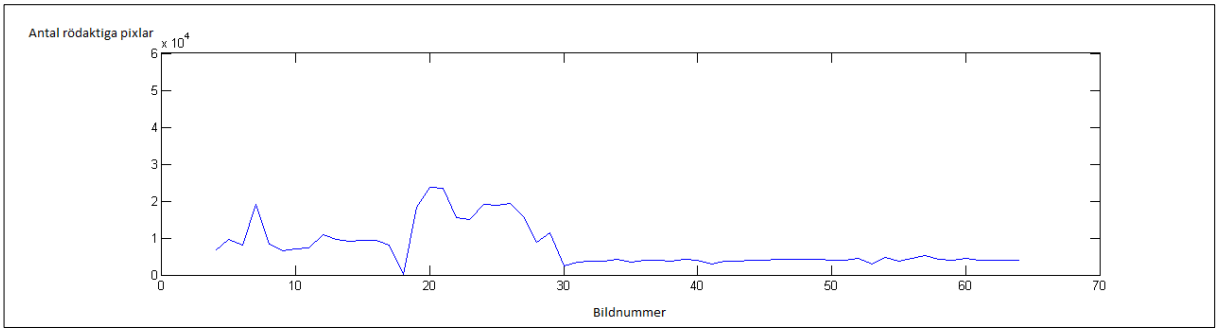
Detta gjordes åtskilliga gånger med olika bilder för att se ett mönster för vilka RGB-värden som representerade det rödaktiga i bilden.

4.1.2 Krav på pixlar

Efter undersökningen kunde vissa krav ställas på pixlarnas RGB-värden. De pixlar som representera de röda fläckarna:

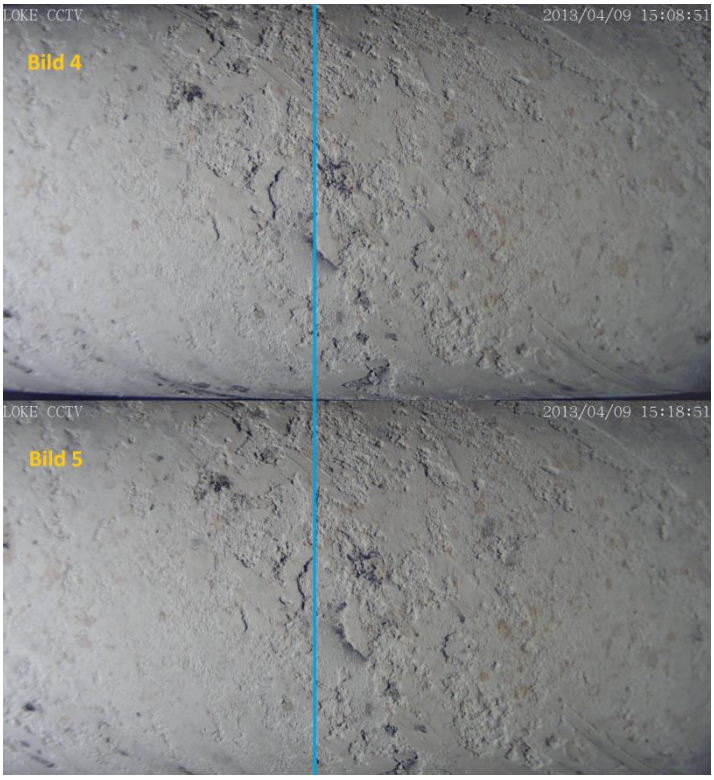
- Det röda värdet måste vara över 100
- Det röda värdet måste vara större än det blå med 10 enheter

Ett program som loopade igenom de bilder som hämtades från det första besöket på Aspa gav följande resultat med kraven ovan se *Figur 4.6*:



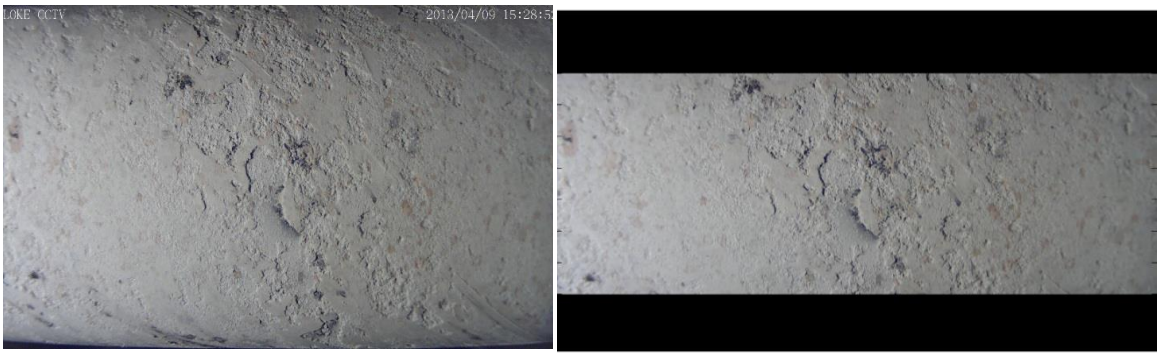
Figur 4.6: Antal rödaktiga pixlar på bild 4 till 64 med kraven röd över 100 och röd större än blå med 10 enheter

Bild nummer 8 till nummer 17 är fotografier på samma lans, se *Figur 4.7*. Detta på grund av att kameran tog kort var 10:e minut och att lansen körs ungefär var 100:e minut. Det är troligtvis på grund av värme och vibrationer som lansen och kameran rör sig lite och därför blir det en skillnad på korten, se *Figur 4.7*. Det borde ändå ge samma antal rödaktiga pixlar.



Figur 4.7: Analys av bild nr.4 och bild nr.5 visar att det är samma bild fast aningen förflyttad. Den blå linjen är en hjälplinje för att förtydliga detta.

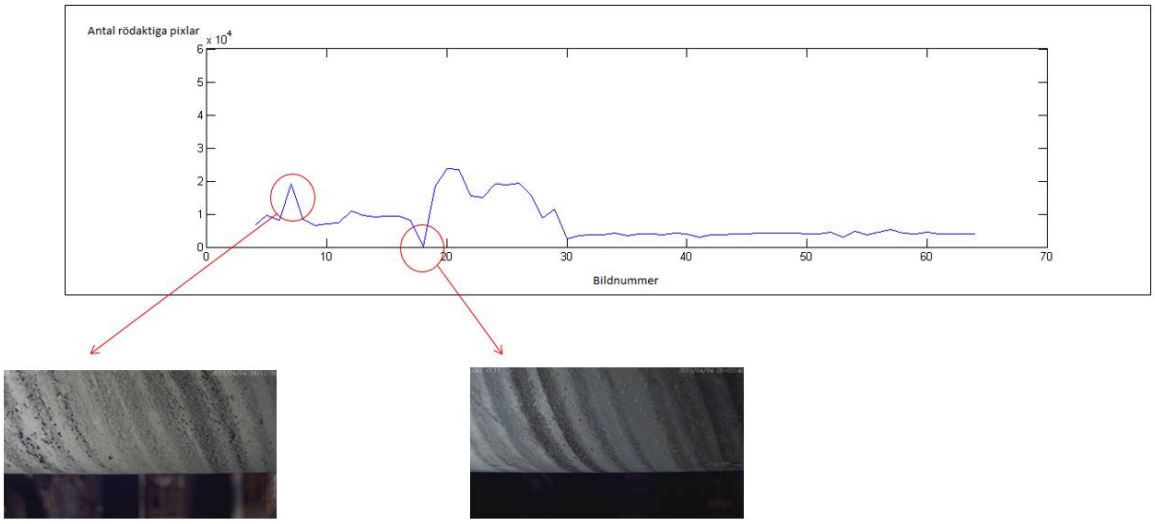
Därför testades det att beräkna en del av bilden för att se om det blir ett jämnare resultat. Det visade sig att det överlag inte har någon större betydelse. Dock valdes det att inte beräkna över och undersidan av kortet då det ofta blir en osäkerhet i algoritmen på dessa ställen. Osäkerheten beror på att det kan bli ett glapp vid över- och undersidan av kortet där bakgrunden syns. Dessutom sparar det tid och datorkraft då man behandlar en mindre del av bilden, se *Figur 4.8*.



Figur 4.8:Den högra bilden visar den del av originalbilden till vänster som programmet analyserar.

Grafen i *Figur 4.6* kan ge ett osäkert resultat när så få bilder analyserades. Därför utfördes test på fler bilder då möjligheten gavs.

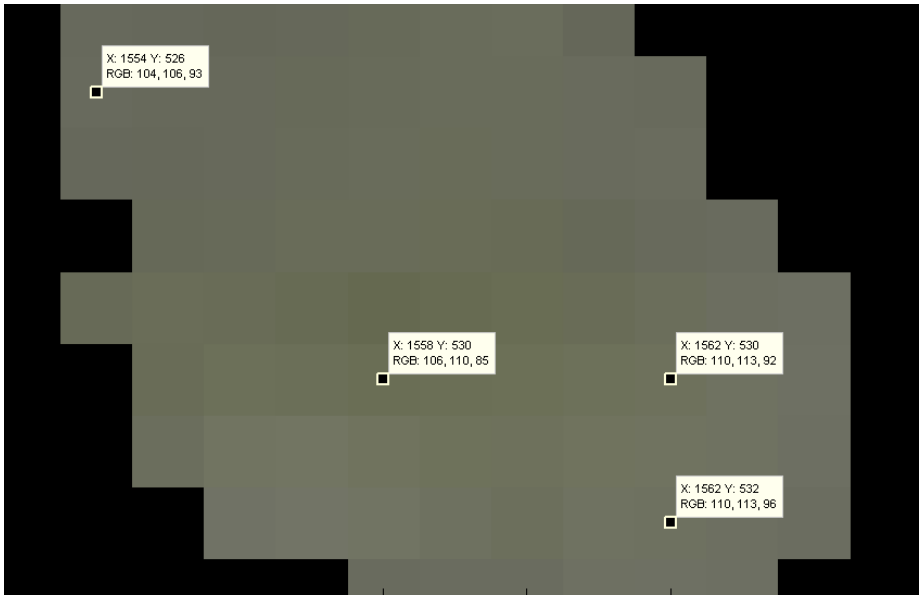
Om man följer grafen i *Figur 4.6* ser man att grafen vid bild numer 18 pikar nedåt. Detta är ett foto på lansens ände och den utsätts aldrig för aska. Detta gör att bilden på lansen är mörk och inte innehåller några rödaktiga fläckar. Detta visas på fler grafer nedan som nedåtgående pikar i grafen, exempelvis i *Figur 4.6*. Bild nummer 7 i *Figur 4.9* är en bild som visar lansens ände men har utsatts för lite aska och är tagen då det är mer ljus i lokalen. Denna del av lansen har inte varit inne i pannan och skall därför inte få ett så högt värde som den har fått.



Figur 4.9: Slutdelen på lansen som inte direkt utsätts för aska, analyserade med krav enligt Figur 4.6

Därför testades det att köra algoritmen på just dessa bilder. Alla pixlar som inte räknades som rödaktiga gjordes svarta se *Figur 4.10*:

Figur 4.10: Bild nummer 7 som sedan gör alla pixlar som inte räknas med svarta.



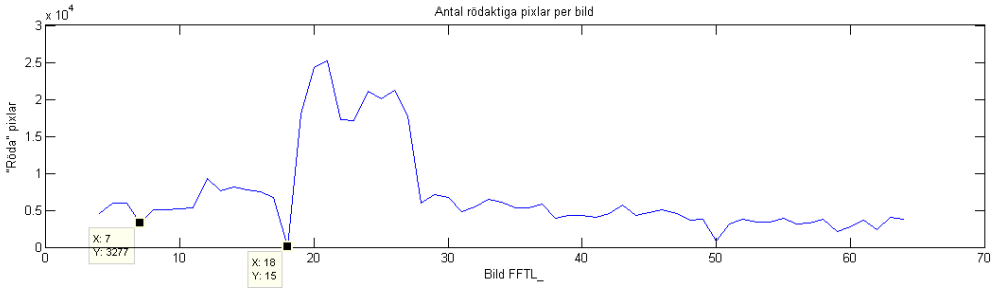
Figur 4.11: Förstoring av bild nummer 7 visar att det är mer grönaktigt än rödaktigt, därför behövs ytterligare krav.

Det visade sig efter förstoring, se *Figur 4.11*, och undersökning av pixlarnas RGB-värden att det behövdes ett krav för den gröna delen i RGB-värdet nämligen:

- Röd måste vara större än grön med 3 enheter.

Figur 4.12: Bild nummer 7 där alla pixlar som inte uppfyller alla tre krav görs svarta

Detta för att inte få med de grönaktiga fläckarna, se *Figur 4.12*. Grafen, *Figur 4.13*, visade nu:

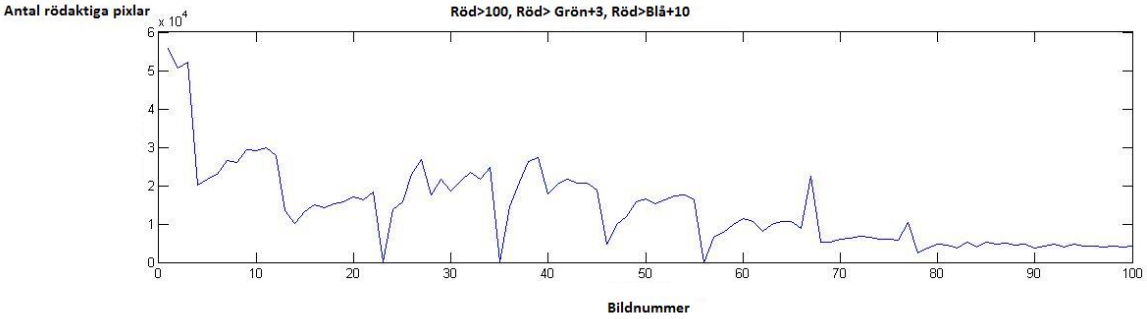


Figur 4.13: Antal rödaktiga pixlar på bild 4 till 64 med kraven röd över 100 och röd större än grön med 3 enheter och röd större än blå med 10 enheter

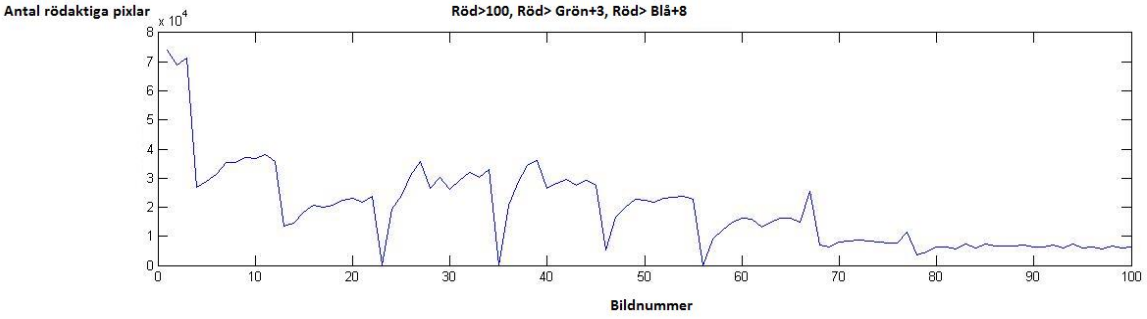
De tre kraven för att en pixel skulle räknas med i programmet blev nu:

- Det röda värdet måste vara över 100
- Det röda värdet måste vara större än det gröna med 3 enheter
- Det röda värdet måste vara större än det blå med 10 enheter

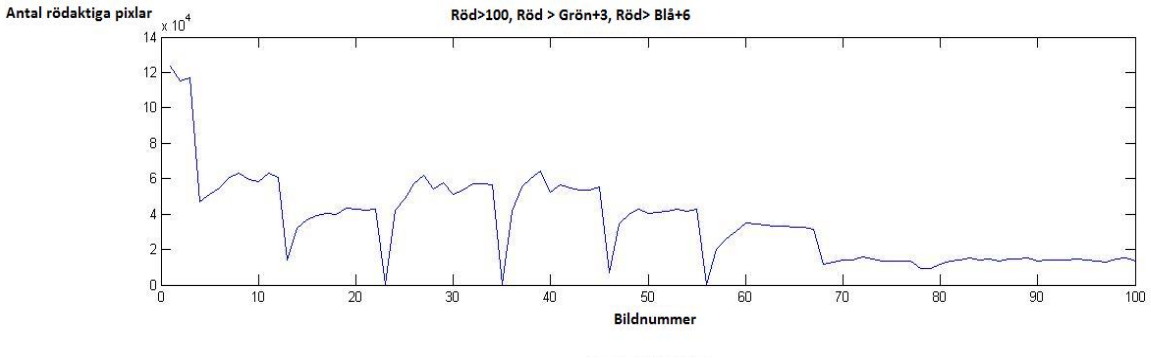
Efter att ha fått nya och fler bilder från andra besöket på Aspa, testades algoritmen på nytt. Denna gång testades att dessutom ändra kraven för RGB-värdena. Detta för att göra så att de bilder som visade i likadana fotografier fick så lika värde som möjligt och för att de bilder som inte skall visa högt värde inte heller gör detta. Se *Figur 4.14- Figur 4.16*.



Figur 4.14: Tre krav: Röd större än 100, röd större än grön med 3 enheter, röd större än blå med 10 enheter.



Figur 4.15: Röd större än 100, röd större än grön med 3 enheter, röd större än blå med 8 enheter.



Figur 4.16: Röd större än 100, röd större än grön med 3 enheter, röd större än blå med 6 enheter

Efter justering av de olika kraven kan man se att platåer uppstår. Det är vad som önskas då platåerna mellan körningarna (pikarna nedåt) är kort på samma lans med samma antal rödaktiga pixlar, se *Figur 4.16*.

4.1.3 Test av algoritm

Efter att algoritmen tagits fram i Matlab var den tvungen att testas. Algoritmen visade ett bra resultat för bilderna på lansen. Testet utfördes för att ta reda på om algoritmen även kunde beräkna hur många rödaktiga pixlar en helt annan typ av bild hade. Detta för att verifiera att algoritmen alltid gav ett trovärdigt resultat.

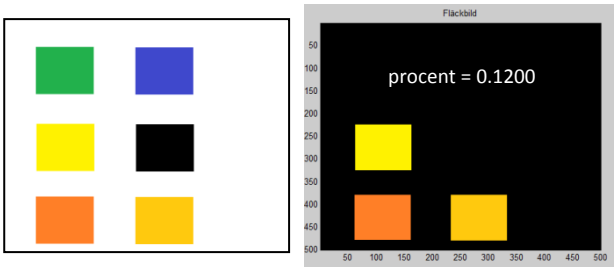
I programmet Paint ritades en vit bild med storleken 500x500 pixlar, i bilden ritades ytterligare några kvadrater in med en storlek på 100x100 pixlar. Kvadraterna färgades med valda RGB-värden. En färgad ruta var 4 % av den vita bakgrunden, se *ekv.1*.

$$\frac{100 \cdot 100}{500 \cdot 500} = \frac{10000}{250000} = 0{,}04 \qquad (ekv.1)$$

Matlab-koden ändrades så att resultatet skulle skriva ut hur stor procent av bilden som var täckt av rödaktiga fläckar istället för antalet pixlar som var röda.

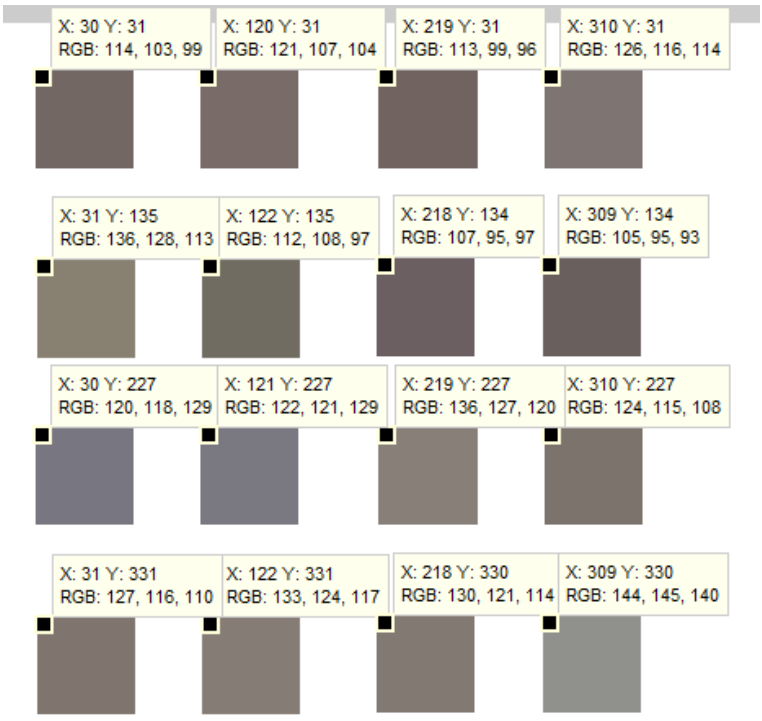
Bilderna sparades med filformatet PNG, Portable Network Graphics. Detta för att png komprimerar bilder nästan utan förluster, så kallad icke-förstörande komprimering. [15] För att bilderna inte skulle komprimeras och fortfarande ge rätt procentuella resultat sparades bilderna i filformatet PNG. Bilderna testades i Matlab och visade rätt procentuella värde för de uträknade kvadraterna. Därför sparades bilderna i PNG-format istället för BMP, bitmap, som annars vore det optimala.

Nedan i *Figur 4.17* kan man se att algoritmen lyckas med att beräkna hur många procent av kvadraterna som räknas med. I *Figur 4.17* kan man också se att algoritmen inte beräknar med de blå och de gröna färgerna. Detta visar att algoritmen brister med att utesluta den gula färgen och de orangea färgnyanserna. Detta blir dock inget problem när det gäller analysen av de rödaktiga fläckarna på lansen då dessa bilder inte har visat sig innehålla några extrema gula färger utan bara gråare toner av dess av dessa färger.



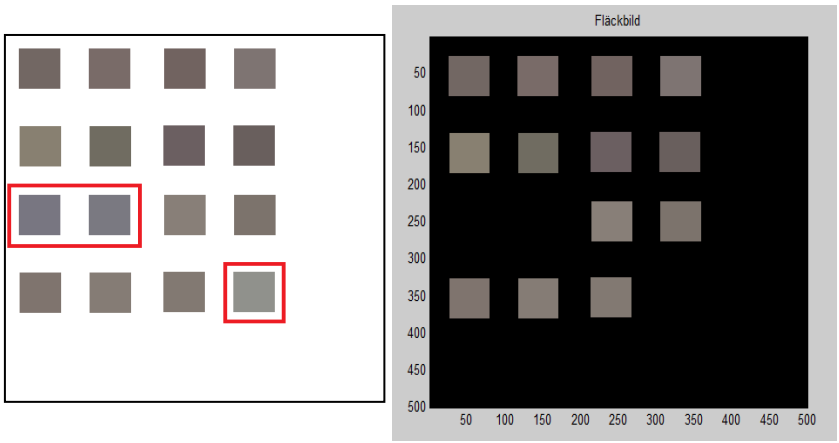
Figur 4.17: Algoritmen beräknar inte med de blå och gröna

Därefter gjordes ett test där kvadraterna har fått färger som finns med på fotografierna av lansen, se *Figur 4.18*. För att få fram färgerna lästes en bild på lansen in i Matlab, sedan användes funktioner i Matlab som läser in pixlar och visar dess RGB-värden.



Figur 4.18: Färger som finns med på bild av lans.

RGB -värdena valdes från de rödaktiga, gulaktiga, rosagråa och blålila områdena. Sista kvadraten har ett RGB-värde som är taget ifrån den gråa ytan kring de rödaktiga fläckarna från ett fotografi.
Nedan i *Figur 4.19* kan man se vilka kvadrater algoritmen inte räknade med. Den västra bilden i *Figur 4.19* är samma som *Figur 4.18*, men utan utmarkerade RGB-värden.



Figur 4.19: Kvadrater som algoritmen inte räknade med

De tre kraven för att en pixel skulle räknas med i programmet var:

- Det röda värdet måste vara över 100
- Det röda värdet måste vara större än det gröna med 3 enheter
- Det röda värdet måste vara större än det blå med 6 enheter

I *Figur 4.19* kan man se att algoritmen inte tog med rutorna med blålila färg och rutan med den ljusgråa färgen. Dessa rutors RGB-värden ses i *Figur 4.18*. De kvadrater som uppfyllde kraven för RGB-värdena hade färger som låg åt det rödgråa hållet.

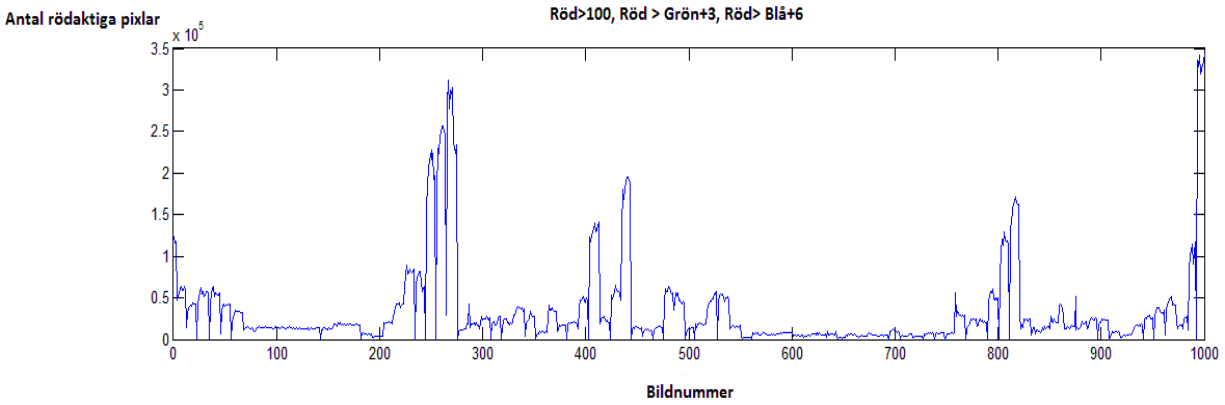
Algoritmen brister med att utesluta den gula färgen, men resultatet blir tillräckligt bra då bilderna på lansen inte har visat sig innehålla någon ren gul färg.
Skulle det vara så att man inte vill att program skall räkna med de gula nyanserna, är det enkelt för en insatt programmerare att lägga till nya krav i algoritm som löser detta. Därför godtogs algoritmen trotts sina brister.

4.1.4 Test av algoritm av flertalet bilder

De slutgiltiga kraven på RGB-värdena blev efter justering följande:

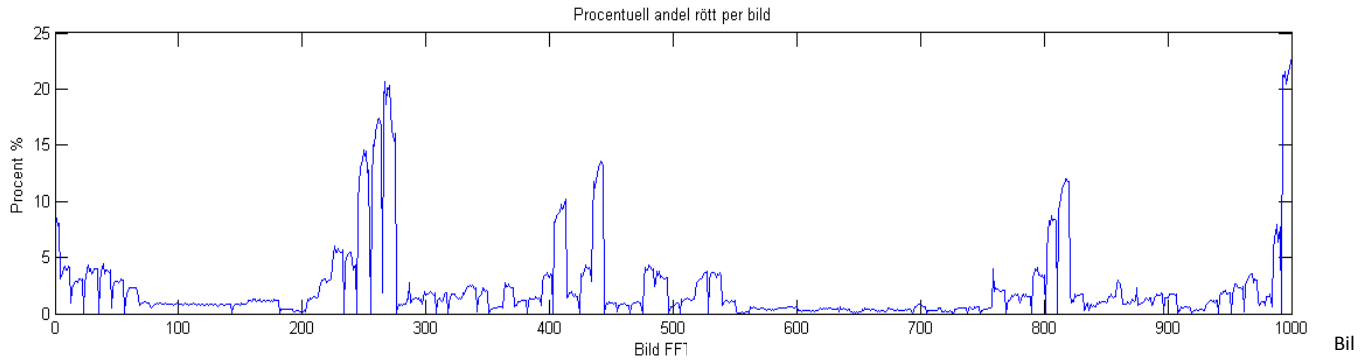
- Röda värdet skall vara högre än 100
- Röda värdet skall vara högre än det gröna värdet med 3 enheter
- Röda värdet skall vara högre än det blå värdet med 6 enheter.

När kraven justerats i koden kördes programmet på 1000 bilder, se *Figur 4.20*.



Figur 4.20: Röd större än 100, röd större än grön med 3 enheter, röd större än blå med 6 enheter.

Resultatet i *Figur 4.20* visar en spegling över hur förbränningen i sodapannan har gått under en veckas tid. Istället för att visa antal pixlar ändrades koden till att visa hur många procent av bilden som beräknas vara rödaktiga pixlar.



Figur 4.21: Procentuell andel rödhet per bild.

4.2 Utveckling av program i Visual Basic

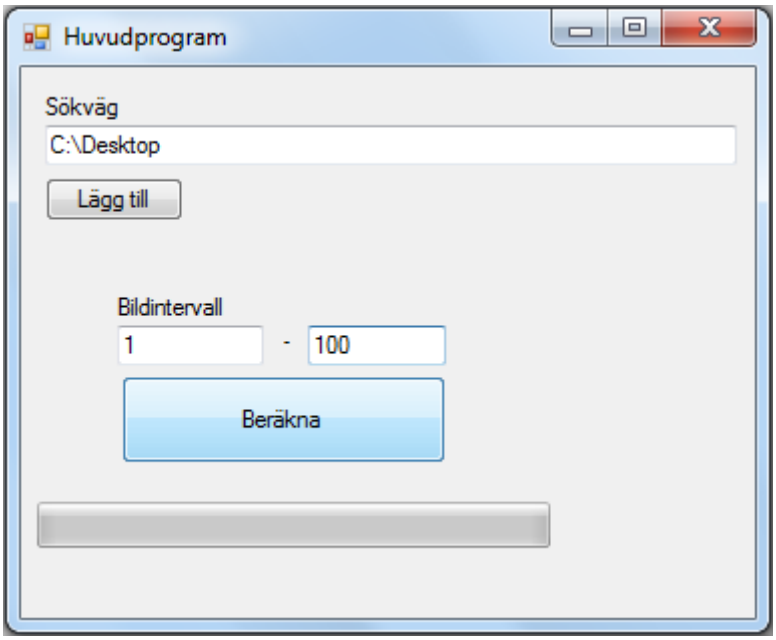
Företaget SootTech AB hade önskemål om att få ett programskelett skapat i Visual Basic, som visar att algoritmen fungerade. Detta för att lätt kunna förstå koden och vidareutveckla programmet till en riktig produkt.

Visual Basic kan beskrivas som ett användarvänligt programmeringsspråk där man enkelt kan skapa ett grafiskt gränssnitt utan att behöva programmera det.

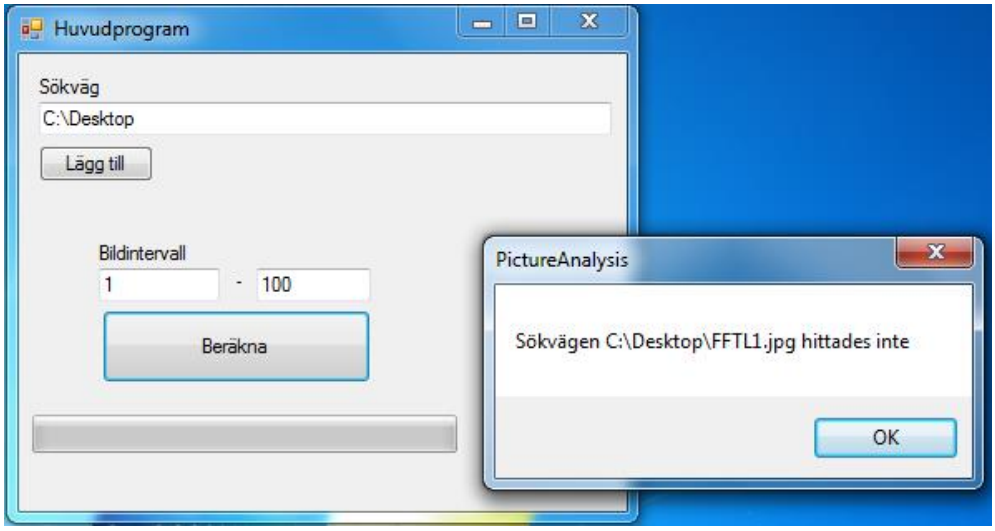
När algoritmen väl var testad gjordes en översättning från Matlab-kod till Visual Basic-kod. Visual Basic hade precis som Matlab en klar funktion som läste in bilder och som läste av varje pixels RGB-värde. I BIL. 2 visar den färdigställda koden. Där visas GetPixel som är den funktion i Visual Basic som hämtar pixlarnas RGB-värden. Koden förklaras noggrant vad som görs. Det förklaras hur man gör för att kunna koppla ihop

knappar i det grafiska programmet med koden och hur man gör för att skriva ut grafer med mera.

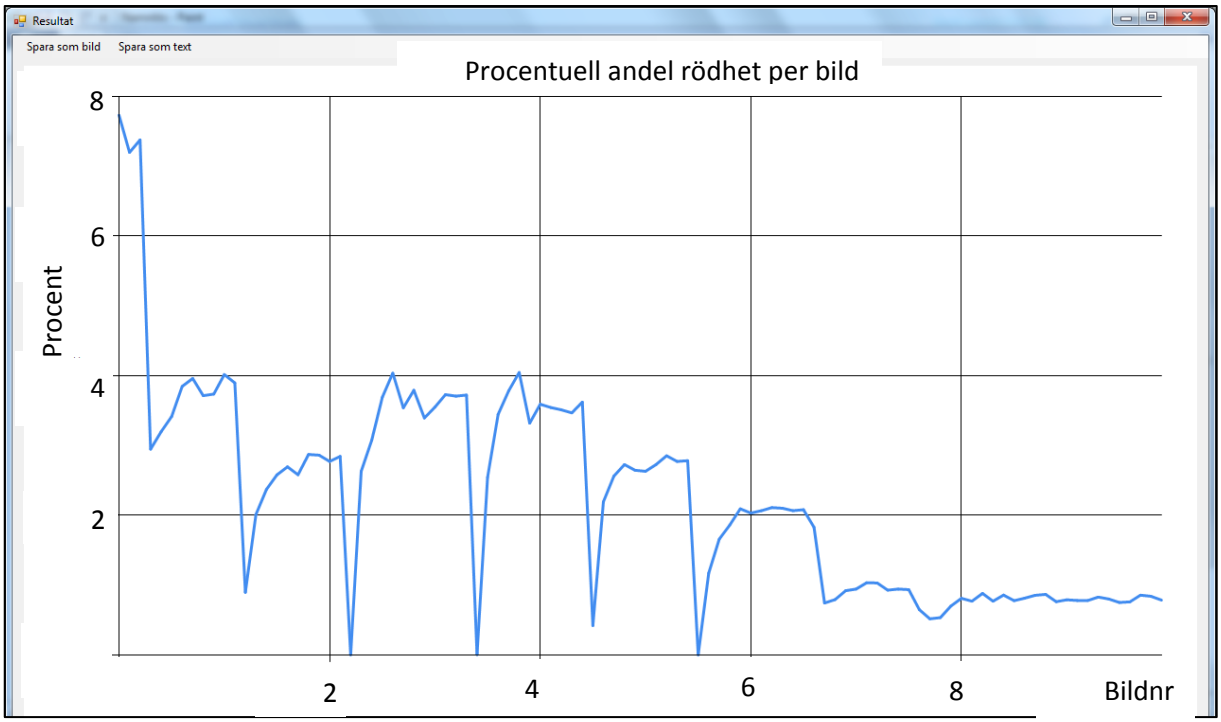
Ett grafiskt gränssnitt utvecklades där man lätt kunde få fram en uträknad graf och verifiera grafen med de faktiska bilderna. Då en graf hade visats gick det också att spara den aktuella grafen som en bild. Det gick dessutom att spara de uträknade värdena som en textfil enligt önskemål. *Figur 4.22- Figur 4.26*, beskriver och visar hur programmet fungerar:



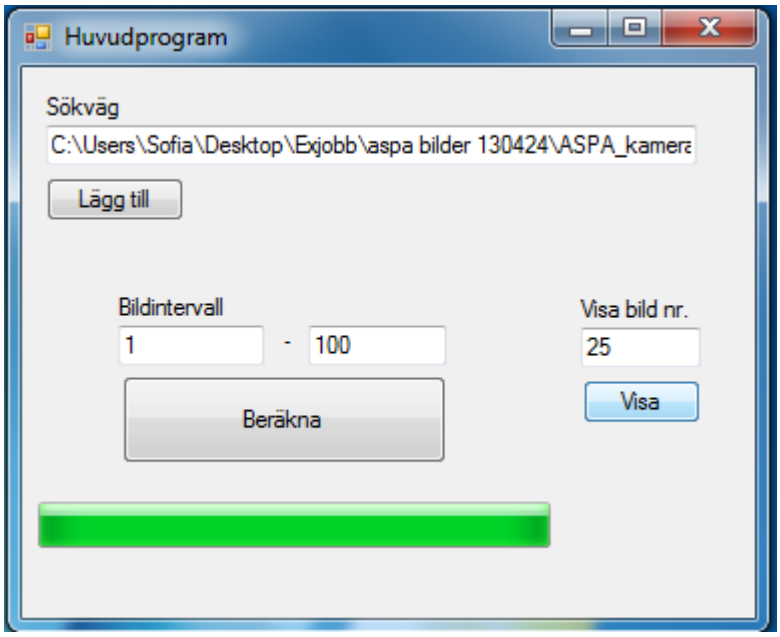
Figur 4.22: Huvudprogrammet. Sökvägen till fotografierna från kameran skall läggas till. Vilka bilder som är intressanta att beräkna mängden rödhet i anges i Bildintervallet. Progressbaren visar att programmet är igång efter ett klick på Beräkna-knappen.



Figur 4.23: Programmet visar varningar om de inte hittar bilderna på angiven sökväg.



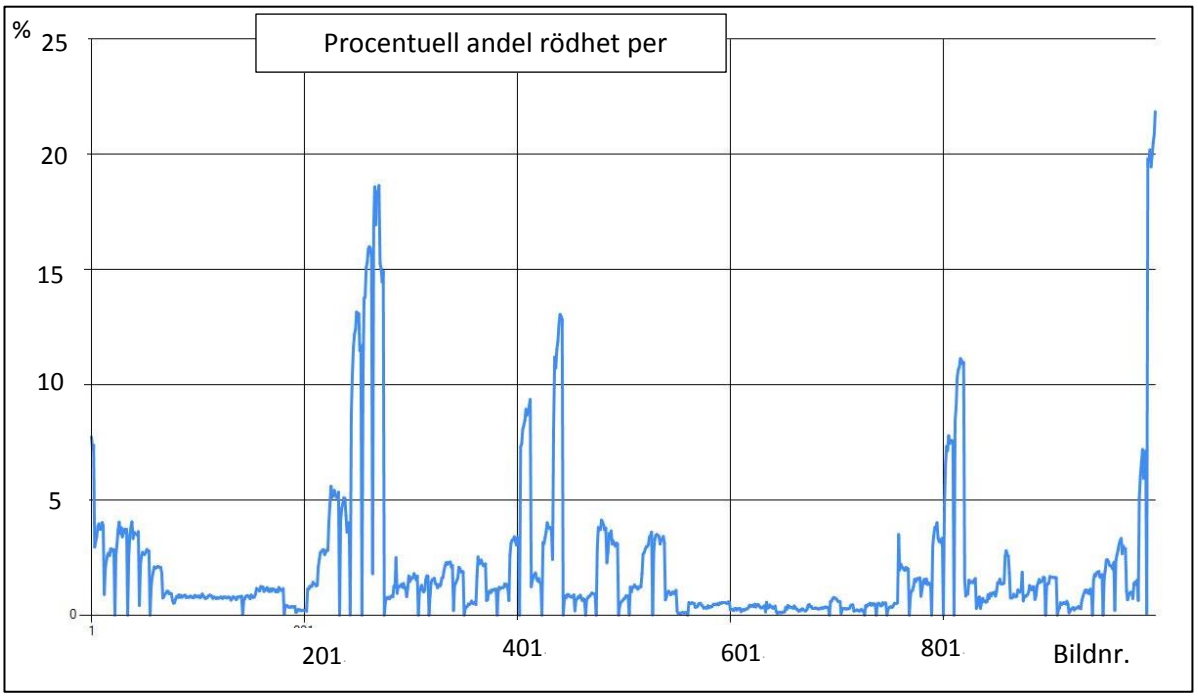
Figur 4.24: Efter att programmet kört igenom alla bilder inom bildintervallet visas grafen. Uppe i vänstra hörnet finns möjlighet att spara grafen som en bild eller att spara värdena som grafen visar som en textfil.



Figur 4.25: Då programmet har visat grafen och man skulle vilja verifiera något värde i grafen med respektive bild, går det nu att skriva in bildens nummer i en ruta i programmet.

Figur 4.26: Den angivna bilden visas nu i ett nytt fönster.

Programmet lyckades att gå igenom flertalet bilder, detta tog längre tid i Visual Basic jämfört med Matlab då Visual Basic inte är ett optimalt beräkningsprogram.



Figur 4.27: Procentuell andel rödhet per bild. Programmet beräknar 1000 bilder.

5 Slutsats och diskussion

För att få den befintliga kameran på Aspa bruk att ta kort kontinuerligt installerades Firefox's tillägg Timelapse. Detta sparar undan korten som tas kontinuerligt med ett inställt tidsintervall. Att ge datorn på Aspa bruk internet och installera DUC som löser att IP-numret uppdateras och dessutom installera en FTP-server, löser bildöverföringen om internet är tillgängligt. I nuläget fungerar inte bildöverföringen. Problemet kan vara att DUC-programmet inte fungerar eller att internet tappar kontakt, detta behövs undersökas ytterligare för att lyckas med bildöverföring.

Det hittades snabbt ett sätt för hur algoritmen skulle utformas för att analysera mängden rödaktiga fläckar. Algoritmen baseras på att utvärdera pixlarnas färg och räkna med de pixlar som uppfyller RGB-kraven för en rödaktig fläck. RGB-kraven blev följande:

- Röda värdet skall vara högre än 100
- Röda värdet skall vara högre än det gröna värdet med 3 enheter
- Röda värdet skall vara högre än det blå värdet med 6 enheter.

Den slutgiltiga programmiljön som programmet skrevs i var Visual Studio, där programspråket Visual Basic valdes. Detta på grund av att företaget SootTech AB hade önskemål om detta. Programmet baseras på den framtagna algoritmen och resultatet visas som en graf som anger procentuell andel rödhet per bild. Detta kan man jämföra med hur många procent av bilden som täcks av rödaktiga fläckar och kan analyseras av en operatör. Grafens värden kunde verifieras genom att visa den verkliga bilden för respektive värde. Dessutom kan grafen sparas som en bild och dess värden kan sparas som en textfil. Programmet kan dock vidareutvecklas och förbättras. Något som exempelvis kan utvecklas är en avbrytknapp, som kan stoppa beräkningsprocessen. Progressbaren i *Figur 4.21* kan utvecklas så att man dessutom ser vilken bild som analyseras. Om en vidareutveckling av programmet görs, bör också en undersökning av algoritmens beräkningskrav göras då den räknar med gula färger.

Referenser

[1] Redoxreaktioner – viktig för återvinning av kemikalier i massaindustrin, Skogsindustrierna, Stockholm 2013, http://www.skogsindustrierna.org/framtid/utbildning/gymnasiet/kemilaromedel/redox/teoritext_5, (Acc 2013-04-29)

[2] Huvudkomponenterna i trä: cellulosa, hemicellulosor och lignin, Forskning i Fokus, Helsingfors 2009, http://www.edu.fi/forskning_i_fokus/inte_bara_brader_och_papper_biorafffinering_av_trad/bioraffinering_till_naturenliga_biomaterial_och_biokemikalier, (Acc 2013-05-21)

[3] Sulfatmassafabrikens kemikalieåtervinning, Skogssverige 2012, http://skogssverige.se/node/38653, (Acc2013-04-29)

[4] Sodapannor, SootTech AB, Göteborg 2013, http://www.soottech.se/sodapannor/, (Acc 2013-04-29)

[5] Forssén M. Backman R. Wallén, J. Hupa M: Flygaskans sammansättning och nedsmutsande tendens i sodapannan, Värmeforsk Service AB, Åbo 1999:

[6] Adams T. N. James Frederick Wm. Grace T. M. Hupa M. Iisa K. Jones A. K. Tran H: Kraft Recovery Boilers, Tappi press, Atlanta 1997.

[7] Industrial Chemicals-Sodium Sulphide Na2s, IndiaMART 2000, http://www.indiamart.com/rudraenterprise/industrial-chemicals.html, (Acc 2013-05-21)

[8] Picture Element, Dictionary.com 2010, http://dictionary.reference.com/browse/picture+element, (Acc 2013-05-08)

[9] Lite teori: Vad är en pixel, PIM-femma 2009, https://sites.google.com/site/pim5bn/vad-aer-en-pixel-, (Acc 2013-05-07)

[10] Sonka M. Hlavac V. Boyle R: Image Processing, Analysis, and Machine Vision, Brooks/Cole Publishing Company, Pasific Grove 1999.

[11] Kläppe B. Sjögreen C. : *Adobe Photoshop CS3 Grundkurs*, Skandinaviska Databöcker AB, 2007

[12] Olika kulörsystem, Lars Ekdahl 2001, http://www.ekdahl.org/kurs/kalibrera_psp/kolorsystem.htm, (Acc 2013-05-16)

[13] Timelapse for Firefox, Add-Ons mozilla Firefox 2013, https://addons.mozilla.org/en-US/firefox/addon/firefox-timelapse/, (Acc 2013-05-07)

[14] FileZilla- The free FTP solution, FileZilla 2013, https://filezilla-project.org/, (Acc 2013-05-07)

[15] Portable Network Graphics (PNG) Specification (Second Edition), W3C Recommendation 2003, http://www.w3.org/TR/2003/REC-PNG-20031110/#4Concepts.PNGImageTransformation, (Acc 2013-05-14)

BILAGA 1

Matlab. Antal rödaktiga pixlar för en bild: pixel.m

```
        function [bild,numpix]= pixel(bild)
%skapar filnamnet
bild = int2str(bild);
name = strcat('C:\Users\Sofia\Desktop\Exjobb\aspa bilder 130424\ASPA_kamera_bilder\FFTL',bild,'.jpg');
%skriv meddelande
disp (strcat('Calculating colors of :',name, ': '))
%läser in bild
RGB= imread(name);
        %variabler
rmin=100;            %lägsta värdet på röd för att det ska antas vara en fläck
treshold=4;                          %tröskelvärde för skillnaden mellan röd och blå
tresholdgreen=3;   %tröskelvärde för gröna delen
numpix=0;        %antal rödaktiga pixlar kommer räknas upp i denna variabel
%loopar kolumner
for c=1:1:1920       %kör en ruta mellan pixlarna med koordinater (c)X = 1 till 1920 och (r)Y = 100
                              %till 900
   %loopar rader
   for r=100:1:900
        %om pixel uppfyller krav, räkna den som hittad
        %kraven är att röd >100 och röd>blå+treshhold och röd > grön + tresholdgreen
         if (RGB(r,c,1)> rmin) && (RGB(r,c,1)> RGB(r,c,3)+ treshold) && (RGB(r,c,1)>
                          RGB(r,c,2)+tresholdgreen)
              numpix= numpix+1;
         end
   end
end
%skriv ut antal hittade pixlar
numpix;
```

Matlab. Graf - Antal rödaktiga pixlar per bild: graf.m

```
X=[];
Y=[];
%Z=[];
num = 0;    %dessa variabler finns i pixel.m också, men måste skrivas här med för att det skall
            %fungera
numpixels = 0;  %antal rödaktiga pixlar kommer räknas här
minpic=1;       %vilken bild som skall den skall börja på
maxpic=100;   %upp till vilken bild som skall beräknas
for i=minpic:1:maxpic
          [num,numpixels] = pixel(i);
          Y = [Y numpixels];        %matris, y-värdena i grafen, antal rödaktiga pixlar
end
X = minpic:1:maxpic;              %x-värdena i grafen, sträcker sig till och med den bild vi valt
plot(X,Y)
xlabel('Bild FFTL_')
ylabel('"Röda" pixlar')
title('Antal rödaktiga pixlar per bild')
```

Matlab. Gör alla pixlar svarta som inte räknas som rödaktig: pixelDetect.m:

```
function [bild,numpix]= pixelDetect(bild)
%skapar filnamnet
bild = int2str(bild);
name = strcat('C:\Users\Sofia\Desktop\Exjobb\aspa bilder 130424\Aspa_Kamerabilder
      old\FFTL',bild,'.jpg');
disp (strcat('Calculating colors of :',name, ': '))          %skriv meddelande
RGB= imread(name);                          %läser in bild
%variabler
rmin=100;              %lägsta värdet på röd för att det ska antas vara en fläck
treshold=10            %tröskelvärde för skillnaden mellan röd och blå
tresholdgreen=3
numpix=0;
%loopar kolumner
for c=1:1:1920         %kör en ruta mellan pixlarna med koordinater (c)X = 1 till 1920 och (r)Y = 100
                       %till 900
  for r= 1:1:199           %dessa forloopar gör alla t utanför intervallet X = 1 till 1920 och (r)Y =
                       % Y=100 till 900, till svart. RGB=(0,0,0)= svart
   RGB(r,c,1) = 0;
   RGB(r,c,2) = 0;
   RGB(r,c,3) = 0;
  end
  for r= 901:1:1080
   RGB(r,c,1) = 0;
   RGB(r,c,2) = 0;
   RGB(r,c,3) = 0;
  end
  %loopar rader
  for r=100:1:900
      %om pixel uppfyller krav, räkna den som hittad
      %kraven är att röd >100 och röd>blå+treshhold och röd > grön + tresholdgreen
   if (RGB(r,c,1)> rmin) && (RGB(r,c,1)> RGB(r,c,3)+ treshold) && (RGB(r,c,1)>
      RGB(r,c,2)+tresholdgreen)
      numpix= numpix+1;
   else                    %Gör alla andra pixlar svarta som inte räknas som rödaktig pixel
      RGB(r,c,1) = 0;
      RGB(r,c,2) = 0;
      RGB(r,c,3) = 0;
     end
  end
end
image (RGB); title('Fläckbild')
%skriv ut antal hittade pixlar
Numpix
```

BILAGA 2.

Visual Basic- kod. Huvudprogram: form_huvud.vb

```
Public Class form_huvud
    Dim bm As Bitmap
    Dim bm_show_pic As Bitmap              'då man vill verifiera bild från graf
    Dim xmax_pxl As Integer                        'bildens maximala pixel-x-värde ex 1920
    Dim ymax_pxl As Integer
    Dim x_pxl As Integer                   'bildens pixel-x-värde
    Dim y_pxl As Integer
    Dim rmin As Integer                    'minsta värde på pixlens röda värde
    Dim tresholdB As Integer                       'tröskelvärde för blå vs röd
    Dim tresholdG As Integer                       'tröskelvärde för grön vs röd
    Dim numpix As Integer                          'antal rödaktiga pixlar i bilden
    Dim minpix As Integer                          'används för att kolla om bilden visar slutet av
        lansen
    Public maxpic As Integer                       'bilder man vill köra på: upp till
        FFTL"maxpic".jpg
    Public minpic As Integer               'från FFTL"minpic".jpg
    Dim j As Integer
    Dim pixelColor As Color                        'kommer att representera pixelns rgb-värden,
        finns för
                                           'att slippa läsa in rgb-värdena flera gånger (optimering)
    Dim red As Byte
    Dim blue As Byte
    Dim green As Byte
    Dim procent As Double
    Public plotY_pxl(0 To 1) As Double     'skapar en array som sedan dimensioneras om på grund
                                           'av att den måste gå att komma åt från form2
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        rmin = 100                         'minsta röda-värde i rgb
        minpix = 200                       'minsta antal röda pixlar (för slutet av lans)
        tresholdB = 6                      'blå tröskelvärde
        tresholdG = 3                      'grön tröskelvärde
    End Sub
    Private Sub calculate_btn_Click(sender As Object, e As EventArgs) Handles calculate_btn.Click
        '-------Testar så att inget annat än siffror kan skrivas in i rutorna-------
        Try
            minpic = txt_bildnr1.Text          'fixa så att det inte går att skriva in något annat än siffror
            maxpic = txt_bildnr2.Text
        Catch ex As Exception
            MsgBox("Endast siffror skall skrivas in")
            Exit Sub
        End Try
        '-------Testar så att bildnummer skrivs i ordning-------
        If minpic >= maxpic Then
            MsgBox("Skriv minsta bildsiffran i vänstra rutan och största bildsiffran i högra rutan")
            Exit Sub
        End If
        '-------Fixar axlarna till grafen-------
        Dim plotX(0 To (maxpic - minpic)) As Integer       'gör en array, för antal bilder,
                                                           'gör så att första bilden börjar på första
                                                           'xvärdet
        ReDim plotY_pxl(0 To (maxpic - minpic))
        j = 0
        pgb_progress.Value = 0
        pgb_progress.Maximum = (maxpic - minpic) + 1       'sätter ut maxvärdet på progressbar
```

```
        '--------Testar om sökvägen finns-------
        For i = minpic To maxpic
            Try
                bm = New Bitmap(txt_filepath.Text + "\" + "FFTL" + i.ToString + ".jpg")
            Catch ex As Exception
                MsgBox("Sökvägen " + txt_filepath.Text + "\" + "FFTL" + i.ToString + ".jpg" + " hittades inte")
                Exit Sub
            End Try
        Next i
        '--------Beräknar antalet rödaktiga pixlar-------
        For i = minpic To maxpic
            bm = New Bitmap(txt_filepath.Text + "\" + "FFTL" + i.ToString + ".jpg")
            xmax_pxl = bm.Width - 1
            ymax_pxl = bm.Height - 1
            numpix = 0
            If ymax_pxl = 1079 Then
                For x_pxl = 0 To xmax_pxl
                    For y_pxl = 200 To 900
                        pixelColor = bm.GetPixel(x_pxl, y_pxl)          'hämtar hem färgen för pixeln
                        red = pixelColor.R                              'lägger det röda värdet i en variabel osv
                        green = pixelColor.G
                        blue = pixelColor.B
                        If (red > rmin) And (red > (blue + tresholdB)) And (red > (green + tresholdG)) Then
                            numpix = numpix + 1                         'Adderar alla rödaktiga pixlar enligt krav
                        End If
                    Next y_pxl
                Next x_pxl
            Else
                MsgBox("Fel bildstorlek")                               'Om bilden är allt annat än 1080pixlar i
                                                                        'höjd, skrivs varningstext - Gör så att
                                                                        'programmet inte räknar vidare sen.
            End If
            procent = (numpix / 1344000) * 100
            plotX(j) = i                                                'j är x-värdesplatsen (första x-värdet skall
                                                                    'det stå den första bilden vi räknar 'exempelvis '4)
            plotY_pxl(j) = procent                                      'skriver ut hur många procent av bilden
                                                                'som täcks med röda fläckar.
            j = j + 1                                           'nästa element
pgb_progress.Value = pgb_progress.Value + 1     'så att progressbaren rör sig ett steg för varje bild som
            räknats
        Next i
        form_resultat.Show()                            'Visar graffönstret "form2"
        txt_show_pic.Visible = True
        lbl_show.Visible = True
        btn_visa.Visible = True
        '------Skriver ut "Procentuellt antal rödaktiga pixlar per bild"-graf-------
        form_resultat.Chart1.Series("Series1").ChartType = DataVisualization.Charting.SeriesChartType.Line
                'Chart1 är vårt grafobjekt. Här väljs typen
                                                                    'av graf, nu valdes Line, går att ändra, men
                                                                'skall användas då antal bilder understiger 'tusen
        form_resultat.Chart1.Series("Series1").Points.DataBindXY(plotX, plotY_pxl)

                                                                'DataBindXY skapar massa datapunkter för 'alla x-y-
                                                                    värden och binder ihop de med 'varandra så
                                                                    det senare kan skriva ut en 'linje mellan de
        form_resultat.Chart1.Legends.Item("Legend1").Enabled = False
```

```
                                                    'Dölj själva Series1 -rutan på sidan (samma ’som visas
                                                        i excell)
        form_resultat.Chart1.Series("Series1").BorderWidth = 3                  'Tjockleken på linjen.
        '------ sätter storlek för "Procentuellt antal rödaktiga pixlar per bild"-grafen-------
'Sätter det minsta värdet för y-axeln på grafen
        form_resultat.Chart1.ChartAreas.Item("ChartArea1").AxisX.Minimum = minpic
        form_resultat.Chart1.ChartAreas.Item("ChartArea1").AxisX.Maximum = maxpic
        form_resultat.Chart1.ChartAreas.Item("ChartArea1").AxisY.Minimum = 0
        '------ ställer in axlarna -------
        form_resultat.Chart1.ChartAreas.Item("ChartArea1").AxisY.LabelStyle.Format = "N0"
        form_resultat.Chart1.ChartAreas.Item("ChartArea1").AxisX.Title = "Bild nr"
        form_resultat.Chart1.ChartAreas.Item("ChartArea1").AxisY.Title = "Procent"
    End Sub
    Private Sub btn_visa_Click(sender As Object, e As EventArgs) Handles btn_visa.Click
        Try
            bm_show_pic = New Bitmap(txt_filepath.Text + "\" + "FFTL" + txt_show_pic.Text + ".jpg")
        Catch ex As Exception
            MsgBox("Kunde inte hitta filen")
            Return
        End Try
        form_show_pic.picbox_foto.Image = bm_show_pic           'Lägg in den angivna bilden i pictureboxen
          som finns i Form 3 (form_show_pic)
        form_show_pic.Show()                                    'Visar form3, "Fotografi"
        form_show_pic.picbox_foto.Show()                                    'Visar bilden
        form_show_pic.lbl_fotonr.Text = ("FFTL" + txt_show_pic.Text + ".jpg")     'Visar på bilden vilken bild
          som faktiskt visas
    End Sub
    Private Sub btn_browse_Click(sender As Object, e As EventArgs) Handles btn_browse.Click
                                                    'Fixar browse, filepath
        Dim select_path As Integer = FolderBrowserDialog1.ShowDialog()
        If select_path = DialogResult.OK Then
            txt_filepath.Text = FolderBrowserDialog1.SelectedPath
        End If
    End Sub
End Class
```

Visual Basic- kod. Huvudprogram: form_huvud.Designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class form_huvud
    Inherits System.Windows.Forms.Form
    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub
    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer
    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.calculate_btn = New System.Windows.Forms.Button()
        Me.txt_filepath = New System.Windows.Forms.TextBox()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.txt_bildnr1 = New System.Windows.Forms.TextBox()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.streck = New System.Windows.Forms.Label()
        Me.txt_bildnr2 = New System.Windows.Forms.TextBox()
        Me.pgb_progress = New System.Windows.Forms.ProgressBar()
        Me.diag_open = New System.Windows.Forms.OpenFileDialog()
        Me.lbl_show = New System.Windows.Forms.Label()
        Me.txt_show_pic = New System.Windows.Forms.TextBox()
        Me.btn_visa = New System.Windows.Forms.Button()
        Me.btn_browse = New System.Windows.Forms.Button()
        Me.FolderBrowserDialog1 = New System.Windows.Forms.FolderBrowserDialog()
        Me.SuspendLayout()
            'calculate_btn
        Me.calculate_btn.Location = New System.Drawing.Point(50, 154)
        Me.calculate_btn.Name = "calculate_btn"
        Me.calculate_btn.Size = New System.Drawing.Size(162, 44)
        Me.calculate_btn.TabIndex = 0
        Me.calculate_btn.Text = "Beräkna"
        Me.calculate_btn.UseVisualStyleBackColor = True
            'txt_filepath
        Me.txt_filepath.Location = New System.Drawing.Point(12, 29)
        Me.txt_filepath.Name = "txt_filepath"
        Me.txt_filepath.Size = New System.Drawing.Size(325, 20)
        Me.txt_filepath.TabIndex = 1
        Me.txt_filepath.Text = "C:\Desktop"
            'Label1
        Me.Label1.AutoSize = True
```

```
Me.Label1.Location = New System.Drawing.Point(9, 13)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(44, 13)
Me.Label1.TabIndex = 3
Me.Label1.Text = "Sökväg"
    'txt_bildnr1
 Me.txt_bildnr1.Location = New System.Drawing.Point(48, 129)
Me.txt_bildnr1.Name = "txt_bildnr1"
Me.txt_bildnr1.Size = New System.Drawing.Size(73, 20)
Me.txt_bildnr1.TabIndex = 7
    'Label2
Me.Label2.AutoSize = True
Me.Label2.Location = New System.Drawing.Point(45, 113)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(60, 13)
Me.Label2.TabIndex = 8
Me.Label2.Text = "Bildintervall"
    'streck
Me.streck.AutoSize = True
Me.streck.Location = New System.Drawing.Point(128, 130)
Me.streck.Name = "streck"
Me.streck.Size = New System.Drawing.Size(10, 13)
Me.streck.TabIndex = 12
Me.streck.Text = "-"
    'txt_bildnr2
Me.txt_bildnr2.Location = New System.Drawing.Point(143, 129)
Me.txt_bildnr2.Name = "txt_bildnr2"
Me.txt_bildnr2.Size = New System.Drawing.Size(69, 20)
Me.txt_bildnr2.TabIndex = 13
    'pgb_progress
Me.pgb_progress.Location = New System.Drawing.Point(8, 217)
Me.pgb_progress.Name = "pgb_progress"
Me.pgb_progress.Size = New System.Drawing.Size(256, 23)
Me.pgb_progress.TabIndex = 14
    'diag_open
Me.diag_open.FileName = "OpenFileDialog1"
    'lbl_show
Me.lbl_show.AutoSize = True
Me.lbl_show.Location = New System.Drawing.Point(276, 114)
Me.lbl_show.Name = "lbl_show"
Me.lbl_show.Size = New System.Drawing.Size(61, 13)
Me.lbl_show.TabIndex = 19
Me.lbl_show.Text = "Visa bild nr."
Me.lbl_show.Visible = False
    'txt_show_pic
Me.txt_show_pic.Location = New System.Drawing.Point(279, 130)
Me.txt_show_pic.Name = "txt_show_pic"
Me.txt_show_pic.Size = New System.Drawing.Size(60, 20)
Me.txt_show_pic.TabIndex = 18
Me.txt_show_pic.Visible = False
    'btn_visa
Me.btn_visa.Location = New System.Drawing.Point(280, 156)
Me.btn_visa.Name = "btn_visa"
Me.btn_visa.Size = New System.Drawing.Size(59, 22)
Me.btn_visa.TabIndex = 22
Me.btn_visa.Text = "Visa"
Me.btn_visa.UseVisualStyleBackColor = True
```

```
        Me.btn_visa.Visible = False
            'btn_browse
        Me.btn_browse.Location = New System.Drawing.Point(12, 55)
        Me.btn_browse.Name = "btn_browse"
        Me.btn_browse.Size = New System.Drawing.Size(69, 22)
        Me.btn_browse.TabIndex = 23
        Me.btn_browse.Text = "Lägg till"
        Me.btn_browse.UseVisualStyleBackColor = True
            'form_huvud
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(366, 275)
        Me.Controls.Add(Me.btn_browse)
        Me.Controls.Add(Me.btn_visa)
        Me.Controls.Add(Me.lbl_show)
        Me.Controls.Add(Me.txt_show_pic)
        Me.Controls.Add(Me.pgb_progress)
        Me.Controls.Add(Me.txt_bildnr2)
        Me.Controls.Add(Me.streck)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.txt_bildnr1)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.txt_filepath)
        Me.Controls.Add(Me.calculate_btn)
        Me.Name = "form_huvud"
        Me.Text = "Huvudprogram"
        Me.ResumeLayout(False)
        Me.PerformLayout()
    End Sub
    Friend WithEvents calculate_btn As System.Windows.Forms.Button
    Friend WithEvents txt_filepath As System.Windows.Forms.TextBox
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents txt_bildnr1 As System.Windows.Forms.TextBox
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents streck As System.Windows.Forms.Label
    Friend WithEvents txt_bildnr2 As System.Windows.Forms.TextBox
    Friend WithEvents pgb_progress As System.Windows.Forms.ProgressBar
    Friend WithEvents diag_open As System.Windows.Forms.OpenFileDialog
    Friend WithEvents lbl_show As System.Windows.Forms.Label
    Friend WithEvents txt_show_pic As System.Windows.Forms.TextBox
    Friend WithEvents btn_visa As System.Windows.Forms.Button
    Friend WithEvents btn_browse As System.Windows.Forms.Button
    Friend WithEvents FolderBrowserDialog1 As System.Windows.Forms.FolderBrowserDialog
End Class
```

Visual Basic- kod. Graf: form_resultat.vb

```
Imports System.Drawing.Imaging
Public Class form_resultat
  Private Sub SparaGrafToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
      menu_save_pic.Click
    Dim saveresult As Integer = diag_save_pic.ShowDialog()
    If saveresult = DialogResult.OK Then                    'kollar i princip om användaren
                                                          ’har tryckt okej i dialogrutan (som kommer ’upp efter
                                                              man klickat på "spara som bild")
      Chart1.SaveImage(diag_save_pic.FileName, System.Drawing.Imaging.ImageFormat.Jpeg)
    End If
  End Sub
  Private Sub menu_save_text_Click(sender As Object, e As EventArgs) Handles menu_save_text.Click
    diag_save_text.FileName = "Data.txt"
    Dim res As Integer = diag_save_text.ShowDialog()                'sparar ett värde beroende
                                                                ’på vilken knapp man ’trycker på i
                                                                    dialogrutan
    If res = DialogResult.OK Then
'------Här skapas en fil och skriver ut vilken sökväg bilderna hittades + 2 nya rader
      System.IO.File.WriteAllText(diag_save_text.FileName, form_huvud.txt_filepath.Text +
      System.Environment.NewLine + System.Environment.NewLine)
          For i = form_huvud.minpic To form_huvud.maxpic
'---- sparar i filen som skapades ovan, skriver i bildnamnet och antal röda pixlar och tar ny rad
      System.IO.File.AppendAllText(diag_save_text.FileName, "FFTL" + i.ToString + " " +
      form_huvud.plotY_pxl(i - form_huvud.minpic).ToString + System.Environment.NewLine)
      Next i
    End If
  End Sub
End Class
```

Visual Basic- kod. Graf: form_resulat.Designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class form_resultat
  Inherits System.Windows.Forms.Form
  'Form overrides dispose to clean up the component list.
  <System.Diagnostics.DebuggerNonUserCode()> _
  Protected Overrides Sub Dispose(ByVal disposing As Boolean)
    Try
      If disposing AndAlso components IsNot Nothing Then
        components.Dispose()
      End If
    Finally
      MyBase.Dispose(disposing)
    End Try
  End Sub
  'Required by the Windows Form Designer
  Private components As System.ComponentModel.IContainer
  'NOTE: The following procedure is required by the Windows Form Designer
  'It can be modified using the Windows Form Designer.
  'Do not modify it using the code editor.
  <System.Diagnostics.DebuggerStepThrough()> _
  Private Sub InitializeComponent()
    Dim ChartArea1 As System.Windows.Forms.DataVisualization.Charting.ChartArea = New
      System.Windows.Forms.DataVisualization.Charting.ChartArea()
    Dim Legend1 As System.Windows.Forms.DataVisualization.Charting.Legend = New
      System.Windows.Forms.DataVisualization.Charting.Legend()
```

```
Dim Series1 As System.Windows.Forms.DataVisualization.Charting.Series = New
  System.Windows.Forms.DataVisualization.Charting.Series()
Me.Chart1 = New System.Windows.Forms.DataVisualization.Charting.Chart()
Me.lbl_redpxl = New System.Windows.Forms.Label()
Me.MenuStrip1 = New System.Windows.Forms.MenuStrip()
Me.menu_save_pic = New System.Windows.Forms.ToolStripMenuItem()
Me.menu_save_text = New System.Windows.Forms.ToolStripMenuItem()
Me.diag_save_pic = New System.Windows.Forms.SaveFileDialog()
Me.diag_save_text = New System.Windows.Forms.SaveFileDialog()
CType(Me.Chart1, System.ComponentModel.ISupportInitialize).BeginInit()
Me.MenuStrip1.SuspendLayout()
Me.SuspendLayout()
    'Chart1
ChartArea1.Name = "ChartArea1"
Me.Chart1.ChartAreas.Add(ChartArea1)
Legend1.Name = "Legend1"
Me.Chart1.Legends.Add(Legend1)
Me.Chart1.Location = New System.Drawing.Point(12, 32)
Me.Chart1.Name = "Chart1"
Series1.ChartArea = "ChartArea1"
Series1.Legend = "Legend1"
Series1.Name = "Series1"
Me.Chart1.Series.Add(Series1)
Me.Chart1.Size = New System.Drawing.Size(1240, 700)
Me.Chart1.TabIndex = 10
Me.Chart1.Text = "Chart1"
    'lbl_redpxl
Me.lbl_redpxl.AutoSize = True
Me.lbl_redpxl.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
  System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
Me.lbl_redpxl.ForeColor = System.Drawing.SystemColors.InfoText
Me.lbl_redpxl.Location = New System.Drawing.Point(553, 9)
Me.lbl_redpxl.Name = "lbl_redpxl"
Me.lbl_redpxl.Size = New System.Drawing.Size(318, 20)
Me.lbl_redpxl.TabIndex = 12
Me.lbl_redpxl.Text = "Procentuella andelen rödaktigt per bild"
    'MenuStrip1
Me.MenuStrip1.Items.AddRange(New System.Windows.Forms.ToolStripItem() {Me.menu_save_pic,
  Me.menu_save_text})
Me.MenuStrip1.Location = New System.Drawing.Point(0, 0)
Me.MenuStrip1.Name = "MenuStrip1"
Me.MenuStrip1.Size = New System.Drawing.Size(1264, 24)
Me.MenuStrip1.TabIndex = 14
Me.MenuStrip1.Text = "MenuStrip1"
    'menu_save_pic
Me.menu_save_pic.Name = "menu_save_pic"
Me.menu_save_pic.Size = New System.Drawing.Size(97, 20)
Me.menu_save_pic.Text = "Spara som bild"
    'menu_save_text
Me.menu_save_text.Name = "menu_save_text"
Me.menu_save_text.Size = New System.Drawing.Size(96, 20)
Me.menu_save_text.Text = "Spara som text"
    'diag_save_pic
Me.diag_save_pic.DefaultExt = "jpg"
Me.diag_save_pic.FileName = "bild"
Me.diag_save_pic.Filter = "JPEG pictures | *.jpg"
    'diag_save_text
```

```
        Me.diag_save_text.DefaultExt = "txt"
        Me.diag_save_text.Filter = "text files | *.txt"
            'form_resultat
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(1264, 730)
        Me.Controls.Add(Me.lbl_redpxl)
        Me.Controls.Add(Me.Chart1)
        Me.Controls.Add(Me.MenuStrip1)
        Me.MainMenuStrip = Me.MenuStrip1
        Me.Name = "form_resultat"
        Me.Text = "Resultat"
        CType(Me.Chart1, System.ComponentModel.ISupportInitialize).EndInit()
        Me.MenuStrip1.ResumeLayout(False)
        Me.MenuStrip1.PerformLayout()
        Me.ResumeLayout(False)
        Me.PerformLayout()
    End Sub
    Friend WithEvents Chart1 As System.Windows.Forms.DataVisualization.Charting.Chart
    Friend WithEvents lbl_redpxl As System.Windows.Forms.Label
    Friend WithEvents MenuStrip1 As System.Windows.Forms.MenuStrip
    Friend WithEvents menu_save_pic As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents menu_save_text As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents diag_save_pic As System.Windows.Forms.SaveFileDialog
    Friend WithEvents diag_save_text As System.Windows.Forms.SaveFileDialog
End Class
```

Visual Basic- kod. Verifierande bild: form_show_pic.vb

```
Public Class form_show_pic
   Private Sub Form3_Load(sender As Object, e As EventArgs) Handles MyBase.Load
   End Sub

    Private Sub picbox_foto_Click(sender As System.Object, e As System.EventArgs) Handles
        picbox_foto.Click
   End Sub

   Private Sub lbl_fotonr_Click(sender As System.Object, e As System.EventArgs) Handles lbl_fotonr.Click
   End Sub
End Class
```

Visual Basic- kod. Verifierande bild: form_show_pic.Designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class form_show_pic
   Inherits System.Windows.Forms.Form
   'Form overrides dispose to clean up the component list.
   <System.Diagnostics.DebuggerNonUserCode()> _

       Protected Overrides Sub Dispose(ByVal disposing As Boolean)
      Try
         If disposing AndAlso components IsNot Nothing Then
            components.Dispose()
         End If
      Finally
         MyBase.Dispose(disposing)
      End Try
   End Sub
   'Required by the Windows Form Designer
   Private components As System.ComponentModel.IContainer
   'NOTE: The following procedure is required by the Windows Form Designer
   'It can be modified using the Windows Form Designer.
   'Do not modify it using the code editor.
   <System.Diagnostics.DebuggerStepThrough()> _
   Private Sub InitializeComponent()
      Me.picbox_foto = New System.Windows.Forms.PictureBox()
      Me.lbl_fotonr = New System.Windows.Forms.Label()
      CType(Me.picbox_foto, System.ComponentModel.ISupportInitialize).BeginInit()
      Me.SuspendLayout()
           'picbox_foto
      Me.picbox_foto.ImageLocation = ""
      Me.picbox_foto.Location = New System.Drawing.Point(12, 12)
      Me.picbox_foto.Name = "picbox_foto"
      Me.picbox_foto.Size = New System.Drawing.Size(1160, 688)
      Me.picbox_foto.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage
      Me.picbox_foto.TabIndex = 0
      Me.picbox_foto.TabStop = False
      Me.picbox_foto.Visible = False
           'lbl_fotonr
      Me.lbl_fotonr.AutoSize = True
      Me.lbl_fotonr.BackColor = System.Drawing.Color.Black
      Me.lbl_fotonr.Font = New System.Drawing.Font("Calibri", 18.0!, System.Drawing.FontStyle.Bold,
        System.Drawing.GraphicsUnit.Point, CType(0, Byte))
      Me.lbl_fotonr.ForeColor = System.Drawing.Color.White
      Me.lbl_fotonr.Location = New System.Drawing.Point(88, 645)
      Me.lbl_fotonr.Name = "lbl_fotonr"
```

```
        Me.lbl_fotonr.Size = New System.Drawing.Size(0, 29)
        Me.lbl_fotonr.TabIndex = 2
            'form_show_pic
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(1184, 712)
        Me.Controls.Add(Me.lbl_fotonr)
        Me.Controls.Add(Me.picbox_foto)
        Me.Name = "form_show_pic"
        Me.Text = "Fotografi"
        CType(Me.picbox_foto, System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)
        Me.PerformLayout()
    End Sub
    Friend WithEvents picbox_foto As System.Windows.Forms.PictureBox
    Friend WithEvents lbl_fotonr As System.Windows.Forms.Label
End Class
```