Avdelning för datavetenskap

Jan Eriksson

Streckkodsavläsare för mobiltelefoni

D uppsats 30 ECTS

2005:10

Karlstads universitet
Avdelning för datavetenskap

Streckkodsavläsare för mobiltelefoni

Jan Eriksson

This thesis is submitted in partial fulfillment of the requirements for the Masters degree in Computer Science. All material in this thesis which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Jan Eriksson

Approved, Date of defense

Opponent: Martin Persson

Advisor: Donald F. Ross

Examiner: Simone Fischer-Hübner

Sammanfattning

Detta magisterarbete handlar om streckkodsavläsning med hjälp av en mobiltelefonkamera. Rapporten visar på en metod för att jämföra olika avkodningsalgoritmer mot varandra. Metoden har i syfte att på ett enkelt och snabbt sätt jämföra prestationsförmågan mellan olika avkodningsalgoritmer i avseende på avkodningsprestanda. Arbetet berör därmed avkodningsalgoritmer för streckkoder i ett mobiltelefonkamerasystem, hur avkodningen kan utföras och de problem som ett mobiltelefonkamerasystem för med sig då det ska användas för streckkodsigenkänning. Problemens inverkan på avkodningssystemet gör streckkodsavläsningen svårare eller lättare att utföra då de påverkar kvaliteten på den avbildade streckkoden. I detta arbete utnyttjas dessa problemfaktorer. Störningsfaktorer utgör grunden för metoden som beskrivs i rapporten. Vidare testas metoden genom en implementation av tre open-source avkodningsalgoritmer. Två är för den 1D streckkoden EAN13 och en för den 2D streckkoden Visual Code.

Abstract

This master thesis deals with barcode reading with the help of a mobile phone equipped with a camera. The report shows a method for comparing different decoding algorithms against each other. The methods purpose is to compare and evaluate the performance capacity between different decoding algorithms on the same type of barcode in a simple way. The thesis discusses the decoding algorithm for barcodes in a system for mobile phones with a camera, how the decoding can be done and the problems that mobile phones with a camera brings to the table when used for barcode decoding. Depending on the affects of the problems on the decoding system the decoding becomes harder to do as the effects decreases the quality of the reproduced image of the barcode. In this thesis these affects is used to produce the method. The interference factors forms the basis for the method described in this report. The method is then tested by implementing three open-source decoding algorithms. Two for the 1D barcode EAN13 and one for the 2D barcode Visual Code.

Förord

D-uppsatsen som presenteras i denna rapport har utförts på Institutionen för Informationsteknologi avdelningen Datavetenskap på Karlstads Universitet under våren 2005. Jag vill tacka min handledare Donald F. Ross för en konstruktiv handledning. Jag vill också tacka Göran Urby vid Briscan AB för inspiration och hjälp samt för möjligheten att göra detta projekt, som för mig varit väldigt intressant och lärorikt.

Jan Eriksson, Karlstad nov 2005

Innehåll

Figurer

Tabeller

Diagram

Förkortningar

CCD - Charge Coupled Device.
CIF - Crystallographic Information File
CMOS - Complementary Metal Oxide Semiconductor.
EAN - European Article Numbering
GPRS - General Packet Radio Service
IrDA - Infraröd dataanslutning
J2EE - Java 2 Enterprise Edition
J2ME - Java 2 Micro Edition
JPEG - Joint Photographic Experts Group
LCD - Liquid Crystal Display
LED - Light Emitting Diode
MIDP - Mobile Information Device Profile.
MMC - MultiMediaCard.
MMS - Multimedia Messaging Service
SDK - Software Developing Kit
SMS - Short Message Service
TFT - Thin Film Transistor
UCC - Uniform Code Council
UPC - Universal Product Code
USB - Universal Serial Bus
VGA - Video Graphics Array
WAP - Wireless Application Protocol

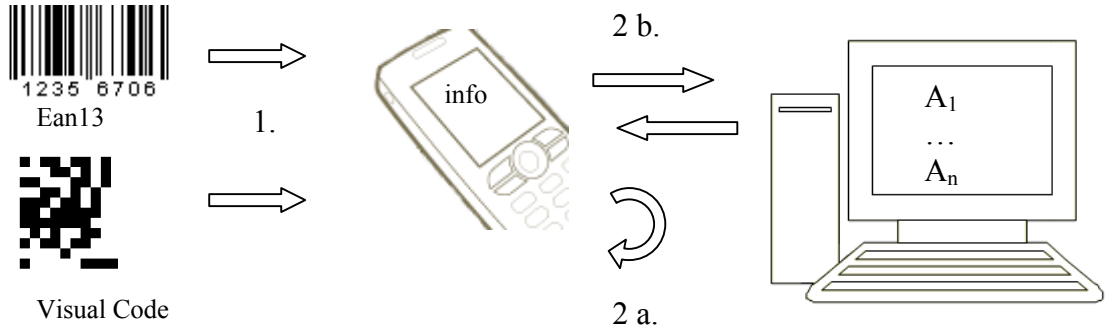
1 Introduktion

En stor del av befolkningen i t.ex. Sverige har sett en 1D streckkod idag. De sitter tryckta på ett stort antal produkter i detaljhandeln och till och med på baksidan av identifierings-handlingar så som körkort m.fl. Vad som inte är allmän vetskap är att det finns en mängd olika typer av streckkoder, endimensionella och tvådimensionella. Finessen med att ha en streckkod på en produkt är att en maskin eller avläsare kan identifiera produkten med en hög tillförlitlighet för att sedan utnyttja denna identitet i något syfte, prissättning, inventering med mera.

1.1 Bakgrund

Att skanna och läsa ut informationen från olika streckkoder har inte varit något privatpersoner gjort utan det sker inom olika professionella tillämpningar i industrin och handeln. Det görs då med specialanpassade läsare. Dessa läsare utnyttjar någon sorts laser- eller belysningsteknik i huvudsak. Det är en enkel teknik att utnyttja i och med att streckkoderna är endimensionella. Endimensionella streckkoder har dock vissa begränsningar, de blir bl.a. långa när de ska representera stora datamängder. Vill man lagra större datamängder än 15-50 tecken är tvådimensionella streckkoder fördelaktiga då de lagrar data både på längden och på höjden vilket medför att de blir mindre jämfört med en endimensionell streckkod vid lagring av samma datamängd [27]. Vill man använda tvådimensionell streckkod blir dock läsarna tekniskt mer avancerade (se kapitel 3) och mer vanligt börjar det bli att de i princip består av en digitalkamera. I gengäld krävs då en avkodningsmjukvara som måste vara ganska avancerad då den jobbar med mönsterigenkänningstekniker.

En sådan mönstergenkänningsalgoritm kan implementeras i antingen en mobiltelefon (figur 1.1, 2a) eller i en server som tar emot en bild från en mobiltelefon via MMS (figur 1.1, 2b). Och genom att utnyttja en kameratelefon skulle mobilen med hjälp av kameran kunna utnyttjas som en streckkodsavläsare. Det medför stora möjligheter. Det skulle bl.a. betyda att det finns en potentiell streckkodsavläsare hos snart varje enskild privatperson då nära 66 % av de 3 300 000 mobiltelefoner som såldes på den svenska marknaden 2004 hade en inbyggd digitalkamera [40]. Marknaden för en sådan tillämpning har stora möjligheter.



1. En 1D eller 2D streckkod fotograferas av en mobiltelefon.

2 a. Via mobilen skickas bilden av streckkoden till en server som genomför avkodning av streckkoden. Svaret messas tillbaka till mobilen.

2 b. Mobiltelefonen avkodar streckkodsbilden med hjälp av en intern applikation som också presenterar resultatet.

Figur 1.1 - Introduktion

1.2 Disposition

Rapportens inledande kapitel behandlar bakgrund och tar upp olika typer av streckkoder som existerar idag, samt vilka som med fördel kan utnyttjas vid fotografering med mobiltelefon. Kapitel 3 tar även upp olika problem som uppstår då en mobilkamera används som läsare av streckkoder. I kapitel 4 beskrivs metodikutveckling, teori för avkodare till 1D och 2D streckkoder samt olika lösningsförslag till en streckkodstillämpning för en mobiltelefon med kamera. Resultat presenteras i kapitel 5 och följs av slutsatser och diskussion kring resultaten.

1.3 Problemställning

När en streckkod avbildas med hjälp av en digitalkamera som i ett mobiltelefonkamerasystem finns risker att avbilden på olika sätt kan ha blivit förstörd eller förändrar gentemot originalet. Dessa faktorers påverkan på den kodade 1D eller 2D streckkoden är avgörande för avkodningsalgoritmens möjlighet att genomföra en lyckad avkodning. Rapporten tar upp och undersöker dessa störningsfaktorer i syfte att använda dem i en metod för att på ett snabbt sätt jämföra olika streckkodsavkodare mot varandra. Typen av streckkod kan dock beroende på sin uppbyggnad vara olika bra på att parera dessa störningar och detta arbete har därför också

tittat på vilka olika typer av streckkoder som finns ute på marknaden idag för att undersöka om någon utav dessa existerande streckkodsformat är lämplig att utnyttja i ett mobilkamerasystem.

1.4 Sammanfattning

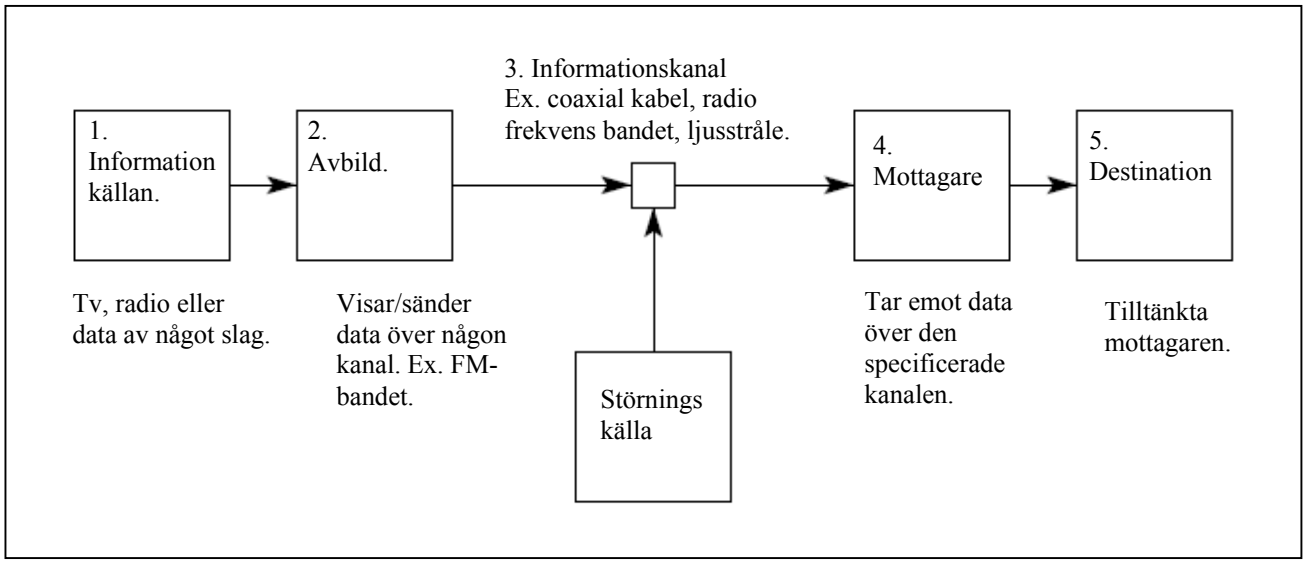
Rapporten behandlar mönstergenkänningsalgoritmer som är ryggraden för ett igenkänningssystem likt de som beskrivs i introduktionen ovan. Rapporten tar upp hur avkodningen kan ske med hjälp av några olika angripningssätt samt problem som ett mobilkamerasystem för med sig då det ska användas för streckkodsigenkänning. Baserat på dessa problem tas en metod fram för att på ett enkelt sätt kunna jämföra prestationsförmågan mellan olika avkodningsalgoritmer på samma streckkod.

2 Bakgrund

2.1 Syfte och frågeställning

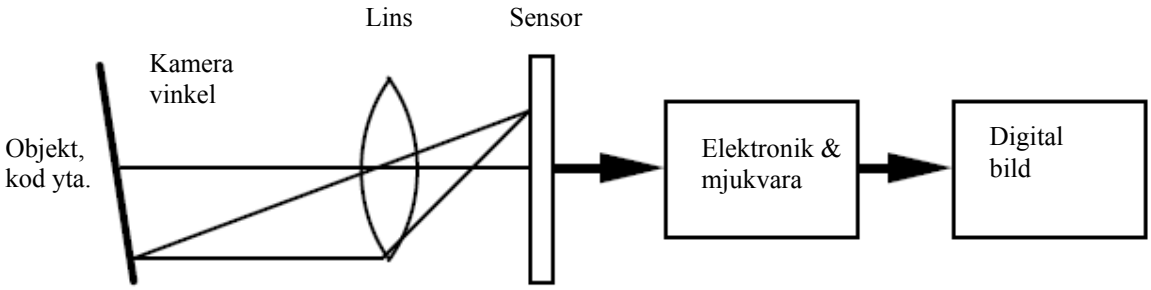
Syftet med denna D-uppsats är att undersöka vilka 1D och 2D streckkodsystem som lämpar sig bäst för avkodning med hjälp av en mobilkamera och att ta fram en metod för att jämföra olika avkodnings algoritmer. För detta behövs en omvärldsanalys av befintliga 1D och 2D streckkodssystem och en undersökning av störningsfaktorer som kan förvränga informationen i ett 1D och 2D streckkodsystem. Ett streckkodsystem kan ses som vilket annat informationssystem som helst, se Shannons diagram (figur 2.1) över ett generellt informationssystem [18]. Figuren visar övergången från:

1. Informationskällan, den information som kodas av 1D, 2D streckkoden.
2. Avbild, själva 1D, 2D streckkoden.
3. Informationskanalen, visuella planet.
4. Mottagare, mobilkamera med avkodnings mjukvara.
5. Destination, innehåll i 1D, 2D streckkod ges till mottagaren.



Figur 2.1 - Shannons informationsdiagram

I övergången från avsändare till mottagare kan informationskvaliteten, likt andra informationssystem, förstöras eller förändras utav olika störningskällor. I ett system för 1D och 2D streckkoder finns ett antal faktorer som påverkar avkodningsalgoritmens möjlighet att avkoda en 1D eller 2D streckkod i en bild tagen av en mobilkamera, en robust algoritm kan parera störningsfaktorerna bättre än en enkel algoritm. Störningseffekter kan grupperas ihop inom 3 olika områden.



Figur 2.2 - Digitalkamera system [1]

1. Kamerafaktorer – Faktorer som påverkar bilden som tas med en kamera, t.ex. linseffekter (ex ”barrel” effekter), upplösning (sensor), komprimering av bild.
2. Miljöfaktorer – Yttre faktorer som påverkas av personen som tar bilden, t.ex. avstånd mellan kod (objektet) och kamera, relativa vinkelskillnader mellan kod och kamera i både X och Y led, så kallade perspektivfel.
3. Tryck och kod – Faktorer som påverkar hur ettor och nollor är representerade i tryckt tillstånd, t.ex. egenskaper i 1D eller 2D streckkodstandarden som bl.a. definierar avstånd mellan svarta och vita streck. Ytan där koden är tryckt på kan även den vara en bidragande faktor, ex runda eller reflekterande ytor.

Dessa faktorers påverkan på den kodade 1D eller 2D streckkoden är avgörande för avkodningsalgoritmens möjlighet att genomföra en lyckad avkodning och är grunden för algoritmjämförelsemetoden. Vidare för att testa rapportens metod skall minst en, helst två avkodningsapplikationer för någon specifik 1D eller 2D streckkod skapas med hjälp av

”open-source” algoritmer. Indata till avkodarna är en bild av en streckkod och resultat ut ska vara det kodade innehållet i streckkoden. Totalt utvecklades 3 olika avkodare med hjälp av open-source algoritmer. 2 avkodare för EAN13 och 1 avkodare för Visual Code. Experimentet består av 2 olika steg, steg 1 är huvudmomentet och steg 2 sker i mån av tid.

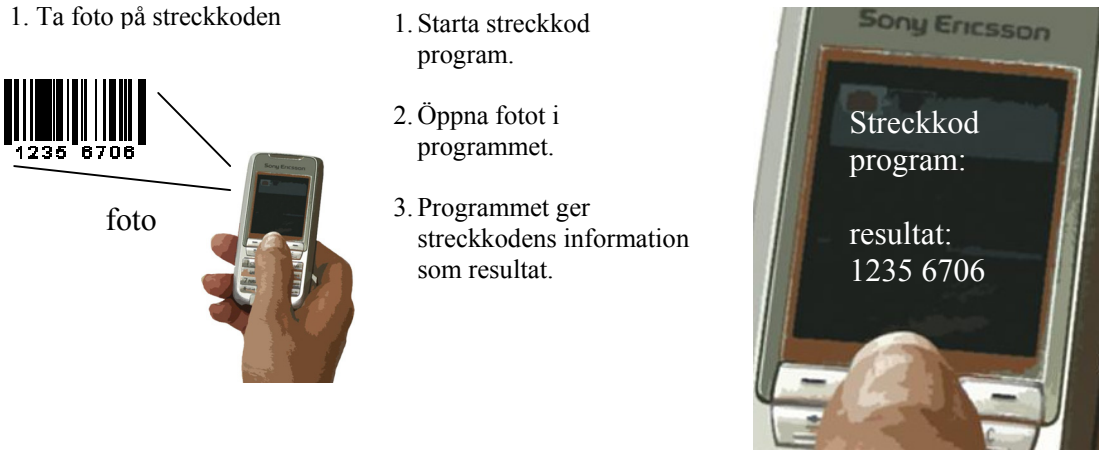
2.1.1 Steg 1

Steg 1, blir att utveckla en avkodnings applikation som körs på en server. Servern mottar streckkoden som MMS från en mobiltelefon, avkodar bilden och skickar resultatet tillbaka till mobiltelefonen via MMS eller SMS. Resultatet i det här fallet är informationen streckkoden representerar.

Figur 2.3 - Steg 1, serverapplikation

2.1.2 Steg 2

I steg 2 skall avkodaren läggas in i en applikation för en kameratelefon, antingen i Java MIDP (Mobile Information Device Profile) eller i C och Symbian. Användaren tar ett kort på en streckkod, öppnar applikationen och väljer bilden som tagits, applikationen kodar av streckkoden i bilden och ger streckkodens information tillbaka till användaren.



Figur 2.4 - Steg 2, mobilapplikation

2.2 Begränsningar

Projektet kommer att begränsas till att först och främst fullfölja steg 1, efter att steg 1 är utfört uppskattas att omkring 80 % av allt arbete redan utförts och att endast omkring 20 % återstår för att genomföra steg 2. Det bör också nämnas att det inte alls är meningen att i detta examensarbete utveckla en avkodningsalgoritm från noll utan att om möjligt använda öppen källkod eller liknande som ska integreras till en fungerande tillämpning. Området med streckkodsavkodning med hjälp av mjukvara och mobiltelefonkameror spänner över en mängd olika områden så som optik, datavetenskap, elektronik, fysik m.m. Det medför en mängd olika faktorer som kan spegla in på de resultat som denna rapport kommer att presentera.

2.3 Terminologi

Nedan ges förklaring till begrepp som förekommer i rapporten i samband med 1D och 2D streckkoder som kan vara oklara i sin betydelse eller nya för en läsare obekant med termer och begrepp från streckkodsstandarden. Förklaringarna är gjorda i en förhoppning att göra bakgrunden mer lättförstålig

Teckenuppsättning Med teckenuppsättning menas det urval av tecken som kan representeras av standarden. Det finns vanligtvis 3 olika uppsättningar: numerisk, alfanumerisk och komplett ASCII.

1D Endimensionella.

2D Tvådimensionella.

Streckkoder 1D och 2D streckkoder.

1D streckkoder Alla 1D streckkoder.

2D streckkoder Alla 2D streckkoder, staplade, multi-rader och matriskod.

Matriskod Matrisstreckkod.

Luminens Ljusstryka

Självkontrollerande Med en självkontrollerande kod menas att ett kodat tecken inte kan misstolkas som ett annat tecken beroende på t.ex. utskrifts defekter. En icke självkontrollerande standard använder sig oftast av ett kontroll tecken för att utesluta fel.

Element Ett streck, eller ett mellanrum i en streckkod.

X-värde Kallas även modulbredd. En benämning på det smalaste elementet i streckkoden. X-värdet bestämmer hur bred en streckkod blir med ett visst datainnehåll. Ett litet X-värde gör att det går att lagra fler tecken på en mindre yta. För 2D matriskod refererar X till dimensionen på den minsta datacellen. X mäts ofta i mils som är en tusendels tum.

Höjd En hög streckkod är lättare att "träffa" med en laseravläsare, en hög streckkod ger också en hög redundans, vilket resulterar i en hög tolerans för skador på streckkoden och en hög korrekthet vid avläsning.

Bred-till-smal (N) Förhållandet mellan breda och smala element i en streckkod, förhållandet påverkar standardens densitet.

Start/Stopp tecken Start och stopp tecken indikerar var standarden börjar och var den slutar, de kan också indikera från vilket håll streckkoden skall avläsas.

Statisk längd En standard med statisk längd kan inte lagra fler eller färre tecken än sin förutbestämda längd.

Varierande längd En standard med varierande längd kan representera hur många tecken som helst. Det finns ingen restriktion för mängden data den kan representera. En sådan streckkod använder sig av start och stopp symboler.

Kontinuerlig Streckkod En streckkod kallas kontinuerlig om alla streck och mellanrum i streckkoden tillhör olika tecken, streckkoden använder sig inte av tomrum för att skilja tecken åt. En kontinuerlig standard använder sig också vanligtvis av start och stopp symboler.

Diskret Streckkod I en diskret standard separeras varje tecken med ett tomrum. Dessa tomrum indikerar ingen information utan används enbart för att skilja olika tecken åt.

Tyst zon Ett tomt område, som inte innehåller några tecken eller märken. Området omringar ofta hela streckkoden. De flesta streckkoder kräver en tyst zon innan start- och efter stoppsymbolen.

2.4 Sammanfattning

En omvärldsanalys av befintliga 1D och 2D streckkodsstandarder och en undersökning av störningsfaktorer i streckkodsavkodningssystem för mobiltelefonkameror kommer att utgöra grunden för att ta fram en metod för algoritmjämförelse. Faktorer som kamera, miljö (yttre faktorer) och tryck/kod har alla del och kan alla påverka en avkodningsalgoritms möjlighet att utföra en korrekt avkodning. Dessa faktorer kommer att studeras noggrannare i nästa kapitel för att med dess hjälp skapa en algoritmjämförelsemetod.

3 Omvärldsanalys av 1D och 2D streckkoder

3.1 Inledning

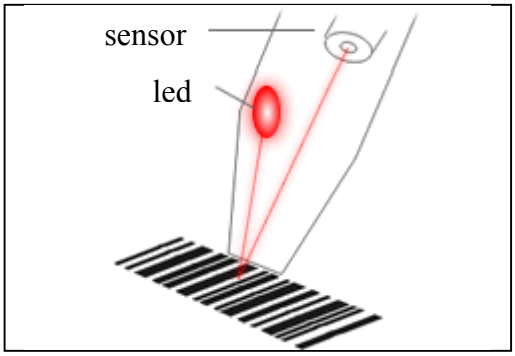
Ett streckkodssystem är en automatisk identifierings teknologi som gör insamling av data snabb och exakt. Streckkoden är idag allmänt utbredd och helt tagen för given. Den har gått från att ha varit en enkel prisgivningsdetalj, till att revolutionera systemen för lagring och spårbarhet. Den har gått från att ha varit en nymodernitet för 25 år sedan, till att idag nästan blivit en symbol för övervakning och associeras ibland ihop med 1984, Orwells ”Big Brother” dystopi [13], ett sorts inhumant totalitärt framtida samhälle. Streckkoden är idag mycket vanlig och stor inom både industrin och detaljhandeln, streckkoder finns överallt.

Vad som inte är helt självklart dock, är att det finns en mängd olika typer av streckkoder. Den vanligaste är endimensionell, den lagrar bara information över bredden och höjden används endast för att göra streckkoderna enklare att läsa av (figur 3.1). Men genom att lagra information även på höjden har streckkoderna blivit tvådimensionella och möjliggjort lagring av upp till 2000 gånger mer information [27].

Figur 3.1 - Avläsning av 1D streckkod

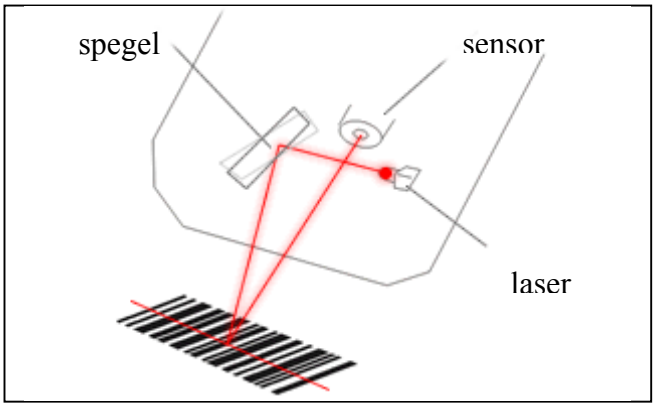
Det finns runt 300 olika streckkodsystem [30], likadant finns det flera olika sätt på vilket en streckkod kan avläsas, men det finns huvudsakligen 3 olika system. När streckkoder först kom fram för 25 år sedan användes vanligtvis en enkel läspenna som kände av streckkoden när den drogs över linjerna. Det var knepigt att utföra men ändå 5 gånger så snabbt och dessutom 10 000 gånger mer exakt än att knappa in priserna manuellt [38]. De första läspennorna använde enkla typer av glödlampor som inte fungerade särskilt länge.

Streckkodmarknaden expanderade därför rejält då dessa bräckliga ljuskällor kunde ersättas av hållbara lysdioder vilket gjorde systemen mer tillförlitliga och mer användarvänliga [27].



Figur 3.2 - Läspenna

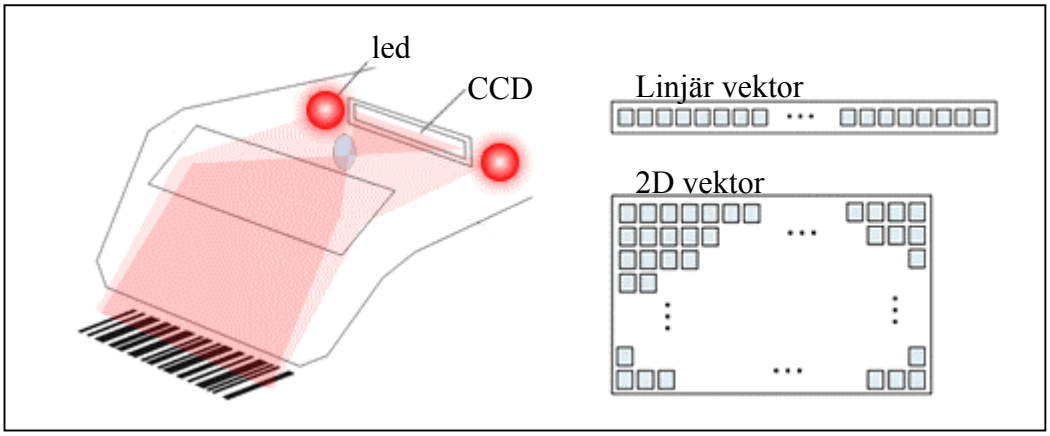
Nästa typ är laseravläsare. Laseravläsare fungerar på samma sätt som en läspenna förutom att den använder en laserstråle som ljuskälla och en spegel eller prisma för att föra ljusstrålen fram och tillbaka över streckkoden. Hos både läspenna och laseravläsare används en fotodiod för att avläsa intensiteten i ljuset som reflekteras tillbaka.



Figur 3.3 - Laseravläsare

Nästa steg i utvecklingen är CCD-läsaren (Charge Coupled Device) eller linjära bildframställare, som med hjälp av ett stort antal fotosensorer kan läsa av ljusa och mörka fält i en streckkod. För att läsa 2D streckkoder behöver en laserläsare eller en linjär bildframställare svepa över koden flera gånger. Det är en positionskänslig och långsam metod.

Genom att byta ut den linjära vektorn i en CCD-läsare mot en tvådimensionell, så är förvandlingen till en digitalkamera inte långt borta och systemen använder sig av processorer och mjukvara för att känna igen mönstren i olika streckkoder. Nästa steg som börjar bli intressant idag är mobiltelefoner med inbyggd kamera.



Figur 3.4 - CCD läsare, linjär och 2D vektor

3.2 Olika typer av streckkodsystem

Varje system har sina egna regler för hur tecken kodas, hur linjer avkodas, för felkontroll m.m. Streckkodsystemen skiljer sig åt både genom vilken typ av data som går att koda samt på vilket sätt de representerar data. I vissa system används bara numeriska tecken, i andra används alfabetiska och numeriska tecken. Vissa hanterar hela ASCII-tabellen och en del kan rekonstruera data om någon symbol blivit skadad [28].

Streckkoder består av en serie svarta linjer skilda av vita mellanrum. Bredden av den smalaste linjen eller mellanrummet benämns som modulbredd X och betecknar längden mellan alla svarta och vita streck, och är i princip ett mått på streckkodens längd. Desto större X är, desto enklare är det att läsa av streckkoden, men ett stort värde på X ger också att streckkoden tar större plats. Den minimala X dimensionen för ett ”öppet system” av en specifik streckkod system är 0,0075 tum, 7.5 milli tum eller 0.1905 mm. Med ett ”öppet system” menas att streckkoden då kommer att vara läsbar av alla streckkodsläsare för den specifika standarden

på marknaden. Modulbredden bestäms av symbolstandarden och är olika för olika standarder. En annan viktig aspekt förutom X dimensionen är förhållandet mellan breda och tunna element. Det är en fördel att ha ett stort förhållande om streckkoden skall vara enkel att avläsa, ett stort förhållande ger dock även en mer utrymmeskrävande symbol. För rätt avläsning använder sig de flesta symbolstandarder av s.k. tysta zoner, dvs. ett tomrum mellan streckkoden och eventuell annan text eller bild. Zonen bör vara minst 10 gånger så stor som modulbredden. Modulbredden mäts ofta i enheten mil som är tusendelar av en tum. Några vanliga mått är 7,5 och 15 mil som motsvarar 0,0075 och 0,015 tum eller 0.1905 mm och 0.381 mm

Alla streckkoder är uppbyggda ungefär på samma sätt, starttecken + information + stopptecken. Start och stopptecken används av avläsaren för att identifiera streckkodens symbolstandard. I informationen representeras varje tecken som skall kodas av ett antal streck och mellanrum, antalet vita och svarta streck som representerar ett tecken definieras i symbolstandarden. Slutligen före stoppteckenet, används ibland ett kontrolltecken som avläsaren kan använda för att verifiera informationens integritet. I vissa symbolstandarder visas också streckkoden som klartext under streckkoden. Klartext används framförallt inom detaljhandeln om streckkoden inte går att läsa med streckkodsläsaren och man måste knappa in tecknen via tangentbord.

Det finns över 300 [30] kända streckkodsystem, men bara ett tiotal av dessa är vanligt förekommande eller används aktivt idag. Det finns i huvudsak två olika kategorier i hur data är representerat i ett streckkodsystem, endimensionella och tvådimensionella.

3.2.1 1D

Streckkoder i sin mest bekanta form 1D, en serie av vertikalt parallella linjer, är fortfarande världens mest spridda teknologi för optisktläsning. Dessa mer vanliga streckkoder är vertikalt redundanta, vilket betyder att samma information lagras kontinuerligt över lodlinjen och som även betyder att de är endimensionella. Redundansen ger att streckkoder med tryckfel som fläckar eller tomrum ändå kan läsas utan att förlora någon information. Högre höjd ger större

sannolikhet att åtminstone en rad inom streckkoden kan avkodas. Höjden på en 1D streckkod kan därför förminskas utan att någon information går förlorad.

Några exempel på vanligt förekommande endimensionella streckkoder och varför de tagits fram följer nedan och avslutas med en sammanfattning.

Codabar

Figur 3.5 - Codabar

Codabar är en tidig streckkod som utvecklades 1972 i syfte att användas inom prismärkning [31]. Livsmedelsindustrin accepterade dock inte Codabar utan valde att arbeta med UPC (Universal Product Code). Streckkodstypen fick dock en spridning inom andra områden, den har används först och främst inom fotoindustrin samt läkemedelsindustrin (blodbanker) men även vid bibliotek m.fl. Codabar var en av de första streckkoderna som introducerades i Japan, den etablerades där 1994 och går under namnet NW-7. Codabar har en variabel längd, den är självkontrollerande men har ingen kontrollsumma, vissa affärsverksamheter bl.a. biblioteken har dock infört en egen kontrollstandard. Codabar är diskret och använder sig av två olika bredder på strecken, smala och breda. Förhållandet mellan smal-bred skall ligga mellan 2,0 och 3,0 ggr och vara konstant i symbolen. Förhållandet måste vara över 2,2 om X dimensionen är under 0,02 tum och symbolen ska ha en höjd på minst 0,25 tum eller 15 % av längden.

Codabar kan koda totalt 16 olika tecken, siffrorna 0-9 och sex special symboler (- $: / . +). Dessutom finns det fyra koder som används som start och stopptecken, A, B, C och D. Olika kombinationer av start- och stoppteckнen kan även de användas för att indikera information. Symbolen består av en tyst zon som skall vara 10 gånger bredden av det minsta strecket, streckkoden börjar med ett starttecken, följt av en zon med kodad data och slutligen ett stopptecken. Varje tecken kodas med hjälp av sju binära bitar av streck. Breda mellanrum står

för logiska ettor och smala representerar nollor. För att koda siffrorna (0-9) och 2 special tecken ($, -) används 2 breda mellanrum, övriga 5 element är smala. Ett av de två breda mellanrummen är svart och det andra är vitt. Övriga special tecken har 3 breda element, alla svarta streck, övriga element är smala. Start och stoppsymbolerna representeras även de av 3 mellanrum och 4 svarta streck, där ett av mellanrummen är svart och två är vita. Tabellen i Bilaga A visar hur varje tecken kodas.

Den här typen av struktur utvecklades för att vara så enkel som möjlig att läsa av trots de många tryckfel som var vanligt förekommande på den tidens skrivare. Strukturen gör också att alla tecken är lika långa oavsett om två eller tre breda element används. Codabar kan tryckas med en modul upplösning på 0,0065 tum, en utav de högsta upplösningarna av alla streckkoder. Datadensiteten är då 11 tecken/tum. Codabar rekommenderas dock att tryckas med ett stegs förstorning till 0,0081 tum, detta för att göra koden enklare att avläsa. Varje förstorningssteg är på 25 %.

Interleaved 2/5

Figur 3.6 - Interleaved 2/5

Interleaved 2/5 är en hög densitets, självkontrollerande, kontinuerlig streckkod med en numerisk teckenuppsättning och variabel längd och utvecklades 1972 av Intermec [36].
Namnet “Interleaved” (ungefär sammanfläta) kommer från att 2 siffror alltid kodas tillsammans som ett tecken. De svarta strecken kodar siffran med udda sifferposition och de vita tomrummen kodar siffran med jämn position. 2/5 i namnet kommer från att varje tecken består av 5 element antingen 5 streck eller 5 tomrum, 2 av dessa är alltid breda och 3 är smala. Varje siffra har sin unika 2 av 5 kombination vilket gör koden enkel att kontrollera. Tabellen i Bilaga B visar hur varje tecken kodas binärt.

Den höga datadensiteten i Interleaved 2/5 kommer från att den inte använder några skiljezoner mellan tecknen samt att den använder både streck och tomrum för att koda sin information. Streckkodens kontinuerliga och sammanflätande struktur gör den dock komplex att trycka och läsa. Sammanflätningen gör också att koden måste ha ett jämt antal siffror. Standardens struktur är dock en av de enklaste, endast två olika bredder används. Breda element symboliserar ettor, tunna element representerar nollor. Det breda elementet är Y gånger så stor som det smala elementet där Y kan variera från 2,0 till 3,0. 3,0 är rekommenderat och multiplikationen måste vara konstant genom hela symbolen. Om X dimensionen är mindre än 0,02 tum så måste multiplikationen minst vara 2,2. Den tysta zonen skall vara 10 gånger X dimensionen och höjden på symbolen skall vara 0,25 tum dock minst 15 % av längden. Interleaved 2/5 har en teckenuppsättning på endast numeriska siffror, 0-9, samt stopp och start mönster. Symbolen består av en startkod, datatecken, kontrolltecken som är valbar, slutligen följt av en stoppkod. Startkoden börjar med ett svart streck, följt av 3 smala element och stoppkoden börjar med ett brett svart streck följt av 2 smala element.

Av säkerhetsskäl frångås koden mer och mer. Kodens start- och stopptecken är nämligen kortare än dess dataelement. Eftersom start- och stopptecknen inte är unika utan kan förekomma i kodens övriga element kan det medföra att skannern mitt i en kod tror att hela symbolen har lästs vilket resulterar i en kortare men fortfarande godkänd streckkod. Problem kan lösas genom att låta avkodarna läsa in ett fastställt antal siffror. Denna symbolstandard har den högsta densiteten för numerisk data, den är dock en äldre typ som ofta ersätts med den mer moderna Code 128 (se nedan).

Code 39

Figur 3.7 - Code 39

Code 39 är den första streckkoden som utvecklades som kunde representera både siffror och bokstäver och utvecklades 1975 av Dr. David Allais och Ray Stevens. Code 39 är idag en av

de mest använda streckkoderna, det är den populäraste streckkoden utanför varuhandeln. Den är mycket vanlig i industrin och används både inom produktion, sjukvård och inom militären. Code 39 används bl.a. av US Department of Defence under namnet LOGMARS.
Streckkoden är diskret, den kan ha variabel längd, är självkontrollerande och är en alfanumerisk kod. Den stora fördelen med Code 39 jämfört med tidigare streckkoder är möjligheten att använda bokstäver, också att den är enkel att läsa. Nackdelarna däremot är att standarden har en låg datadensitet vilket gör den utrymmeskrävande och det ger ofta rent praktiskt snabbt en gräns för hur många tecken som kan representeras. Desto mindre yta desto färre tecken kan representeras.

Teckenuppsättningen består av 10 siffror (0-9), 26 bokstäver (A-Z) och 7 special tecken (-.$%/+SPACE). Code 39 finns också i en version som kallas "Utökad code 39" [49] och den kan då representera 127 tecken ur ASCII tabellen. Varje tecken består av 9 element, 5 är svarta streck och 4 är vita mellanrum. 3 av elementen är dessutom breda och resterande smala. Tecknen är skilda åt av ett mellanrum som är av samma storlek som ett smalt element. En tabell över teckenkodningen finns i Bilaga C. Förhållandet mellan breda och tunna element är från 2,0 till 3,0 gånger, där 3 är optimalt. Om X dimensionen är mindre än 0,02 tum måste förhållandet vara av storleksordningen 1:2,2 eller större. Höjden på symbolen skall vara minst 15 % av symbolens längd eller 0,25 tum. Varje streckkod måste också ha en tyst zon före och efter sig, zonen skall vara på 10 gånger X dimensionen. Minsta X dimensionen som är rekommenderad för Code 39 är 0,0075 tum. Streckkoden är uppbyggd av först ett start tecken ’*’, följt av den kodade informationen samt ett stopp tecken ’*’.

Eftersom koden är självkontrollerande och att den är enkel att läsa gör att Code 39 inte behöver använda ett kontrolltecken, även om möjligheten finns att lägga in en checksiffra när datasäkerhet är extra viktig, sjukvården använder bl.a. denna metod. Code 39 är nästan den enda vanligt förekommande streckkoden som inte behöver någon checksiffra vilket gör koden mycket bra för tillämpningar där det är mycket svårt eller omöjligt att göra kontroll beräkningar. En annan speciell egenskap för Code 39 är sammanlänkning av två eller flera streckkoder. Det kan ibland vara fördelaktigt att dela långa koder till fler men kortare. Detta går till på så sätt att om första tecknet i streckkoden är ett mellanslag bifogas efterföljande streckkod som läses till i slutet på den föregående.

EAN 13

Figur 3.8 - EAN 13

Streckkoder av typerna EAN [33] (European Article Numbering) och UPC [50] är de överlägset mest vanliga streckkoderna. Kodtypen utvecklades ursprungligen för att automatidentifiera dagligvaror på amerikanska stormarknader och antogs formellt som standard år 1973 av den amerikanska handelorganisationen och fick då namnet UPC-A. EAN-13 är i princip en påbyggnad av den amerikanska UPC-standarden, de två streckkodssystemen är helt lika med undantag för längden data de kan lagra. EAN-13 har en längd på 13 tecken, medan UPC-A klarar 12 tecken. UPC finns i huvudsak i två varianter, UPC-A och den korta varianten UPC-E, som klarar att koda 7 siffror och motsvaras av EAN-8 som kan lagra 8 siffror. UPC kan konverteras till EAN genom att en nolla läggs till framför den första siffran. EAN introducerades i Europa 1977 och har sedan dess blivit den internationella standarden, med undantag för USA och Kanada. EAN-13 är den EAN/UPC-kod som är mest utbredd i hela världen, koden finns på ungefär 90 % av alla förbrukningsvaror som säljs inom detaljhandeln. Omkring 300 000 företag använder EAN standarden globalt, plus ytterligare 138 000 under UCC (Uniform Code Council), som arbetar tillsammans med EAN i Nord-Amerika.

EAN-13 är en kontinuerlig, numerisk streckkod med en statisk längd. EAN-13 består av 13 siffror inklusive kontrollsiffra. De första två eller tre tecknen definierar en nationell kod, (Sverige har kod 73). Därefter följer 9 eller 10 siffror beroende på landskoden, där första blocket av siffror står för det levererande företaget, nästa block är reserverat för företagets egna artikelnummer. Den sista siffran är ett kontrolltecken. ISBN-numreringen som finns på alla böcker använder sig också av EAN-13 och koden har då ett prefix på 978. De

efterföljande siffrorna betecknar själva ISBN-numret. Varje tecken kodas av 7 lika stora moduler, där varje tecken består av 2 svarta streck och 2 vita tomrum, varje streck eller tomrum kan ha en bredd på 1, 2, 3 eller 4 element vilket ger ett storleksförhållande på 1:2:3:4. Rekommenderad X-dimension är 0,013 tum, men kan variera från 0.0104 upp till 0.0240 tum. Storleken på en normal (X = 0,013 tum) EAN-13 streckkod är 1,46 tum bred och 1,020 tum hög, förstorningar beroende på X-dimensionen kan variera på 80 till 200 % av normal storleken.

Symbolen delas vidare upp i två delar, varje del består av 6 siffror separerade av ett centrummönster och alla siffror omsluts av två vaktmönster. Dessa mönster känns igen på att de är lite längre än resterande element. Två likadana siffror kodas olika beroende på vilken sida om centrum de representeras på. Koden på vänster sida börjar alltid men ett tomrum och på höger med ett svart streck. En nackdel med EAN-13 är att den pga. sin struktur 1:2:3:4 kräver en hög precision vid utskrift och tryck, dvs. streckkoden är känslig för utspridning av trycksvärtan så att inte en modul av t.ex. bredden 2 kan misstolkas vara av en större bredd.

EAN numren är internationella, vilket innebär att samma nummer ska användas oavsett till vilket land som artikeln exporteras, utom USA och Kanada. Detta är ett problem för företag som exporterar och säljer produkter in och ut från USA och Kanada som då måste dubbelmärka sina produkter med två olika streckkoder. UCC har dock rört sig mot att gå över till EAN-13 standarden, den kommer att gå under benämningen UPC-13 och det betyder att varor som använder EAN-13 inte längre behöver byta streckkod för att säljas eller exporteras till Nord Amerika. UCC har meddelat att från och med den 1 januari 2005 måste alla avkodningssystem i Nord Amerika även kunna acceptera EAN-13.

Code 128

Figur 3.9 - Code 128

Code 128 är en kontinuerlig, alfanumerisk streckkod med variabel längd och hög datadensitet. Streckkoden lanserade 1981 och utvecklades av Ted Williams som ett svar på kravet av en kompakt alfanumerisk kod [41]. Grundkravet var att koden skulle kunna framställas med dåvarande teknik, ofta matrisskrivare. Möjligheten att kunna skriva koder med mellan 10 och 32 tecken framhölls som betydande. Koden ersätter ofta andra äldre koder, t.ex. Code 39, pga. dess mycket höga utrymmessnålhet samt den stora variation av tecken som går att använda.
Varje tecken kodas med hjälp av 3 svarta streck och 3 vita mellanrum. Elementen kan vara utav 4 olika storlekar, 1, 2, 3 eller 4 gånger det minsta elements bredd. Tabellen i Bilaga D visar kodningen för hela teckentabellen. Summan av de svarta elementens bredd för ett tecken är alltid jämn och alltid udda för vita element vilket används som bas för tecken kontrollen. X-dimensionen rekommenderas inte vara mindre än 0,0075 tum. Höjden skall vara minst 15 % av längden eller 0,25 tum och en den tysta zonen skalla vara 10 gånger X dimensionen.
Code 128 har en teckenuppsättning på siffrorna 0-9 och 128 tecken ur ASCII tabellen.
Numeriska värden kan dessutom kodas med dubbel täthet, dvs. 2 siffror kodas i ett tecken, vilket ger möjligheter för att ytterligare minska storleken på koden.

Code 128 är delad i tre olika typer, A, B och C. Där varje typ känns igen på sin individuella startsymbol. De olika typerna har alla speciella kontrolltecken som gör att det går att byta typ mitt i en streckkod. Typ A inkluderar standard ASCII symboler, siffror och versaler. Typ B inkluderar standard ASCII symboler, siffror, versaler och gemener. Typ-C komprimerar två siffror till ett tecken vilket ger en mycket kompakt symbolstandard.

EAN 128

Figur 3.10 - EAN 128

EAN-128 är baserad på Code 128 och har samma teckenuppsättning och teckenkodning. Det som skiljer dem åt är att EAN-128 använder sig av standardiserade applikationsidentifierare,

AI. Genom att använda dessa går det att ge innehållet en speciell struktur. AI används för att dela upp informationen och para ihop informationen med vad den representerar, dvs. olika AI. Standarden är utvecklad för att möjliggöra en större utväxling av information om produkten eller tjänsten än bara dess id-nummer. Exempel på AI är produktionsdatum, hållbarhetsdatum, serienr, varunr, antal, längd, bredd, fakturanummer, postnummer.
EAN-128 är inte tänkt för användning i detaljhandel utan används mest i system för lagring eller transport. Genom att para ihop en AI med information så talar man om för mottagaren, eller en part under transporten, var han hittar den information som behövs.
Ett exempel för att visa användningsområdet är t.ex. att koda vikten 40,36 kg i standarden Codabar genom att helt enkelt koda 40,36. Men en avläsande applikation vet inte om siffrorna står för en produkt kod, en tid eller kanske som i det här fallet, en vikt. Även om den avläsande applikationen visste att det handlade om en vikt går det kanske inte avgöra om det handlar om gram, kilo eller kanske pounds, det är här AI kan utnyttjas.

3.2.2 Sammanfattning 1D

De första typerna av streckkodsstandarder utvecklades med krav på framförallt en hög läsbarhet. Streckkoderna skulle också kunna skrivas ut med dåtidens matrisskrivare trots eventuell bläckutflytning som var vanlig på de flesta typer av utskrifter. Streckkoden skulle trots dessa problem ändå bibehålla ett bra tryck, vilket resulterar i en förenklad och högre läsbarhet. Att kunna ha en hög streckkodskvalité på dåtidens begränsade skrivare var en avgörande faktor för att kunna få en hög procentuell läsbarhet med den tidens tekniska resurser i form av avläsare. Den begränsade teckenuppsättningen på enbart siffror medförde att teckendensiteten snabbt utvecklades till en hög nivå. Nästa steg i utvecklingen blev att utöka teckenuppsättningen för att även införa bokstäver och utvecklingen har därefter gått mot att kunna lagra fler tecken i samma symbol, ur ett större urval av symboler och på en allt mindre yta.

Tabell 3.1 - Överblick över 1D streckkoder

Streckkod	År	Teckenuppsättning	Längd	Diskret/Kontinuerlig
Codabar	1972	Siffror (0-9) 6 special tecken ($-:/.+)	Variable	Diskret
Interleaved 2/5	1972	Siffror (0-9)	Variable	Kontinuerlig
Code 39	1975	Siffror (0-9) Versaler (A-Z) 7 special tecken (space-+.$/%)	Variable	diskret
EAN 13	1977	Siffror (0-9)	13 tecken	kontinuerlig
Code 128	1981	128 ASCII tecken	Variable	kontinuerlig
EAN 128	1989	256 ASCII tecken	Variable	kontinuerlig

Tabell 3.2 - 1D streckkoders densitet

Streckkod	År	X dim (milli tum)	Densitet (tecken/tum)
Codabar	1972	8.14	6.82
Code 39	1975	8.14	7.68
Code 128	1981	8.14	11.17
EAN 13	1989	8.14	17.96
Interleaved 2/5	1972	8.14	20.48

3.2.3 2D

Streckkodstekniken blev populär tack vare två egenskaper, snabb avläsning och hög precision. Allt eftersom streckkodernas förmånlighet blev förstådd, började marknaden fråga efter koder som kan lagra mer information, fler tecken, och som kan placeras på allt mindre förpackningar eller ytor.

Många försök gjordes för att öka datamängden som streckkoder kunde lagra, bl.a. genom att öka antalet siffror i streckkoden eller genom att använda multipla streckkoder. Men dessa försök resulterade endast i att öka streckkodens lagringsyta, komplicera avläsningen och öka kostnaderna för att tryck och utskrift.

En 1D streckkod representerar sällan mer än ett dussin symboler, problemet med ett 1D streckkodsystem som kan lagra hela alfabet eller mer är att streckkoderna då blir horisontellt långa jämfört med 2D streckkoder (figur 3.11). 1D streckkoder används därför mer som ett id till en plats i en databas där relevant information är sparad. De 2-dimensionella streckkoderna togs fram bl.a. för att lösa detta problem.

Code128, PDF417 respektive DataMatrix.
Alla representerar samma kod: ”12345678901234567890”
Med samma X-dimension

Figur 3.11 - Storleksjämförelse, 1D, 2D-staplad och 2D-matris

Det finns framförallt två olika pådrivare för utvecklingen av de 2D streckkoderna. På senare tid har även en tredje faktor tillkommit.

1. Den ena faktorn var krav på en ökad datakapacitet för att kunna utveckla och använda streckkoder som portabla datafiler. Detta krav växte fram för att slippa ha konstant tillgång till en databas, utan istället kunna lagra all information om produkten direkt i streckkoden vilket ger en större portabilitet då streckkoden i sig själv fungerar som en databas i sig själv. Ett exempel är att lagra namn, adress, telefonnummer, mobilnummer och annan information. 2D streckkoder har också fått en högre tillförlitlighet och säkerhet på grund utav sin ökade datakapacitet som tillåter införandet av extra data för redundans och felberäkningar.

2. Det andra skälet hade ett behov av att lagra lika mycket information som de traditionella 1D streckkoderna men på en mycket mindre yta. Detta skäl uppkom för att expandera streckkodernas användningsområde och för att utöka streckkoders användbarhet till andra områden som tidigare inte tillämpat tekniken. Elektronikindustrin visade snabbt ett stort intresse för 2D streckkoder med hög densitet eftersom fria ytor på kretskort och elektronikkomponenter är knapp.

3. Det tredje behovet som nyligen framkommit på marknaden är behovet av 2D streckkoder som med hjälp av bildanalys är enkla att avkoda i bilder med dålig kvalitet. Detta relaterar till fotografier tagna av gemene man med mobiltelefonkameror med liten upplösning. Hög datakapacitet och liten lagringsyta är inte primära framtagningsskäl för dessa tillämpningar. Få streckkoder har tagits fram för att tillgodose dessa ändamål.

2D streckkoder är uppdelad i två olika typer. Staplade eller multi-rader och matris typer. Staplade 2D streckkoder grundar sig på 1D streckkoder, bl.a. Code 39 som utvecklades till Code 49. Staplade streckkoder använder sig av tunna skivor utav 1D streckkoder. Skivorna läggs ovanpå varandra och skapar då multipla rader som bildar en staplad streckkod. Extra information inuti streckkoden kompletterar koden och indikerar de rader som innehåller kod. Code 49 och Code 16K utvecklades först, därefter presenterades PDF417 och då med utökad datakapacitet, förbättrad datadensitet och förbättrad läslighet. Dessa egenskaper medförde

även utökad feldetektering och även rättningstekniker. PDF417 är den vanligaste staplade koden.

En matriskod, som är en del av 2D streckkoder, är likt ett schackbräde. En matriskod är nämligen uppbyggd av ett flertal svarta och vita rutor eller celler. Cellerna kan vara utav olika mönster beroende på streckkodstandarden men är vanligtvis antingen fyrkanter, trianglar, cirklar eller andra former tagna ur geometrin. Data kodas sedan genom färgen på cellerna t.ex. svart eller vit. En matriskod kan också ses som en area av pixlar, där färgen på varje pixel kodar information. Matriskoder kan lagra 3 till 4 gånger så mycket information på samma yta som staplad streckkod och har den högsta datadensiteten utav de streckkodstyper som finns idag.

Matriskoder kan på grund utav sin komplexitet bara skannas av läsare med CCD teknik och igenkänningsmjukvara. Matriskod ger också avancerad feldetektering och felkorrigering med hjälp av bl.a. Reed Solomon algoritmer [23]. Det ger möjlighet att hitta felaktigheter i 2D streckkoder samt att rätta skadade 2D streckkoder. Upptill 35 % utav koden kan vara skadad och ändå ge en korrekt avläsning hos 2D streckkoder som använder sig av Reed Solomon. Om den 2D streckkoden inte går att avläsa stoppas avläsningen, det vill säga avkodaren till en 2D streckkod ger rätt kod eller ingen kod alls utan bara ett felmeddelande.

Matriskod är skalbar, stora tysta zoner behöver dessutom inte användas i samma utsträckning som vid 1D streckkoder vilket gör att de kan integreras med omliggande text eller grafik på ett bättre sätt. Anledningen till detta är att matris koder har ett unikt igenkänningsmönster som avläsaren initialt letar efter. Tex. MaxiCode har tre cirklar inneslutande i varandra, ett typ av ”bulls-eye” mönster. Det finns över 20 olika 2D symbolstandarder idag [3]. Nedan följer några av de mest populära och mest använda.

3.2.3.1 Staplade streckkoder

Code 49

Figur 3.12 - Code 49

Code 49 lanserades 1987 av Intermec Technologies [36] för att täcka ett behov för en symbol som kunde lagra en stor mängd data i en väldigt liten yta. Code 49 är den första 2D streckkoden som utvecklades. Skaparen David Allais designade streckkoden som en korsning mellan Code 39 och UPC. Code 49 är en multiraders, kontinuerlig streckkod med variabel längd som kan koda hela ASCII tabellen.

Varje symbol består av mellan 2 till 8 rader, raderna separeras av en skilje rad. Varje rad innehåller radnummer och den sista raden specificerar också det totala antalet rader i symbolen. Varje rad består av en tyst zon, ett inledande mönster, ett informations fält som kodar upp till 8 tecken. Slutligen ett stoppmönster följt av en tyst zon. Streckkoden kan skannas både av modifierade laseravläsare och CCD avläsare. Code 49 maximala längd är 49 alfanumeriska tecken eller i numerisktläge 81 siffror. I numeriskt läge går det dessutom att koda 5 siffror på samma längd som 3 alfanumeriska tecken, vilket ökar kodens densitet. Code 49 har en maximal densitet på 93 alfanumeriska tecken/tum eller 154 siffror/tum.

PDF 417

Figur 3.13 - PDF 417

PDF417 utvecklades 1991 av Symbol Techonlogies [48]. PDF417 är en 2D, staplad streckkod med variabel längd. Streckkoden är rektangulär, men formen kan ändras i viss utsträckning.

Symbolstandarden där PDF i namnet står för Portable Data File har en mycket hög datalagringskapacitet, den kan lagra upp till 1 850 ASCII tecken, 2 710 numeriska siffror eller 1 108 binära tecken per symbol och den kan representera alla tecken i ASCII standarden och 8-bit Binary Data.

En PDF417 symbol består av 3 till 90 rader omringad av en tyst zon och utan skiljelinje mellan raderna. Varje rad består av ett startmönster, en vänsterradsindikator, mellan 1 och 30 datatecken följt av en högerradsindikator och slutligen ett stoppmönster. Varje rad innehåller också radidentifierare som gör det möjligt att sätta ihop koden korrekt oberoende av i vilken ordning raderna läses in. Både längd och höjd är specificerbar vilket gör att symbolen går att reglera beroende på var den ska sitta och vad den ska användas till.

PDF417 har i binärt läge en maximal densitet på 686 bytes / kvadrat tum och i ASCII läge 1 144 tecken / kvadrat tum. Streckkoden utnyttjar då ingen felkorrigering. PDF417 har annars flera nivåer av skydd mot databortfall. Det finns 9 olika säkerhetsnivåer där den högsta nivån tillåter att hela 60 % av streckkoden är borta eller förstörd. En hög nivå ger en utmärkt datasäkerhet men gör också så att en stor del av informationen i streckkoden används för datakontroll, vilket leder till minskad maximal datakapacitet. Avkodningstiden ökar även den då fler felkontrollsberäkningar måste utföras.

3.2.3.2 Matriskoder

DataMatrix

Figur 3.14 - DataMatrix

DataMatrix är en 2D matris streckkod, utvecklad av CiMatrix [24] 1989 i syfte att lagra så mycket information som möjligt på en så liten yta som möjligt. DataMatrix representerar data

genom mörka och ljusa fyrkantiga datamoduler, den är variabel i längd och kan representera alla ASCII tecken, alla ISO [37] tecken samt alla EBCDIC [34] tecken. Streckkoden har en maximal datakapacitet på 2 335 text tecken, 3 116 siffror eller 1 556 bitar och är skalbar från 1 kvadrat mil till 14 kvadrat tum vilket tillåter upp till 50 tecken på en yta av 3 mm^2.

Det finns två underkategorier i DataMatrix. ECC 200 som använder sig av Reed Solomon feldetekterings metod och ECC 000 till 140 som använder sig av en mindre avancerad feldetekteringsteknik. Symbolen kan endast vara fyrkantig i version ECC 000-140, men i version ECC 200 kan streckkoden även vara rektangulär i utseende. DataMatrix är uppbyggd av minst 9 rader x 9 kolumner med ett max av 49 x 49 moduler för ECC 000-140 och ett max av 144 x 144 moduler för ECC 200.

En tyst zon på en X-dim omsluter symbolen helt. Två helt mörka linjer i ett ”L” mönster i utkanten till vänster och undersidan av symbolen används som igenkännings mönster. Ytterligare två streck på motstående sidor, varierande mörka och ljusa används för igenkänning av symbolens datadensitet och orientering. De fyra linjer omsluter i sin tur det området där informationen är kodad. Informationen kodas genom absolut punkt position jämfört med relativ position, det gör koden mindre känslig för felaktigheter i trycket jämfört med en traditionell streckkod då koden blir mer spridd över symbolen.

MaxiCode

Figur 3.15 - Maxicode

MaxiCode är en 2D matris streckkod utvecklad av United Parcel Service [51]. MaxiCode designades av UPS för ett behov av en streckkod som kunde användas för att sortera brev i hög hastighet och lanserades 1992. MaxiCode kan använda alla 256 tecken ur ASCII, i 5 olika kodtyper A till E. Matris streckkoden har en fast storlek med en tyst zon runt symbolen på en cell. Den kan lagra upp till 93 alfanumeriska tecken eller 138 siffror med en moduldimension

på 35 mil och på en yta av 1.1 x 1.05 tum. Streckkoden är uppbyggd av en matris av sammankopplade mörka och ljusa sexhörningar vilket gör den 15 % tätare än en streckkod med fyrkantiga celler men det kräver också en högre upplösning vid utskrift. Sexhörningarna, maximalt 866 stycken i 33 rader, är samlade runt ett igenkänningsmönster bestående av tre cirklar liknande en skottavla. Mönster gör streckkoden enkel att hitta oberoende av rotation på symbolen och gör MaxiCode tillsammans med dess exakta storlek speciellt anpassad för snabb avläsning.

MaxiCode är utvecklad med två valbara nivåer av feldetektering och korrigering, båda nivåerna använder sig av Reed Solomon. Streckkoden klarar en förstörelse av upp till 25 % och kan bara skannas av en läsare med CCD teknik eller liknande.

QR Code

Figur 3.16 - QR Code

QR Code står för Quick Response Code och är utvecklad i Japan av Denso Wave [32]. Streckkoden presenterades 1994, den är fyrkantig till formen bestående av ljusa och svarta fyrkantiga moduler och är en 2D matris streckkod. QR Code är av variabel längd och kan koda en mängd olika tecken. Förutom siffror och alfanumerisk tecken, kan QR Code även representera Japanska Kana-Kanji tecken. Max antal för en symbol är 4 296 alfanumeriska tecken, 7 089 siffror, 2 953 bytes eller 1 817 Kana-Kanji. Streckkoden är utvecklad för att snabbt kunna läsas av med hjälp av CCD kameror och bildbehandling. Streckkoden använder sig av ett igenkänningsmönster bestående av tre kvadrater som är placerade i tre av symbolens hörn. Kvadraterna har en svart ytterrad, en vit mittenrand och ett svart centrum.

Symbolen har minst 21 x 21 celler, en cell kodar 1 bit. Symbolen växer i steg av 4 celler med en maximal storlek på 105 x 105 celler och kräver en tyst zon på 4x modulbredden. QR Code

har fyra nivåer för feldetektering, och kan som hantera upp till 30 % av skadad kod. L nivån klarar 7 %, M 15 % och Q 25 %.

Aztec Code

Figur 3.17 - Aztec Code

Aztec Code [3] introducerades 1995 av Welch Allyn [30]. Aztec Code är en 2D matris streckkod bestående av ett fyrkantigt nät med ett större fyrkantigt ”bullseye” mönster i mitten av symbolen. Data kodas i lager runt omkring ”bullseye” fyrkanten i mitten. Varje lager omsluter mitten symbolen och ökar i storlek då mer information kodas in men behåller sin fyrkantiga form. Aztec streckkoden är skalbar och kan koda upp till 1 914 bytes från hela ASCII tabellen. Den minsta delen i symbolstandarden kallas för en modul, modul storleken bestämmas av användaren tillsammans med felkorrigeringen. Aztec symbol storlek kan varierar från 15 x 15 modules upp till 151 x 151 modules, där modulens storlek kan variera från 15 till 30 mils för att kunna skannas av dagens avkodare. Den minsta symbolen kodar då 13 numeriska eller 12 alfabetiska tecken och den största kodar 3832 numeriska eller 3067 alfabetiska tecken.

Aztec kod designades för att vara enkel att skriva ut och för att vara enkel att koda av. Streckkoden ger användaren möjlighet att själv bestämma hur stor Reed Solomon felkorrigering som skall användas. Rekommenderad nivå är 23 % av symbol kapaciteten, nivån går att variera från 5 % till 95 %.

3.2.4 Sammanfattning 2D

Utvecklingen av 2D streckkoder är ganska tydlig, den har gått från att vara en ganska enkel vidareutveckling från 1D streckkoder genom att stapla flera 1D streckkoder ovanpå varandra till ett mer matrisliknande format. Utvecklingen har därefter tagit fram streckkoder med allt större datalagrings kapacitet och även format med en allt mer distinktare och mer lätt identifierad detekterings symbol eller ”bullseye”.

Tabell 3.3 - Överblick 2D streckkoder

Symbol Standard	**Lanserades**	**Typ**	**Tecken Uppsättning**	**Max tecken /symbol**	**Densitet**	**Fel Detektering Kontroll**
Code 46	1987	Staplad	ASCII	49 tecken 81 siffror	93 tecken /tum	
Data Matrix	1989	Matris	ASCII ISO EBCDIC	2 335 tecken 3 116 siffror 1 556 bytes	500 tecken /tum	25%
PDF417	1991	Staplad	ASCII 8-bit bin data	1 850 tecken 2 710 siffror 1 108 bytes	1 144 tecken /tum	Upp till 50%
MaxiCode	1992	Matris	ASCII	93 tecken 138 siffror	100 tecken /tum	Upp till 25%
QR Code	1994	Matris	Alfa-num Kana-Kanji	4 296 tecken 7 089 siffror 2 953 bytes		7% till 30%
Aztec Code	1995	Matris	ASCII	3 067 tecken 3 832 siffror 1 914 bytes	1200 tecken /tum	5% till 95%

3.3 Reed-Solomon felkontrollkod

Feldetekterande och felkorrigerande kod är en sofistikerad teknik för att parera och neutralisera störningar vid lagring och sändning av data. Irving S. Reed och Gustave Solomon [14] introducerade redan 1960 några idéer som är kärnan för dagens felkorrigerings tekniker i allt från hårddiskar till CD-spelare. Reed-Solomon [23] koder gör det möjligt att lyssna på en repig musik cd och fungerar som ett säkerhetsnät, en matematisk försäkring, under dagens hårddiskinfiltrerande samhälle.

Nyckeln till felkorrigering är redundans. Det enklaste sättet att upptäcka fel vid dataöverföring är att skicka all information två gånger [9]. För att få en felrättande kod måste dock en dataström sändas minst tre gånger. Meddelandet 1 0 1 skulle bli 111 000 111 och om meddelandet då kommer fram som 110 010 111 kan mottagaren genom majoritetsbeslut rätta meddelandet till 1 0 1. Denna metod leder dock till att för n fel måste informationen sändas ($2*n + 1$) antal gånger [9]. Denna kraftsamligsmetod går dock emot själva iden om hög datahastighet och hög datadensitet. För att kunna minska datamängden som sänds behövs mer intelligenta kodningsmetoder.

En förbättrad metod för feldetektering är att lägga till en paritet på dataströmmen [52]. Denna bit anger om meddelandet innehåller ett jämt eller udda antal ettor. Detta kallas för jämn respektive udda paritet. Ett annat sätt att implementera enkel feldetektering på byte nivå är att använda sig av kontrollsummor. En vanlig implementation av kontrollsummor är att använda sig av modulo beräkning. En kontrollsumma läggs på, sådan att den tillsammans med den ursprungliga datan ger resten 0 vid en modulo beräkning. Detta är en metod som utnyttjas flitigt för feldetektering med 1D streckkoder. En mer avancerad teknik är Hammingkod som introducerades av Richard Hamming [52] ungefär samtidigt som Claude Shannon [18] myntade sin teori om informationssystem på 1940-talet. Hammingkod baseras på införandet av paritetsbitar på ett sådant sätt att de överlappar varandra och tillsammans täcker maximalt antal bitar. På så sätt går det korrigera alla en-bits-fel eller detektera alla två-bits-fel. Det finns dock ingen möjlighet att göra båda. Antalet paritetsbitar ges av Hammings regel: $d + p + 1 =< 2^p$, d är antal bitar information som ska överföras och p är antal paritetsbitar [31].

Kort efter Hamming formulerade Reed och Solomon en modell för felsökning och korrigering, metoden ligger på den teoretiska effektivitetsgränsen vilket gör att någon effektivare metod inte står att finna, den extra kod som krävs är minimal oavsett vilken nivå på felkorrigering man önskar uppnå. Reed-Solomon [14] arbetar med multi-bit symboler istället för att arbeta med enskilda bitar och är förhållandevis enkla att avkoda och är mycket lämpliga att använda för korrigering av skur-bit-fel samt multipla slumpmässiga fel. Av just denna anledning används Reed-Solomon koder för felkorrigering på bl.a. CD-skivor där den största orsaken till fel är repor och smuts vilket ger upphov till skur-bit-fel. Reed-Solomon koder är baserad på en speciell del av matematiken känd som Galois fält eller finita fält. Ett finit fält har den egenskapen att aritmetiska operationer (+,-,x,/ osv.) på fält element alltid har ett resultat i fältet. Ett Reed-Solomon kodord genereras genom att använda ett speciellt polynom. Alla giltiga kodord är exakt delbara av den genererade polynomen. Polynomets allmänna form är:

$$g(x) = (x-a^{i})(x-a^{i+1})\ldots(x-a^{i+2t})$$

Kodorden konstrueras med:

$$c(x) = g(x).i(x)$$

där g(x) är det genererande polynomet, i(x) är informations blocket, c(x) är ett giltigt kodord och a är ett primitivt element av fältet. Matematiken i Reed-Solomon kodning och avkodning är relativt komplex, för en detaljerad beskrivning hänvisas till [21] och [17].

3.4 Störningsfaktorer

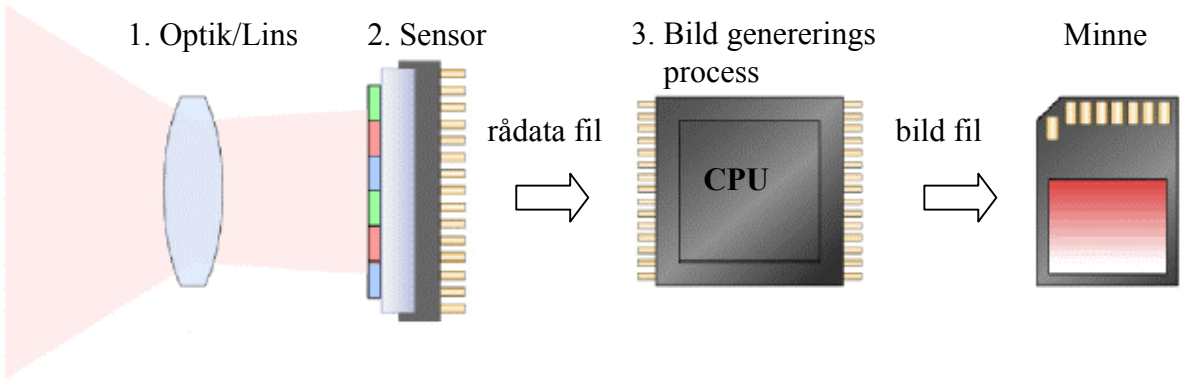
Den vanligaste typen av streckkodsavkodare är idag avläsare med laserteknik. Men om man tittar på utbudet av avkodare för detaljhandel och industri idag är det allt vanligare att utnyttja en avläsare med någon from av kamerateknik tillsammans med bildanalys för avkodning av streckkoden. Tekniken har fördelen att det går att avkoda både 1D och 2D streckkoder. En nackdel som man kommer ifrån genom att använda en digitalkamera vid avkodning istället för en laseravläsare är en fråga om avstånd, en traditionell laseravläsare har nämligen den nackdelen att avståndet mellan avläsaren och streckkoden måste vara nära noll för att en

lyckad avläsning skall kunna genomföras, även om det idag finns sofistikerad laserteknik för avläsning på flera meters avstånd. En digitalkamera som ger högkvalitativa bilder har en större tolerans mot avstånd. Men det finns andra problem med att använda en digitalkamera och dessa problem blir mer tydliga i ett kamerasystem för en mobiltelefon. De största problem med att avkoda en streckkod med hjälp av en mobiltelefonkamera är kvaliteten på bilderna och på grund utav olika typer av störningar och förvrängningseffekter kan bilden på en streckkod tagen av en kamera avvika från verkligheten och resultatet kan bli en felaktig avkodning eller omöjliggöra en korrekt avkodning helt och hållet.

Att ta ett fotografi är inte en enkel process. Det sker i flera steg och i varje steg finns det risker för försämringar av den datakvalitet som representerar bilden. De olika stegen som är involverade i ett system för korttagning avbildas i figur 2.2 i kapitel 2.1. Störningseffekterna kan delas in i tre olika grupper, kamerafaktorer, miljöfaktorer och tryck-kod faktorer. De största störningarna återfinns i framförallt linsen (kamerafaktorer) och i vinkeln mellan kodyta och kamera. Upplösning, skärpa och kontrast är också viktiga faktorer för att en lyckad avkodning av en streckkod skall kunna genomföras snabbt och utan alltför stora resurser.

3.4.1 Kamerafaktorer

Förvrängningseffekterna som uppstår på bilder tagna med en mobilkamera beror på hur kamerasystemen är uppbyggda och är även vanliga i digitalkameror och till viss del även i analoga kameror eftersom de är uppbyggda av ungefär samma komponenter. En mobilkameras bildmodul består, precis som vilken annan digitalkamera av tre huvudkomponenter.



Figur 3.18 - Digital fotomodul

1. Optik/Lins: Huvuddelen i vilket kamerasystem som helst, dess funktion är att koncentrera ljusstrålar som representerar den scen eller det objekt som skall fotograferas. Linsen projicerar ner ljusstrålarna på ett känsligt element av något slag, en sensor. Om linsen bryter ljusstrålarna felaktigt så att ljuset före eller efter sensor ytan blir bilden oskarp. Linsen i ett kamerasystem fokuserar aldrig 100 % perfekt, det finns alltid en viss oskärpa i bilden även med den bästa fokuseringen, ex. ytterkanterna.

2. Sensor: En sensor är gjord av pixlar och konverterar ljusinformationen som fås via linsen till elektroniska signaler. En sensor känner bara av ljusintensiteten och tar inte upp någon information om färg, utan för att få fram en färgbild används tre olika filter, röd, grön och blå, som placeras framför sensorns pixlar. En bild ger en representation av en scen eller ett objekt, antalet pixlar ger i viss mån ett mått på äktheten av skildringen. Avståndet mellan pixlarna begränsar upplösningen i bilden, men en stor mängd pixlar är inte alltid att föredra, om ljusnivån in till sensorn är låg, sprids ljusintensiteten över en större mängd pixel sensorer och ger därmed en bild med sämre kvalitet än om pixelmängden varit lägre. Det finns framförallt två typer av sensorer CCD och CMOS (Complementary Metal Oxide Semiconductor). Bildsensorer skiljer sig åt i avseende på storlek, ljuskänslighet, upplösning och energiförbrukning. CCD sensorer används oftast i digitalkameror på grund utav dess överlägsna kvalitet. CMOS är mer vanlig i mobiltelefonkameror för att den drar mindre energi, upp till 100 gånger mindre [4], kretsen tar också mindre plats och har alla komponenter

integrerade i en krets. Ytterligare fördelar är dess låga produktionskostnad. Nackdelen är att CMOS har en lägre upplösning och ger oftast större förvrängningseffekter och brus, pga. dess uppbyggnad.

3. Bildgenereringsprocessen: För att skapa en visuell bild från informationen given av sensorn genomgår informationen en komplex matematisk behandling. För att få en bild med hög kvalitet räcker det nämligen inte bara med att ha så många pixlar som möjligt. Råinformationen som fås från sensorn: färger, kontrast, skärpa, detaljer, måste kombineras ihop av till exempel en mikroprocessor för att skapa en digital synbar bild. Ett system med dålig optimerad mjukvara kan ge märkbara skillnader i bildkvalitet jämfört med ett system med mer avancerad och påkostad mjukvara.

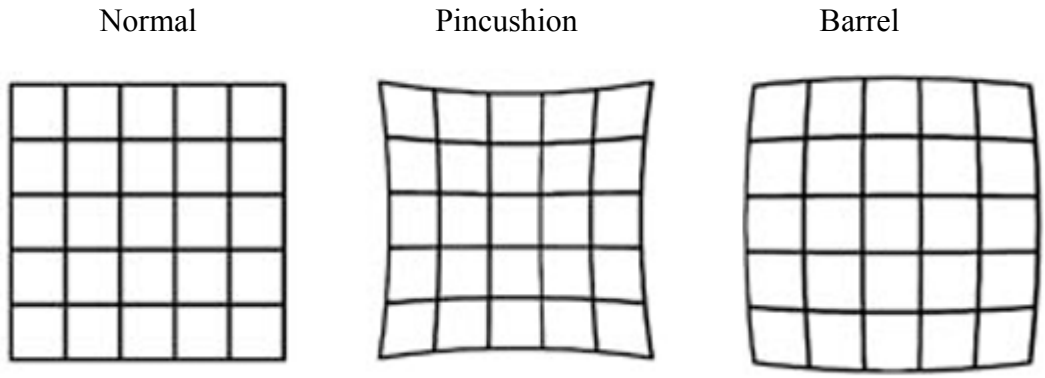
Slutligen skall bilden lagras, för att lagra så många bilder som möjligt komprimeras ofta bilden och formatet som vanligtvis utnyttjas är JPEG. Genom matematisk bearbetning är det möjligt att reducera bildfilens storlek markant men det medför dock även en nackdel. Problemet med komprimeringstekniker är att de försämrar bildens kvalitet och är som ett filter utanpå bilden som reducerar bildens upplösning.

3.4.1.1 Linsförvrängningar

Modern optik presenterar normalt en mycket bra bild, men helt perfekt är inte bilden. Optiska linser har en rundad yta för att kunna fokusera ljuset ner till en sensor. Den rundade ytan gör det svårt att presentera en helt platt bild. Från centrum och ut till kanterna av fotot tenderar nämligen bilden att tänja på sig, jmf fisheyeobjektiv eller starka vidvinkelperspektiv. Denna typ av förvrängning påverkar dock de flesta bilder bara minimalt, men fenomenet är däremot mycket vanligt.

Två andra typer av förvrängningar som ger upphov till större förvrängningar är ”Pincushion” och ”Barrel” effekter. Dessa båda beror på att centrum och kanterna får olika stora förstorningar i avbildningsvyn genom linsen. Skillnaderna gör så att bilden förvrängs från centrum och utåt mot kanterna, vilket gör att perspektiven i olika typer av former dras ut.

”Barrel” förvrängningar (se figur 3.19) förstorar upp centrum av bilden och får den att till synes bukta ut, raka streck avbildas som aningen sneda. ”Pincushion” (se figur 3.19) representerar motsatsen då centrum förstoras mindre än kanterna vilket får en inbuktande effekt. Dessa typer av förvrängningar ger särskilt upphov till problem i applikationer som mäter avstånd, eller som letar efter t.ex. raka linjer i bilder. ”Barrel” och ”Pincushion” ger dock inte upphov till någon förlust av bilddata. Informationen finns fortfarande kvar i bilden, den har bara blivit felplacerad vilket gör att effekterna kan bli korrigerade med hjälp av särskilda filter.

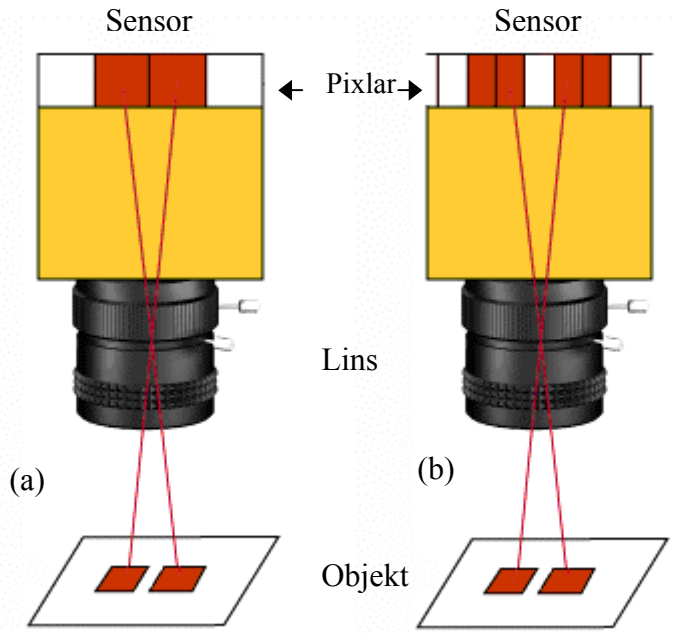


Figur 3.19 - Pincushion och Barrel effekt

3.4.1.2 Upplösning

Sensorn i en digital kamera är uppbyggd av pixlar som är små ljuskänsliga punkter. Sensorerna i de nya vanliga mobilkamerorna idag är gjorda av miljoner av pixlar, där pixlarna registrerar ljusstyrkan hos ljuset som fångas upp när ett foto tas. Antalet pixlar i en bild är ungefär lika med antalet pixlar i sensorn. Detta antal anger en bilds upplösning, desto högre mängd pixlar desto högre upplösning och desto högre upplösning desto fler detaljer kan representeras i en bild. Upplösning anger också ett bildsystems möjlighet att särskilja mellan två olika detaljer som sitter nära varandra. Ett system med hög upplösning visar också exempelvis en kantövergång i färre pixlar än en sensor med låg upplösning.

Figur 3.20 visar en förenklad bild av två kvadrater som blir avbildade av en CCD kamera. I figur 3.20a avbildas de två kvadraterna som en rektangel av systemet på grund utav att pixel upplösningen inte kan skilja dem åt. Det vita utrymmet mellan fyrkanterna upptäcks helt enkelt inte med den angivna upplösningen. För att särskilja kvadraterna behövs ett längre vitt avstånd mellan kvadraterna eller en högre upplösning där en pixel även representerar det vita mellanrummet (figur 3.20b).



Figur 3.20 - Avbildnings exempel

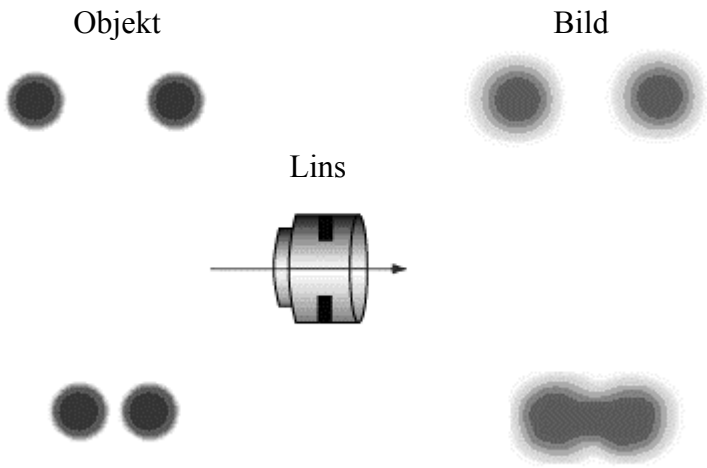
Upplösningen är ett mått på hur bra kort en digitalkamera kan göra, och måttet definieras av antalet pixlar som bygger upp en bild, dvs. antalet punkter av ljus. Kvaliteten som kan tas av en mobilkamera beror på sensorns upplösning. Ett större antal pixlar betyder vanligtvis en bild som producerar en bättre kvalitet och som återger fler definitioner i bilden. Om antalet pixlar är otillräcklig kommer resultatet av den dåliga upplösningen att bli uppenbart synlig i bilden genom en kornighet i fotografiet. Enbart ett systems upplösning anger dock inte systemets hela upptäckningsgrad av detaljer. Graden påverkas nämligen även av effekter som linsförvrängningar och bristfälligheter i ljus.

Den bästa upplösningen i Europa och USA ligger idag runt 1.0 till 1.3 megapixel (1 152 x 864 och 1 280 x 960 pixlar). Vid denna upplösning ges bilder med en mycket god kvalitet. God nog även för utskrifter. Utvecklingen går dock kontinuerligt framåt vilket visades på CeBIT mässan i tyskland i år (2005) där Samsung presenterade en mobilkamera med en upplösning på 7 megapixel [45]. Upplösningen går också att variera i de flesta mobiltelefonkameror. Användaren kan oftast välja mellan 2 till 3 olika nivåer av upplösning för att på så sätt kunna skräddarsy fotot beroende på situationens krav. Andra upplösningsformat som är vanliga är 352 x 288 pixlar, som ger en total bildupplösning på 101 376 pixlar. Denna upplösningsnivå kallas för CIF standarden och kom med de allra första mobiltelefonkamerorna. Upplösningen räcker i stort sett endast till för visning på en mobiltelefonskärm, men knappast till utskrifter eller liknande. En standard som ger bättre upplösning är VGA kameror som ger bilder i formatet 640 x 480 pixlar. VGA standarden ger en upplösning på 307 200 pixlar, 3 ggr bättre än CIF och ger klar bättre fotografier. Under 2004 var de flesta sålda mobilkameror utrustade med en VGA sensor och de är idag en vanligt förkommande upplösning hos mobiltelefoner med kamera.

Upplösning och kontrast nämns ofta var för sig med de är också nära besläktade, upplösning är mängden av detaljer som återges av en bild men att avbilda ett objekts kontrast är minst lika viktigt som ett objekts upplösning. Kontrast anger storleksskillnaden mellan ljusa och mörka element i en bild. En bild med hög kontrast blir och verkar skarpare än en bild med låg kontrast även om bilderna har samma upplösning. För att en bild skall framträda tydligt skall de svarta detaljerna i objektet verkligen vara svarta, de vita detaljer vara vita och alla nivåer av grått i mellan måste också vara korrekt representerade. Skillnaden mellan ett objekt och nivåerna av gråhet som t.ex. representerar skuggorna i bakgrunden är även den en viktig faktor vid upplösning och bildkvalitet.

Ett exempel: antag att två prickar är placerade relativt nära varandra och avbildas av en lins (se figur 3.21 nedan). Medan dessa två punkter närmar varandra, förenas de tillslut och punkterna blandas ihop. Punkternas mönster kombineras ihop i en region, som gör punkterna svårare att särskilja. Medan punkterna kommer allt närmre varandra blir det allt svårare att skilja punkterna åt, regionen mellan punkterna smälter nämligen samman alltmer och blir allt

mörkare. Trots en hög upplösning gör en dålig kontrast att det blir svårt att särskilja de två punkterna. Det blir i stort sett omöjligt att avgöra var den ena punkten slutar och var den andra börjar. Bildsystemet behöver hitta en kontrastskillnad i övergången mellan punkterna och det vita fält som skiljer dem åt för att kunna avgöra om området representerar två små punkter eller en större punkt.



Figur 3.21 - Kontrast

Faktorer som lins, sensor och belysning spelar alla en stor roll för bildens kontrast. En lins kontrast definieras i procent utav ett objekts verkliga kontrast som är återgiven. En sensors förmåga att återskapa kontrast specificeras i decibel i analoga kameror och i bitar hos digitala kameror. Upplösningen i ett system beror inte bara på antalet pixlar utan även på otydligheter som skapas av optiska felaktigheter, punktmellanrum och ett systems förmåga att upptäcka kontrast.

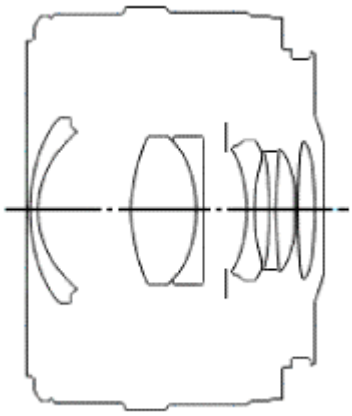
3.4.1.3 Komponenter

Mobiltelefoner med kamera skall vara små enligt användarundersökningar [20]. Det är viktigt att mobiltelefonen får plats i handen eller i fickan. Användarna vill inte att deras mobiltelefoner skalla vara stora som vanliga kompakta kameror, även om dessa också

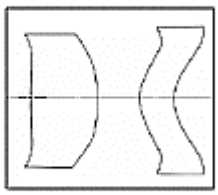
betraktas vara av en mindre storlek. Krav på små komponenter leder till att optik, sensorer och andra komponenter som används i digital och analoga kameror måste krympas för att användas i en mobiltelefonkamera. Att minska komponenternas storlek och samtidigt bibehålla högkvalitativa bilder ut från komponenterna är inte enkelt, det är kostsamt och medför komponenter som ger sämre bilder jämfört med fullstora digitalkameror. Kostnad, yta och energiförbrukning är tre begränsande faktorer för komponenter i ett mobilkamerasystem.

Dessa tre faktorer, kostnad, storlek och energiförbrukning är alla bidragande orsaker till att komponenter som CMOS-sensorer ofta används i mobilkameror. CMOS-sensorer drar mindre energi, är billigare att producera och tar mindre plats jämfört med CCD-sensorer som däremot har en större upplösning och ger bättre bilder.

Minskade kostnader tillsammans med krav på så små storleksförhållanden ger också restriktioner på linssystemen. Det medför krav på att bildsystemen som används i mobilkameror skall vara enkla system med så få linselement som möjligt. Vanligtvis används enkla, dubbla eller i bästa fall tre linselement. Figur 3.22 visar skillnaden i komplexitet mellan ett optisktsystem för en avancerad digitalkamera och en enkel kamera för en mobiltelefon.



Optiskt system för en Canon digitalkamera.[1]



Enkelt optiskt system för en mobilkamera.[1]

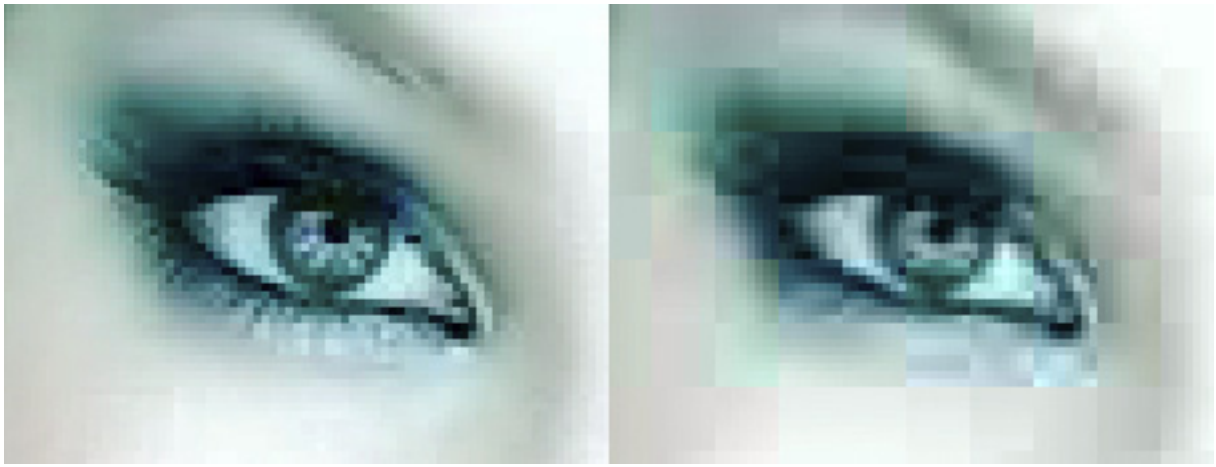
Figur 3.22 - Optisktkamerasystem

[1] Figur som visar skillnaden i komplexitet mellan två optiska system, en systemkamera och en mobilkamera. Storleksförhållandet mellan systemen överensstämmer ej.

3.4.1.4 Komprimering

Ytterligare en begränsande faktor för bildkvalitet är komprimering som ofta används i digital- och mobilkameror för att göra det möjligt att spara fler bilder i mobil eller digitalkameran. När man komprimerar bilddata kan man göra på två olika sätt, förstörande och oförstörande komprimering. Med förstörande komprimering menas att filen tappar i kvalitet när den öppnas igen och används. Det blir förvrängningar i bilden vilket inte alltid är så lyckat, men med förstörande komprimering kan man pressa ihop filen betydligt mer än med oförstörande komprimering. Eftersom det finns många tillfällen då det nästan inte alls märks att kvaliteten försämrats, kan försämringen vara ett billigt pris att betala för att få ned storleken. Det finns en praktisk skillnad också, en oförstörande komprimering använder mer processor kraft vid packning och uppackning men bevarar då i gengäld kvaliteten. Medan den förstörande komprimeringen är lättare att jobba med för datorn.

De flesta mobilkameror använder sig av industristandarden JPEG för att spara bilder, namnet står för Joint Photographic Experts Group. JPEG komprimerings tekniker har blivit mycket populär tack var dess höga komprimeringsgrad och används i en mängd olika områden. JPEG används bland annat i digitalkameror, färgskanners, på webben eller för att skicka bilder via e-post. JPEG är en metod för att komprimera pixeluppbyggda bilder och använder sig av en komprimeringsteknik som leder till en försämrad bildkvalitet. En bild som sparas i JPEG format kommer att förlora en del av bildinformationen på grund utav hur JPEG komprimerar bilderna. JPEG använder sig av en metod som bygger på hur det mänskliga ögat uppfattar små färgvariationer mindre exakt än småförändringar i t.ex. ljusstryka. JPEG utnyttjar detta genom att medelvärdesbilda närliggande pixlar av färgdata ner i ett block. Bilden delas upp i ett jämt fördelat rutnät där pixlarna i varje ruta jämförs mot varandra och sedan läggs samman, desto högre kompressionsgrad desto större rutor i nätet. När bilden sedan packas upp fördelas den sparade färginformationen inuti rutan och nyanserna smetas ut och det får till följd att detaljerna försvinner.



Figur 3.23 - Okomprimerad bild kontra JPEG komprimerad

Den använda komprimeringslogiken betyder att varje gång den används försvinner en liten bit av information, en del av detaljerna smetas ut och förloras. Bilden förlorar framförallt fina detaljer som t.ex. kanter och andra skarpa objekt. Mängden av förlorad information bestäms av en kvalitetsfaktor. Bilderna kan med hjälp av JPEG komprimeras i upp till 100 olika kvalitetsnivåer. Är komprimeringsgraden mellan ca 10:1 och 20:1 är det svårt för det blotta ögat att se någon kvalitetsförsämring. Vid komprimeringsgrader på ca 30:1 till 50:1 börjar försämringen märkas och vid ca 100:1 är bilden rent utav dålig.

Utav dessa nivåer använder digitalkameror oftast upp till tre olika komprimeringsnivåer, hög, normal och låg. Detta betyder att användaren har möjlighet att påverka bildens kvalitet. Att spara bilderna med hjälp av så låg komprimeringsgrad som möjligt är att föredra. Igenkänning av streckkodssymboler bygger på analys av små detaljer i bilden, dvs. möjligheten att hitta kanter och mäta avstånd från kant till kant eller dimensioner på speciella kännetecken. En bild med hög information är också enklare att förbättra med hjälp av olika typer av filter t.ex. för ökning av skärpa m.fl.

Ett utav de stora fördelarna med att ta bilder med sin mobiltelefon ligger i möjligheten att kunna dela med sig av de bilder man tar snabbt och enkelt via MMS. Egenskaperna för att skicka MMS i avseende på storlek m.m. är dock olika mellan mobiltelefonmodeller, komprimeringen av bilderna i vissa modeller är större än andra, och flera

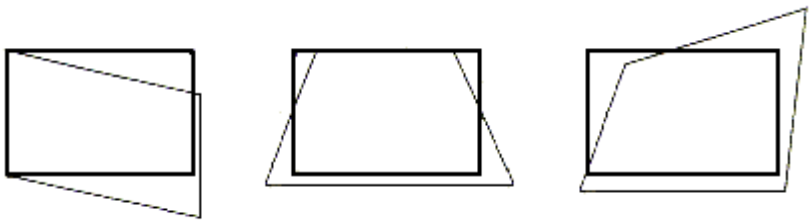
mobiltelefonleverantörer tillåter idag inte MMS med bildstorlekar över 300 kb, vilket resulterar i att bilden inte kan skickas alls eller att en hög komprimeringsgrad tillsammans med en konvertering till en lägre upplösning måste utnyttjas, både dessa faktorer sänker kvaliteten på bilden.

3.4.2 Miljöfaktorer

Om objektet som fotograferas inte är alldeles platt eller helt upprätt gentemot kameran kan det ge resultatet att delar av bilden inte kommer att vara i fokus, vilket resulterar i oskärpa.

3.4.2.1 Kameravinkel

Vinkeln mellan kameran och objektet kan orsaka perspektivförvrängningar i olika format. Förvrängningar uppstår då en del av ett objekt är närmre kameran och därmed kommer att uppta en större del av bilden än delen som är längre ifrån kameran (se figur 3.24). Denna typ av perspektiv fel är även det fenomen som vår hjärna låter tolka som den tredimensionella världen, vi förväntar oss att ett objekt som är nära ska verka vara relativt större än en objekt som är längre bort. Detta fenomen finns alltså även i avbildningssystem, där ett objekts storlek ändras beroende på avståndet till linsen. Perspektiv fel är som mest bekymmersamt i applikationer som mäter avstånd i en bild.



Figur 3.24 - Perspektivfel

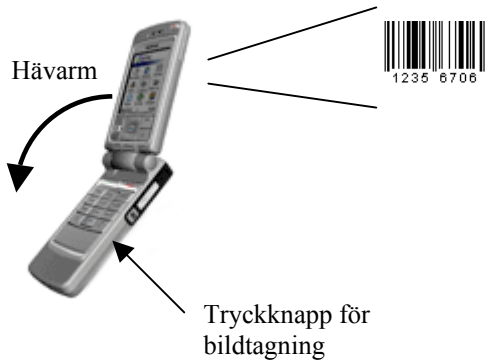
3.4.2.2 Luminans

De flesta vanliga kameror är stora förbrukare av batterikraft med t.ex. blixtar, digitalkameror med on-screen displayer m.m. För att kringgå en alltför stor energiförbrukning i mobiltelefonkameror är det vanligt att ersätta CCD sensorer med CMOS sensorer. CCD sensorer har större pixelkapacitet men drar mer ström p.g.a. att fler elektroniska enheter måste utnyttjas. CMOS sensorer däremot som har all elektronik i en enda modul och använder därmed mindre ström. Blixt är en komponent som drar mycket ström och används därför sällan till mobiltelefoner. Men en mobiltelefonkamera skall kunna används i alla typer av förhållanden precis som en digitalkamera, därför måste en mobilkamera även klara av förhållanden som dåligt upplysta platser, dåligt väder och fotografering inomhus. Alla dessa situationer är förhållanden där en blixt skulle kunna utnyttjas för att ge förbättrade bilder. Om blixt ändå används i ett kameramobilsystem utnyttjas framförallt ljusförstärkning med hjälp av LED dioder för att dra ner på energiförbrukningen, en konstruktion med hjälp av LED dioder drar upp till 10 gånger mindre ström än en konstruktion med blixtlampor [20] [21], ljusstyrkan blir dock med LED upp till 400 gånger lägre.

För att fånga upp så mycket ljus som möjligt utnyttjas därför även andra lösningar. En vanlig konstruktion och lösning även vid traditionell fotografering är framförallt att använda mikrolinser med en mycket stor bländaröppning. En sådan mikrolins förstärker dock också de naturligt optiska effekter så som förvrängningar och bländning. Ytterligare en funktion som utnyttjas vid mörka miljöer är stor bländaröppning och lång slutartid. Bländaren bestämmer hur mycket ljus som kommer in i kameran och slutartiden bestämmer hur länge ljuset släpps in. Bländaren påverkar även skärpedjupet i bilden, stor bländaröppning ger ett litet skärpedjup och försvårar möjligheterna att ta en bild med bra skärpa och kontrast. Lång slutartid resulterar även det i vissa degraderingar av bildkvaliteten, långt ljusinsläpp resulterar nämligen ofta i rörelseoskärpa.

3.4.2.3 Rörelseoskärpa

Mobiltelefonkamerorna är små och lätta, vilket medför att minsta darrning på handen i tagningsögonblicket överförs till kameran. Ofta sitter också själva kameran ute vid änden på mobiltelefonen (se figur 3.25), detta medför att det skapas en hävarm mellan kamera och användarens hand. En lång hävarm förstärker alla rörelser och gör dem större, en mobilkamera med lång hävarm är därför svårt att hålla absolut stilla vid själva fotograferingsögonblicket. Det är dessutom mycket lätt att oavsiktligt röra på kameran när man trycker av. Till detta kommer att många mobilkameror har en viss fördröjning från det att man trycker av till dess att bilden faktiskt tas vilket gör att användaren kan luras att börja röra på kameran innan bilden är färdig. Ofrivilliga kamerarörelser under exponeringen resulterar ofta i rörelseoskärpa även i bra ljusförhållanden. I sämre belysning, när hela bländaröppningen måste utnyttjas och exponeringsautomatiken gör att långa slutartider används, ställs stora krav på att kameran hålls stilla.



Figur 3.25 - Rörelseoskärpa

3.4.2.4 Avstånd mellan objekt och kamera

Avståndet mellan mobiltelefonkameran och objektet som fotograferas har en viss inverkan på möjligheten att avkoda en 1D eller 2D streckkod i ett foto. Desto längre det är mellan mobiltelefonkameran och ytan där den 1D eller 2D streckkoden sitter desto mindre bildyta används till att representera själva streckkoden, detta medför att färre antal pixlar används för att representera den 1D eller 2D streckkoden i bilden. Ökat avstånd ger därmed ungefär

samma effekter som förminskad upplösning. Den 1D eller 2D streckkoden bör därför uppta så mycket av bildytan som möjligt utan att bidra till en försämrad fokus i bilden.

3.4.3 Tryck

1D och 2D streckkoder är enkla och billiga att reproducera. Att skriva ut eller trycka 1D eller 2D streckkoder på papper är inte alls komplicerat eller dyrt. Pappersetiketter med streckkoder på finns på mängder av olika föremål och objekt så som t.ex. tidningar, kartonger, PET-flaskor m.fl. Var streckkoden sitter på ett objekt och hur objektet ser ut eller vad den har för form medför konsekvenser vid fotografering och avkodning med hjälp av en mobiltelefonkamera. Ett runt objekt som t.ex. en flaska medför en böjd etikett och därmed även en förvrängd streckkod vid fotografering. Perspektivförändringar i bilden är ofrånkomliga oberoende av kamerans X och Y vinkel mot objektet. Effekten blir att streckkodselement utmed kanterna tycks krympa i storlek i jämförelse med elementen i mitten på flaskan både på bredden och på längden. På grund utav denna effekt som ges utav runda föremål så som flaskor trycks alltid en 1D streckkod roterad och lagd på höjden på dessa produkter. Effekterna av en flaskas böjda form minimeras därmed. Effekterna går dock inte eliminera för 2D streckkoder på detta sätt då 2D streckkoder lagrar data både på höjden och på bredden.

Figur 3.26 - Simulerad bild av en 1D streckkod på ett runt föremål, t.ex. en flaska.[2]

[2] I figur har streck i mitten normal storlek, motsvarande streck utmed kanterna har blivit förminskade pga ett längre avstånd till kameran.

3.4.4 Sammanfattning brusfaktorer

Det finns en mängd olika faktorer som påverkar kvaliteten på en digitalbild utav en streckkod. Olika typer av faktorer förändrar eller förvränger den tagna bilden på olika sätt, vissa effekter märks också mer än andra. Tillsammans kan de dock göra så att en avkodning inte är möjlig att genomföra. Effekterna kan delas in i olika delar. Huvudgrupperna kan ses som kamerafaktorer, miljöfaktorer och tryck/kod faktorer. Inom dessa grupper finns dessutom en rad olika delar. Kamerafaktorer består av flest delar, optik/lins ger upphov till olika typer av förvrängningar, sensor ger upplösningen i systemet, och bildgenereringsprocess ger bl.a. komprimeringseffekter. Miljöfaktorer handlar mest om olika avstånd, rörelse, ljusstryka samt olika typer av vinklar gentemot den fotograferade ytan som ger upphov till perspektiv fel. Sist är kod/tryck vilket företräder kvaliteten på tryck, typen av ytan koden sitter på samt typen av kod. Alla effekter ovan kan spela in och ge upphov till effekter som gör streckkoden mer eller mindre svår att avkoda.

4 Metodutveckling och implementation

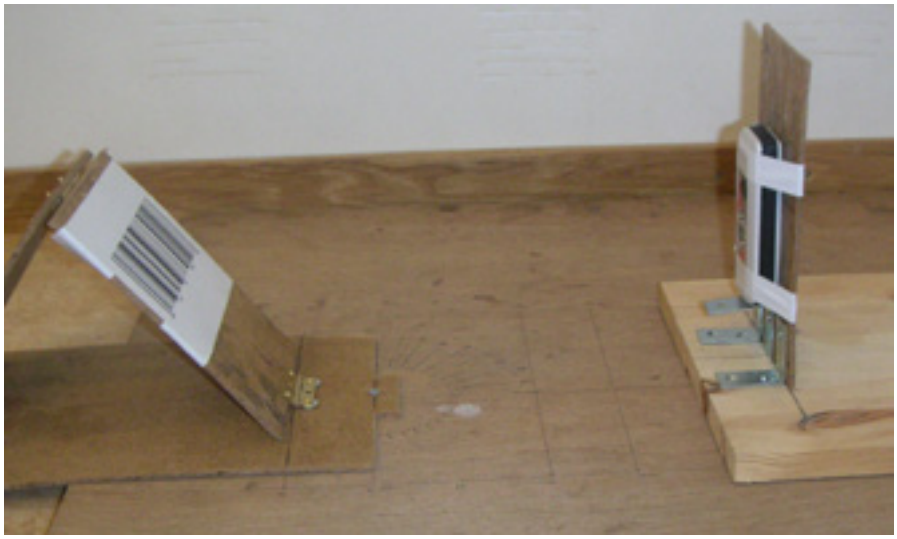
4.1 Inledning

Grunden i en avkodningsalgoritm för igenkänning av en 1D eller 2D streckkod från en kamera är inte helt oväntat en digital bild. Därefter kan avkodningen eller igenkänningen ske på flera olika sätt beroende på vilken typ av streckkod som skall avkodas samt även beroende på utvecklares tillvägagångssätt eller på systemets uppbyggnad och struktur. I detta kapitel tas dessa delar upp tillsammans med ett experiment som skall ge ett samband mellan datorgenererade bruseffekter och bruseffekter från riktiga bilder tagna med olika mobiltelefonkameror. Därefter kommer en beskrivning på den metod som tagits fram för att praktiskt undersöka olika 1D eller 2D avkodningsprogram mot varandra. Själva avkodningen kan ske på olika sätt i olika avkodningsalgoritmer. Två olika programutvecklare som oberoende av varandra utvecklar två liknande applikationer löser sannolikt inte samma uppgift på samma sätt varvid olikheter och även möjliga prestandamässiga olikheter uppstår mellan de båda programmen. Algoritmer från en viss utvecklare kan kanske dessutom vara bättre på att parera vissa typer av bruseffekter jämfört med andra algoritmer. Tanken med metoden i denna uppsats är därför att på ett snabbt och enkelt sätt kunna göra ett test av algoritmernas robusthet gentemot avkodningsrelaterade problem som kan uppstå för 1D och 2D streckkoder. För att testa metoden har dessutom tre olika avkodningsprogram skapats med hjälp av open-source avkodningsalgoritmer. Två applikationer, dBarScanJ och Papier-Mâché avkodar den 1D streckkoden EAN13, en applikation ETH avkodar en tämligen nyutvecklad 2D streckkod kallad Visual Code [16] [15]. Metoden används sedan för att jämföra avkodningsalgoritmernas möjlighet att avkoda streckkoder i olika nivåer av förvrängningar.

4.2 Experiment - äkta och datorgenererade bruseffekter

Figur 4.1 - Modell ovanifrån

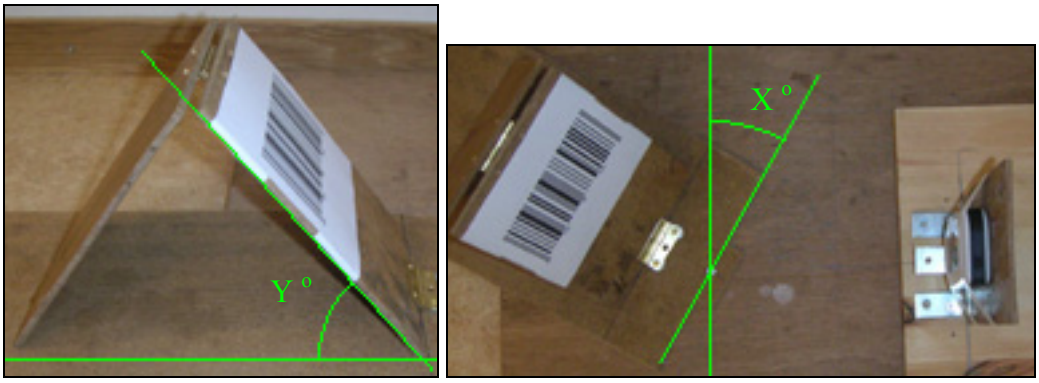
En metod har tagits fram och med dess hjälp jämförs olika avkodnings algoritmer av 1D och 2D streckkoder mot varandra. Metoden genererar ett antal bilder som används för referens och test. Bilderna genereras utifrån streckkoden som algoritmen har kapacitet att avkoda. Med hjälp av de genererade bilderna skapas ett underlag som kan användas för att visa vilken algoritm som klarar av att avkoda flest antal genererade bilder. Störningarna genereras med hjälp av ett professionellt bildbehandlingsprogram, Photoshop 7.0. Men för att kontrollera att bilderna som genereras i metoden kan skapas av verkliga förhållanden jämförs de genererade bilderna gentemot bilderna tagna i ett laborativt experiment. De bilder som undersöks är olika upplösningsnivåer samt perspektivförvrängningar.



Figur 4.2 - Modell sidan

Två olika mobiltelefonkameror används för att i olika vinklar och avstånd skapa bilder med olika perspektivförvrängningar och med olika långa avstånd mellan kamera och streckkod. Dessa bilder jämförs sedan med de datorgenererade bilderna som skapas i metoden. De kategorier som undersöks här är horisontella och vertikala förvrängningar samt upplösning. Modellen eller ställningen som används för att ta bilderna ser ut som i bilden i figur 4.1 och 4.2. Ställningen används för att kunna ställa in lutning, vridning och avstånd mellan kamera och streckkod samt för att fixera kamera och streckkod under bildtagningsserierna. Bilden i figur 4.1 och 4.2 visar till höger ställningen som är till för att hålla mobiltelefonen i ett fast läge under experimentet (se även figur 4.3a). Mobiltelefonens ställning kan vidare flyttas framåt och bakåt för att förskjuta avståndet mellan mobiltelefonkamera och streckkod. Streckkoden sitter även den fast på en ställning till vänster i figur 4.1 och 4.2. Denna ställning, figur 4.3b, går att vinkla gentemot kameran i X och Y led (se figur 4.4). Ställningen och modellen i sin helhet är en träkonstruktion med vissa detaljinslag av metall.

Figur 4.3 - Kamera- och streckkodsställning



Figur 4.4 - Y-axel och X-axel

Det är svårt att mäta en specifik störningsfaktor i en bild tagen med en mobiltelefonkamera och jämföra enskilda störningar med likartade simulerade effekter som är datorgenererade. Detta beror på att alla störningsfaktorerna, varav flera har nämnts tidigare i denna rapport, inte går att utesluta ur den tagna bilden utan påverkar jämförelsen med de datorgenererade bilderna. De bilder som genererats av experimentet finns i bilaga H, i följande kategorier:

1. Avstånd - Siemens. 6 bilder, från 15 cm till 40 cm mellan kamera och streckkod med steg av 5 cm.

2. Horisontell förvrängning (Y) - Siemens. 12 bilder, från 10 grader till 65 graders Y vinkel i steg på 5 grader.

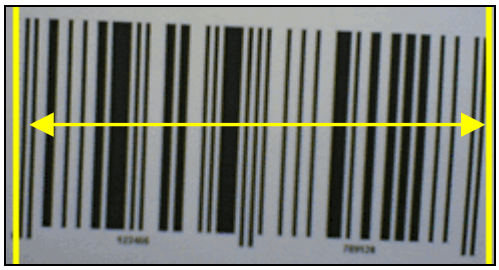
3. Vertikal förvrängning (X) - Siemens. 9 bilder, från 10 grader till 50 graders Y vinkel i steg på 5 grader.

4. Avstånd - Nokia. 6 bilder, från 15 cm till 40 cm mellan kamera och streckkod med steg av 5 cm.

5. Horisontell förvrängning (Y) - Nokia. 12 bilder, från 10 grader till 65 graders Y vinkel i steg på 5 grader.

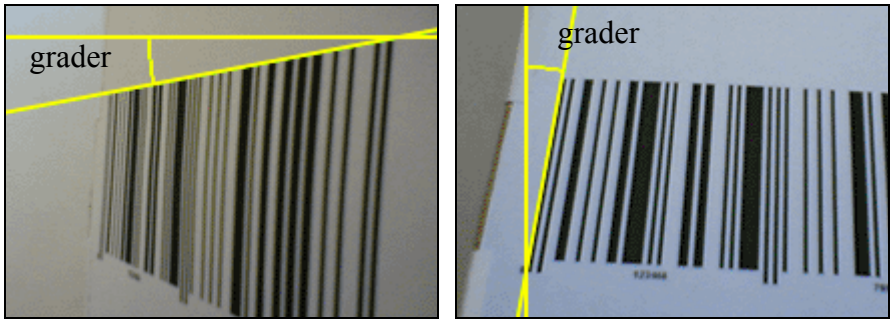
6. Vertikal förvrängning (X) - Nokia. 9 bilder, från 10 grader till 50 graders Y vinkel i steg på 5 grader.

Först jämförs bilder i kategorierna avstånd med motsvarande datorsimulerade bilder i kategorin upplösning. Bilderna ur den datorsimulerade upplösningsgruppen simulerar bilder tagna med olika typer av kameror, dvs. sensorer med olika nivåer av prestanda och pixel kvantitet. Detta motsvaras av bilderna tagna i experimentet med de två olika kamerorna. Bilderna ur den datorsimulerade gruppen upplösning simulerar också bilder på streckkoder tagna på olika längders avstånd mellan kamera och streckkod. Motsvarande situation upprepas i experimentet genom att förändra avståndet mellan kamera och streckkod. Jämförelsen görs genom att titta på antalet pixlar som representerar streckkoden horisontellt (figur 4.5) i de olika bilderna för att se om något samband kan urskiljas. Resultat från jämförelsen visas i bilaga I, del 1.



Figur 4.5 - Pixelantal

Den andra jämförelsen som görs är en undersökning av bilder mellan de datorsimulerade bilderna ur grupperna horisontella perspektivförvrängningar samt vertikala perspektivförvrängningar. I experimentet togs motsvarande bilder genom att vinkla ytan där streckkoden är placerad gentemot kameran. En uppsättning bilder togs genom att vinkla ytan över Y axeln och därefter tas motsvarande bilder genom att vinkla X axeln. Jämförelsen för att undersöka om de datorgenererade bilderna överensstämmer med bilderna tagna i experimentet görs genom att undersöka förvridningsvinkeln med hjälp av att titta på strecken längst ut i streckkoden efter och före förvrängningen. Resultat från jämförelsen visas i bilaga I, del 2.



Figur 4.6 - Vinkel som jämförs

4.2.1 Sammanfattning

Experimentet är tänkt att generera bilder med olika nivåer av störningar. Ställningen som tagits fram gör det möjligt att ändra avstånd, vertikal och horisontell vinkel på det plan där streckkoden sitter. Dessutom gör ställningen det möjligt att byta kamera och streckkod. Vissa begränsningar finns genom att streckkoden och kamera inte kan vara hur nära varandra som helst då streckkoden vid en viss punkt inte kan visas i sin helhet i kameransdisplay eller fotograferingszon. Bilderna från experimentet kan sedan jämföras med motsvarande brusfaktorer från den datorgenererade modellen i kapitel 4.6.

4.3 Teori avkodningsalgoritm

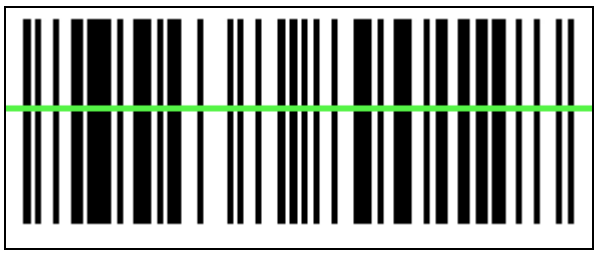
4.3.1 1D

Det finns ett antal sätt på vilket en avkodningsalgoritm för en 1D streckkod skulle kunna arbeta. Angripssätten kan variera med grundförutsättningarna är ändå de samma och det som framförallt skiljer angripningssätten åt är hur de hanterar själva bilden, samt hur algoritmen arbetar för att hitta streckkoden i bilden, alltså hur algoritmen gör om vita respektive svarta fält från en färgbild med grå respektive färgskalor till den ström av ettor och nollor som streckkoden representerar. Från dataströmmen sker översättningen till ASCII tecken, bokstäver eller siffror på mycket likartade sätt.

En skillnad mellan olika avkodare är valet mellan att förvandla den bild som skall genomsökas till en svart-vit representation eller till en bild i gråskala.

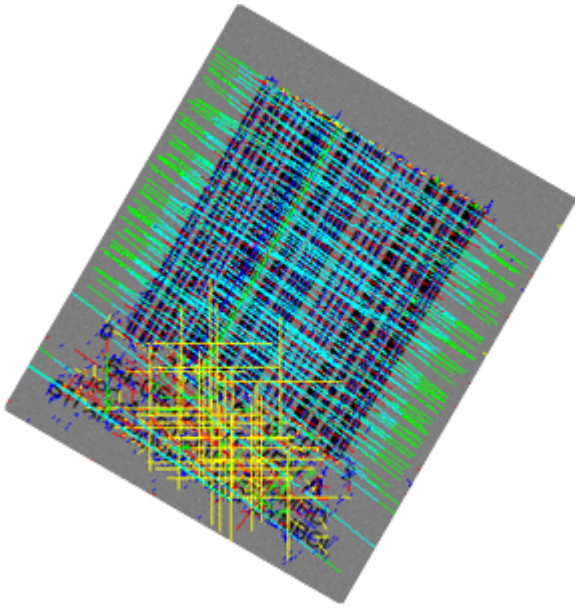
4.3.1.1 Svart-vit

Bilden görs svart-vit med hjälp av en metod som kallas tröskling (se kapitel 4.3.2). En bild görs binär för att få klara, exakta gränser i övergången mellan streck och tomrum. Basmetoden för att avkoda symboler är att projektera ut en tänkt linje över mitten på streckkoden så att linjen korsar alla streck och tomrum (figur 4.7), en typ av simulation av en läspenna eller laseravläsare (se kapitel 3.1).



Figur 4.7 - Projektion av linje över streckkoden

Processen bygger på att läsa varje pixel en för en och spara längden av antalet pixlar som utgör en rad, svart eller vit. Avläsningen ger slutligen även storleken på streckkoden. Med dessa värden som grund kan de svarta linjerna och vita strecken avkodas med hjälp av dess proportionella längd. Avkodningen ger en binär sträng som representerar den kodade värdet. Om ett kort svart streck representeras är 1 pixel lång, representeras den av en etta. Ett brett svart streck representeras av två eller flera ettor, likadant vita streck som däremot representeras av en nolla. Denna binära sträng kan sedan omvandlas beroende på den speciella streckkodens kodningsregler. Mönstret av streck och tomrum matchas mot en databas med kodvärden. Om mönstret finns och checksumman i streckkoden stämmer har algoritmen hittat och gjort en lyckad avkodning. Om inget mönster hittas kan det bero på att den tänkta avkodningslinjen inte är korrekt vinklad jämfört med streckkoden, metodiken kan då gå vidare med att avkodning sker genom att slumpa ut avläsningslinjer i olika vinklar över bilden till dess att den hittar ett mönster av svarta och vita fält som skulle kunna vara en streckkod. Avläsningslinjer ritas ut om och om igen (figur 4.8) ända tills en linje ger en lyckad avkodning.



Figur 4.8 - Vinklade och slumpvisa avläsningslinjer

För att med mer säkerhet kunna avgöra om linjen som lästes av är helt korrekt avkodad kan flera linjer med samma vinkel som föregående lyckade avläsning ritas ut och testas för att på

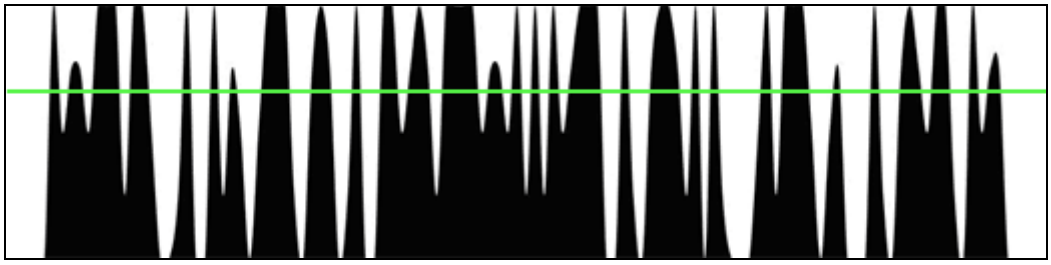
så sätt kontrollera det föregående resultatet. Är någon utav de påföljande avkodningarna överensstämmande ger algoritmen en lyckad avkodning. Metodiken kan även förminskas, så att undersökningen endast sker inom ett angivet område I bilden. Exempelvis kan undersökningen specificeras till området omkring centrum av bilden, och att streckkoden troligtvis endast är riktad inom ett visst gradantal. På detta sätt kan algoritmen spara tid som annars går åt till att söka. Problemet med metoden är att bibehålla rätt skala på streck och tomrum vid förvandlingen från en bild i gråskala/färg till en svart-vit bild. Om förvandlingen, dvs. vilka nivåer av grått skall vara vita och vilka som skall vara svarta, inte blir optimal gör det att storleksproportionerna mellan vita och svara streck förändras och algoritmen kommer då få problem med avkodningen.

En annan lösningsmetod för avkodning [10] [22] är att låta algoritmen söka efter raka linjer, Hough transformen [22] är en vanlig bildbehandlings metod för detta ändamål. Sedan sparas linjernas vinkel gentemot någon specificerad axel. Om det finns många raka linjer inom samma område med samma vinkel, möjligtvis inom ett visst intervall, kan det antas att en 1D streckkod har hittats. Streckkodens längd definieras som området från den första linjen till den sista linjen inom ett intervall av linjer med samma referens vinkel. Höjden är lika med längden på den längsta linjen utav linjerna som hittats. Då området har hittats söks det igenom för avkodning genom metodik beskriven ovan, dvs. en tänkt linje speglas ut över mitten av streckkoden och pixlarna avläses.

4.3.1.2 Gråskalor

En annan metod för avkodning av 1D streckkoder är att bibehålla alla gråa värden som framspeglas vid fotografering av streckkoden. I metodiken förklarad tidigare utnyttjas inte alla nyanser i bilden. Ett sätt att utnyttja hela registret av färger är att plotta varje läst pixel som representerar streckkoden i en graf beroende på pixel nyansen (figur 4.9). Grafen bildar en representation av de svarta och vita strecken i streckkoden med hjälp av all grånyanser. Representationsbilden skannas därefter rad efter rad, varje horisontell nivå som skannas kommer att representera streckkoden med olika binära strängar. Om det ur den binära strängen går att hitta rätt antal kodade tecken utifrån kodningsmallarna samt att den beräknade

checksumman blir korrekt går det anta att en streckkod har hittats och algoritmen kan returnera ett resultat.



Figur 4.9 - Streckkods graf

En mer sofistikerad metod som liknar metodiken ovan och lämpar sig bra för bilder med låg upplösning och kontrast är superresolution [1][2](eng. super-resolution). Superresolution av streckkoder grundar sig på sammanslagning av ett flertal bilder tagna på samma källa för att på så sätt förbättra bildens kvalitet. Med superresolution menas att öka mängden information, i det här fallet med hjälp av de gråa nyanserna i bilden. Faktumet att en 2D bild på en 1D streckkod redan innehåller flertalet representationer av streckkoden i vertikalled ger att det inte behövs flera bilder. En liten rotation av streckkoden gentemot kameran ger dessutom att varje rad blir individuellt olika jämfört med andra rader. Raderna kombineras därefter ihop efter att förskjutningen mellan raderna beräknats för att få en korrekt uppställning. De lågupplösliga bilderna blir omskannad med en högre sampling och ger en mer högupplöslig representation (figur 4.10). För en mer detaljerad beskrivning hänvisas till [1][2].

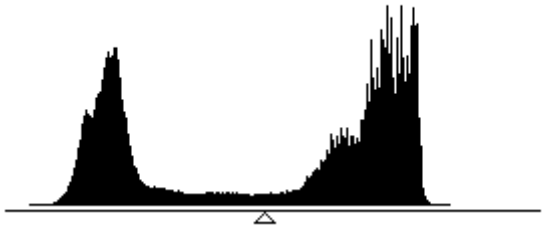
Figur 4.10 - Superresolution [1]

4.3.2 2D

I många bildapplikationer är det ofta fördelaktigt att kunna separera områden i bilden som innehåller objekt som är utav intresse, från bildområden som tillhör bakgrunden [7]. Tröskling (eng: thresholding) [35] ger ett enkelt sätt att utföra denna separation med grund i skillnaden i intensitet eller färg i förgrunden jämfört bakgrunds regioner. Metoden utnyttjas även vid avkodning av streckkoder [12] [6] p.g.a. orsakerna ovan samt för att ta bort färger och gråskalor ur bilden och göra om bilden till en svart-vit avbildning.

För att få en binär bild och för att undertrycka eventuellt brus i bilden trösklas bilden, dvs intensiteter över ett visst värde sätts till ett (svarta pixlar) och de under till noll (vita pixlar), varpå en binär bild fås. Varje pixel i bilden undersöks och ut fås ett diagram med två toppar, en topp för mängden mörka eller svarta pixlar och en topp för vita, ljusa pixlar. Därefter väljs ett värde som vanligtvis befinner sig ungefär mittemellan dessa två toppar, det är detta värde som anger nivån för vilka pixlar som skall konverteras till svarta respektive vita pixlar (se figur 4.11). Om ingångsbilden är av dålig kvalitet samt om trösklingsnivån är illa vald kommer bilden inte innehålla tillräcklig information för att kunna göra en avkodning. Bilden i figur 4.11 visar ett optimalt trösklings förhållande mellan svart och vit i en bild som endast innehåller en 1D eller 2D streckkod. Om streckkoden endast upptar en liten del av bilden påverkar övriga detaljer i bilden trösklingsnivån.

Om bilderna varit av sådan kvalitet att tröskling inte varit tillräckligt kan ett medelvärdesfilter användas. Då ett medelvärdesfilter används minskas bruset till medelnivån av dess näraliggande pixlar. Brus kännetecknas av att det är högfrekvent, dvs. skiljer sig markant i intensitet från sin omgivning. Med hjälp av medelvärdesfilter sänks därför brusets intensitet till ett påtagligt lägre värde. Nackdelen med denna metod är att skarpa kanter i bilden ”smetas ut” på samma sätt som brus gör.



Figur 4.11 - Thresholding

Avkodning av en 2D streckkod sker i 3 olika steg [29].

1. Grafiska nivån - Den grafiska nivån består av den fysiska uppsättningen av svarta och ljusa celler som utgör den 2D symbolen som definierar den kodade dataströmmen.

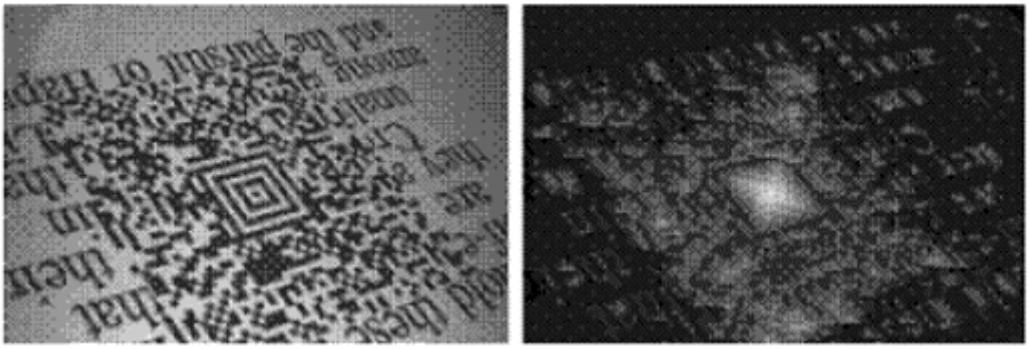
2. Matematiska nivån – Den matematiska nivån representerar felkontrollskodningen som ligger applicerad på dataströmmen både för datasäkerhetsändamål och för att kunna återskapa förlorad data i form av skanningsfel eller skador på symbolen.

3. Meddelandenivån – Meddelandenivån innehåller översättningsregler mellan dataströmmen och det ASCII eller 8-bitars meddelande som symbolen är tänkt att representera.

Dessa tre steg beskrivs nedan samt kan ses och utnyttjas i avkodningen för vilken 2D streckkod som helst då de principiellt är uppbyggda på samma sätt. För att gå in mer i detalj

på hur en avkodningsalgoritm för en 2D streckkod arbetar utnyttjas Aztec koden som exempel och utgångspunkt, grunderna är dock snarlika för andra 2D streckkoder.

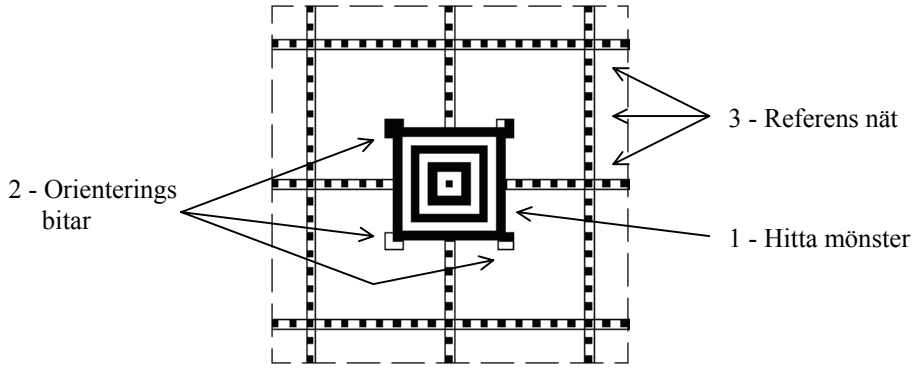
4.3.2.1 Den Grafiska datastrukturen



(a) **(b)**

Figur 4.12 - Hitta mönstret [25]

Hitta mönstret eller målområdet är som namnet föranleder det mönster som avkodningsalgoritmen börjar leta efter. Algoritmen söker det rektangulära pyramidliknande området i mitten av figuren (se figur 4.12a), området ses ”upplyst” figur 4.12b. Målområdet finns alltid i mitten av symbolen och består av 7 lager av kvadrater omväxlande svart och vit. Mönstret är relativt enkelt att hitta tack vare mönstrets skarpa kanter och genom att utnyttja det enklaste av förhållanden i bild analys - pixel anknytning. Mönstret hittas genom att undersöka närliggande pixlar med varandra.



Figur 4.13 - Hitta, rotation och referens mönster [29]

Så fort som målområdet har påträffats, skannas varje ring för att (a) fastställa centrum och för att (b) hitta positionerna för de fyra 3-bit kluster av orienterings bitar.

Med hjälp av riktning och distans mellan närliggande hörn uppskattas skala och riktning för två centrum linjer (se figur 4.14a). Men beroende på perspektiv och liknande störningar kan de uppskattade linjerna vara av felaktig skala och/eller ha en felaktig vinkel. Utåt med start i centrum projekteras cellernas eller modulerna position, små korrektioner görs så att de angränsande modulerna centreras. Detta utförs fram till att det första lagret utanför målområdet påträffats. I detta lager hittas de 4 orienteringsklustren tillsammans med det första dataområdet. Orienteringsbitarna läses av för att bestämma symbolens rotation i nord och öst led, övrig information i mode lagret läses, avkodas och rättas därefter. Modeinformationen anger antalet lager som symbolen består av och hur många data ord som resterande lager innehåller (se figur 4.14b).

(a) **(b)**

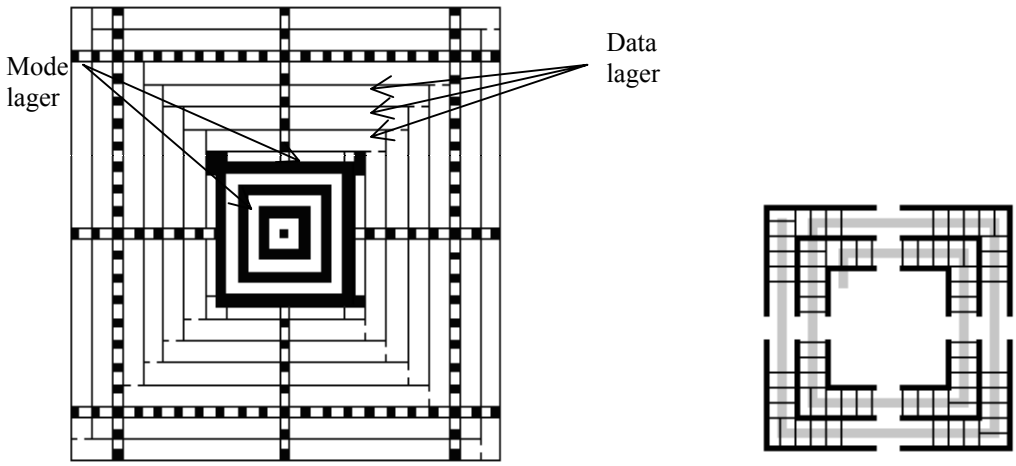
Figur 4.14 - Centrum och rutnätet [25]

Om symbolen består av max 4 datalager används metoden ovan att genom små korrektioner centrera angränsande moduler tills hela symbolen avlästs. Om symbolen är större än 4 datalager utnyttjas symbolens referens nät (se figur 4.13 och 4.15a) för att med dess hjälp specificera samplingspunkterna för alla data positioner (se figur 4.15b), referensnätet sprider ut sig horisontellt och vertikalt på var 16 rad utåt i symbolen och används för att upptäcka och parera olika förvrängningar i symbolen, som t.ex. barrel eller perspektiv förvrängningar. Båda metoder ger en karta över alla databitar.

(a) **(b)**

Figur 4.15 - Referensnät och bitkarta [25]

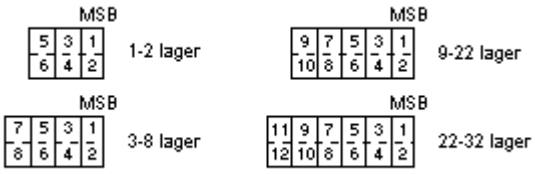
Varje datalager består av 2 rader av celler och bitarna i de två raderna formar speciella kodord. Bitarna som ligger parvis, genom att två rader celler utgör ett datalager, sprider ut sig likt ett dominomönster (se figur 4.16) medsols runt centrum. En bit är 1x stor, två bitar är 1x bred och 2x hög.



Figur 4.16 - Lagerstruktur och dominomönster [29]

Den välstrukturerade domino layouten har dock 2 onaturligheter.

a) Kodorden i layouten varierar beroende på symbolens storlek. Kodorden består av symboler med 1-2 lager av 6 bitar, i symboler med 3-8 lager av 8 bitar, 9-22 lager av 10 bitar och i symboler med 23-32 lager av 12 bitar.



Figur 4.17 - Kodord struktur [29]

b) Avkodningsfel och felaktigheter i symbolen antas ske i en större utsträckning utmed och ut mot symbolens kanter, därför är de överensstämmande Reed-Solomon kontrollbitarna lagda i omvändordning, dvs. det första kodordet i lager 1 är det sista kontrollordet.

4.3.2.2 Den matematiska nivån

Modelagret består av 40 bitar, de 5 första bitarna som läses av anger antalet lager symbolen består av, egentligen antalet datalager minus ett i binär form (ex ”00001” anger att symbolen har 2 datalager). Nästa 11 bitar anger antalet kodord i symbolen, egentligen antalet kodord minus ett. Övriga 24 bitar är Reed-Solomon kontrollbitar. De 16 första bitarna är uppdelade i ord på vardera 4-bitar. Därefter följer 6 kontrollord skapade med Reed-Solomon kodning över Galois Field, GF(16) baserat på en prim modulation polynom av $x^4 + x^1 + 1$ (se kapitel 3.3 om Reed-Solomon koder). Med hjälp av kontrollorden kan informationen i de första 16 bitarna rättas till en viss grad. Det första kontrollordet används för feldetektering och felkorrigering av det sista 4-bitar ordet som representerar antalet datalager + antalet dataord, det andra kontrollordet är kontrollsumma för det näst sista 4-bitars ordet.

Datalagren kodas liksom modelagret av Reed-Solomon. Kontrollkoden skapas i olika nivåer av GF, nivån beror på symbolens storlek. Om symbolen består av 1 till 2 lager används kodord på 6-bitar över GF(64) och polynomet $x^6 + x^1 + 1$ [29]. Lager 3-8 har kodord på 8-bitar över GF(256) med polynomet $x^8 + x^5 + x^3 + x^2 + 1$ [29]. Datalager 9-22 utnyttjar 10-bitars kodord med GF(1024) och $x^{10} + x^3 + 1$ [29], i datalager 23-32 används 12-bitar över GF(4096) och polynomet $x^{12} + x^6 + x^5 + x^3 + 1$ [29]. Informationen i datalagren felkontrolleras och informationen återskapas med hjälp av Reed-Solomons kontrollord i de fall där det behövs. Kontrollorden ligger liksom i modelagret i omvänd ordning.

4.3.2.3 Meddelandenivån

Meddelandeavkodningen sker i två steg. Först förvandlas kodorden, med angiven storlek beroende på antalet datalager i symbolen, till en bitström. Kodorden är rakt kodade i sekvenser av 6, 8, 10 eller 12 bitar, med den mest signifikanta biten först. Två undantag finns dock, ett kodord kan inte bestå av enbart nollor utan i dessa fall har en etta skjutits in i kodordets minst signifikanta bit och efterföljande nolla i nästa kodord hör så att säga ihop med föregående kodord. Likadant fungerar det för motsvarande förlopp med enbart ettor. Det finns med andra ord inget speciellt förhållande mellan tecken och byte gränser i det avkodade meddelandet och själva kodorden.

I steg två omvandlas eller översätts bitströmmen till ASCII eller 8-bit tecken. Bitströmmen omvandlas till tecken med hjälp av tabellen i bilaga G. Bitströmmen börjar alltid med att avkodas i ”Upper mode” och ur strömmen tas 5 eller 4 bitar, ibland även 8, för att avkodas gentemot tabellen, bitströmmen kan även hoppa till andra lägen (modes) för att kunna komma åt alla tecken i tabellen. Upper, Lower, Mixed, och Punct mode kodar alla tecken med 5 bitar, i Digit mode används 4 bitar. Binary Shift är ett specialläge som använder 8-bitar och kan användas för vissa speciella ändamål.

4.4 Avkodningsapplikationer

Det finns flera sätt på vilket en avkodningsapplikation för 1D eller 2D streckkoder kan arbeta. Varje lösning eller angripningssätt för att skapa en fungerande tillämpning har varje, sina fördelar respektive nackdelar. Lösningarna kan delas in i 3 olika typer av förhållanden.

1. Klientlösning.
2. Serverlösning
3. Klient-server lösning.

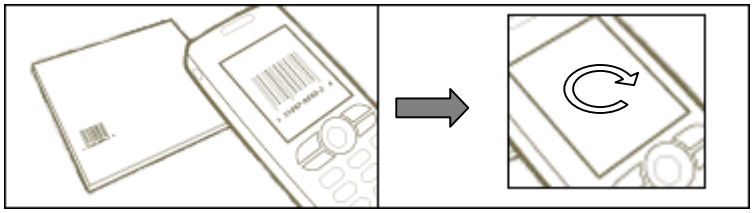
4.4.1 Klientlösning

Med en mobilbaserad lösning menas att skapa en applikation som körs direkt i mobiltelefonen. Det betyder att hela avkodningsförloppet sker direkt i mobiltelefonen, från fotografering till avkodning, igenkänning och resultatvisning. Datakommunikation med utomstående hård- och mjukvara begränsas och kan t.o.m. uteslutas.

En mobilbaserad lösning blir dock helt beroende på mobiltelefonens resurser i avseende på processorkapacitet, minnesutrymme, operativsystem m.m. Avkodningsalgoritmen har i en sådan miljö klara begränsningar. Mönsterigenkänningen kan t.ex. inte lösas genom en avkodningsalgoritm som i sin metodik kräver en stor mängd processorkraft och tid. Algoritmen får då begränsas i avseende på t.ex. eventuella bildkorrigeringar eller andra tidskrävande operationer. Avkodningsmjukvaran har i och med detta inte lika stora

möjligheter att göra en korrekt avkodning på bilder som är av mindre bra kvalitet. Andra nackdelar eller begränsningar med programvara som körs direkt i mobilen är de operativa resurserna, en mobiltelefon har vanligtvis t.ex. inte stöd för hela Java J2EE (Java 2 Enterprise Edition) utan bara för en mindre kompaktare version J2ME (Java 2 Micro Edition), speciellt anpassad för mobila enheter. Ett program för bildigenkänning som används i en mobiltelefon måste också installeras i mobiltelefonen för att kunna köras. Att lägga till tredjepartsapplikationer i en mobil är inte så vanligt för den normala mobiltelefonanvändaren. Saken förenklas dessutom inte av att installationsförloppet ofta sker på olika sätt i olika mobiltelefoner även om ett mer och mer standardiserat förlopp har börjat införas.

Fördelar med denna metod finns dock. En applikation som finns direkt i mobiltelefonen har bl.a. större möjligheter att handleda användaren. Applikationen kan helt enkelt hjälpa till att guida det mobila handhavandet för att med hjälp av en interaktivitet få användaren av mobilkameran att ta en bättre bild och därigenom minska behovet av en resurskrävande avkodningsalgoritm. Andra fördelar med en mobilbaserad lösning är att ingen tid går åt till kommunikation med utomstående parter, t.ex. servrar. Avkodningsförloppet av bilden med den 1D eller 2D streckkoden begränsas då inte på nått sätt, rent tidsmässigt, utav fördröjningar i de mobila näten som används vid överföringen av bilden eller utav de mobila nätens överföringshastigheter.



Figur 4.18 - Mobillösning

4.4.2 Serverlösning

En tillämpning som körs på en server utnyttjar det enkla faktum att mobiltelefonerna inte har tillgång till lika stor processor kraft och minne som en server. Fördelen är att bilden på den 1D eller 2D streckkod som skall avkodas inte behöver ha lika god kvalitet.

Avkodningsalgoritmen i en server har större resurser för att göra förbättringar på bilden med hjälp av bildbehandling och filter om en initial avkodning skulle misslyckas. Bildigenkänningen i en server sker helt enkelt avsevärt mycket snabbare varvid avkodningsförsöken kan vara många fler under samma tidsrymd som motsvarande avkodning i ett mobiltsystem som utnyttjar en mindre avancerad avkodningslogik. Språk och programvara är heller inga direkta hinder som t.ex. begränsar applikationen till Java J2ME.

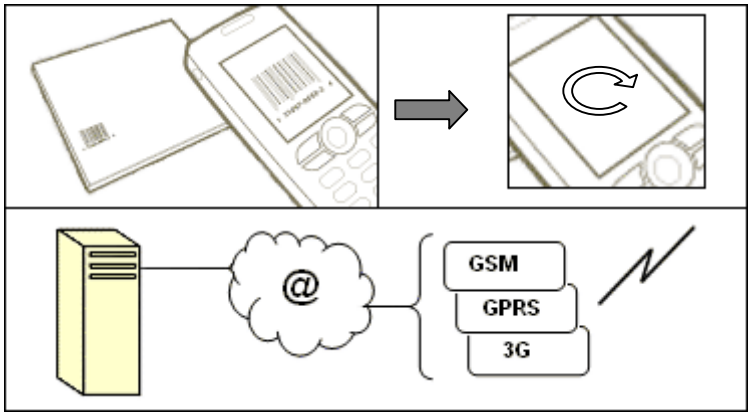
Nackdelarna är kanske inte helt oväntat att i och med att det inte finns någon aktiv applikation som snurrar i mobiltelefonen finns det heller ingen möjlighet till en aktiv guidning av användaren i eventuella försök att hjälpa användaren att vinkla eller zooma in den 1D eller 2D streckkoden, för att på så sätt få bättre bilder och en bättre grundbild för att göra avkodning på. Andra problem är överföringen av själva fotografiet som tas av den speciella streckkoden. Är nätet högt belastat kan det ta lång tid att skicka iväg bilden, och även för att få tillbaka ett svar. Är dessutom bilden av sådan kvalitet att avkodningen inte går igenom så måste användaren återupprepa processen. Allt detta leder till långa väntetider, vilket allt som oftast, oberoende av situation är ett irritationsmoment.

Figur 4.19 - Serverlösning

4.4.3 Kombinerad lösning

Den tredje typen av lösning är en medelväg som använder sig av både en applikation direkt i den mobila enheten samt en central server. Tanken är att utnyttja en mobilapplikation för att initialt leda användaren till att ta ett bra kort på den 1D eller 2D streckkoden. Därefter skickas bilden iväg via det mobila nätet till en server som utför den lite mer krävande bild-igenkänningen. Tack vare guidningen i början blir avkodningen enklare att hantera då

förvrängningarna i bilden inte kan bli lika extrema. Avkodningsprocenten ökar och omtagning och omsändning av bilder minskar. Kommunikationen över de mobila nätet kvarstår dock i vilka fördröjningar vid hög belastning inte går att förhindra. Installation och programspråks kompabilitet kvarstår här likt lösning 1.



Figur 4.20 - Kombineradlösning

4.4.4 Sammanfattning

Alla dessa lösningar har sina för- respektive nackdelar. Affärstillämpningen och omständigheter avgör ofta vilken lösningen som är bäst, det optimala vore är att realisera alla lösningar och därefter låta användaren välja vilken metod som är mest användarvänligt och bäst förenligt med hans/hennes speciella krav, behov och möjligheter. Utvecklingen av applikationer i denna D-uppsats med hjälp av open-source algoritmer är helt och hållet en serverbaserad lösning.

4.5 Dagens teknologi – mobiltelefoner

Här presenteras ett litet urval ur dagens marknad av mobiltelefoner med digitalkameror. Dessa kameror används sedan även i det experiment som försöker anlägga ett samband mellan de bilder som genereras av metoden beskriven ovan samt bilder tagna i verkliga förhållanden.

4.5.1 Siemens SX1

Siemens SX1 släpptes i fjärde kvartalet 2003, den är en trippelband GSM 900/1800/1900 mobiltelefon med Symbian OS som operativsystem och är kompatibel att köra Java program enligt Java 2 Micro Edition, MIDP 1.0. Bluetooth finns inbyggt, likaså IrDA. Telefonen har 1Mb internt minne och ett externt minnesplats av MMC-typ som används för eventuella uppgraderingar av minnet, 16 MB medföljer. En USB-sladd medföljer i paketet och gör det enkelt att föra över bilder/låtar/filer. Mp3-spelarfunktion samt radio finns inbyggt och är förhållandevis enkel att använda. Telefonen är utrustad med en TFT färgskärm på 176x220 pixlar som visar 65 000 färger. Digitalkameran är integrerad i telefonen, kameran har en 0.3 megapixel CMOS sensor som klarar av att ta bilder i VGA-upplösning 640x480 pixlar och har 8x digital zoom. Den har 24-bitars färgdjup, ljuskänslighet på > 30 Lux och fokus från 30 cm till oändligheten. SX1 har GPRS kapacitet med möjlighet till WAP, MMS och e-post. Telefonen har dimensionerna 109 x 56 x 19 mm, en vikt på 110 g och standby på 400 tim. All teknisk data är tagen från Siemens hemsida [46].

Figur 4.21 - Siemens SX1

4.5.2 Nokia 7610

Nokia 7610 släpptes under det tredje kvartalet 2004. 7610 är en mobiltelefon som bygger på operativsystemet Symbian OS, den har trippelband kapacitet över GSM 900/1800/1900. Java finns i telefonen och det går att köra Java MIDP 2.0-program. Nokian har bluetooth och möjlighet till USB-anslutning, den saknar dock IrDA. Det interna minnet ligger på 8 MB med en extern minnesexpansion för MMC-kort i krympt format där ett 64 MB stort kort ingår. Mp3-spelarefunktionen saknar stereoljud och någon radio finns inte alls. Nokia 7610 har en högupplöst kamera med 4x digital zoom och klarar att ta bilder i upplösningen 1152x864, med en 1.3 megapixel CMOS sensor. Kameran har självutlösare och specialläge för natt. Skärmen är en ljusaktiv TFT-färgdisplay som är 176x208 pixlar stor och klarar av att visa 65 536 färger. Mobiltelefonen är klarar av att skicka data över GPRS i en hastighet på upp till 40,3 kbit/s, den har WAP funktion tillsammans med MMS, e-post och chatt. De yttre dimensionerna ligger på 108.6 x 53 x 18.7 mm. Passningstiden ligger på 240 tim och vikten är 118 g. Alla specifikationer har tagits från Nokias hemsida [42].

Figur 4.22 - Nokia 7610

4.5.3 Sony Ericsson s700i

S700i släpptes under det fjärde kvartalet 2004 den har trippelkapacitet över GSM 900/1800/1900. Java och Java MIDP 2.0 program går att köra i mobilen. Bluetooth, IrDA respektive USB-anslutning finns alla tillgängliga. Internminnet ligger på 32 MB och ger tillsammans med ett Memorystick Duo på 32MB ett generöst lagringsutrymme. Mobilen har FM radio som automatiskt söker upp stationer och Mp3-spelarfunktion. Telefonen domineras annars av den stora skärmen på 1,9 tum och som upptar hela framsidan, telefonen är därför

utvikningsbar för att på så sätt komma åt knappsatsen. Skärmen har en 240x320 bildpunkters LCD färgdisplay med bakgrundsbelysning och som visar 262K färger. S700i är vidare ett steg fram mot att helt integrera kamera och mobil. Hela baksidan av telefonen är utformad som en digitalkamera med linsskydd. Sony Ericsson har en 1,3 megapixlars CCD type sensor integrerad digitalkamera med en upplösning på 1280x960 pixlar, den har dessutom 8x digital zoom och en LED blixt som ger effekt upp till någon meter ifrån kameran. Inställningar som vitbalans, mörkerläge och självutlösare finns också. S700i har stöd för GPRS, HSCSD, WAP och e-post. Telefonen har 300 h pasningstid. Vikten ligger på 137 g och storleken i hopfällt läge är 107.5 x 49 x 24.5 mm. Tekniska specifikationer är hämtade från Sony-Ericssons hemsida [47].

Figur 4.23 - Sony Ericsson s700i

Tabell 4.1 - Mobiltelefonkameror

Mobil	**Kamera**	**Upplösning**
Siemens SX1	CMOS	640 x 480 pixlar
Nokia 7610	CMOS	1280 x 960 pixlar
Ericsson s700i	CCD	1280 x 960 pixlar

4.6 Metoden

Metoden är tänkt att göra det enklare att avgöra vilken algoritm i valet av två eller flera som har den bästa 1D eller 2D streckkodsavkodningen, dvs. vilken avkodningsalgoritm som har störst procentuella chans att avkoda en streckkod i en bild beroende på olika nivåer av bildstörningar. Metoden skall vara enkel att genomföra och helst ske som en automatisk process där svaret efter genomgången testning beskriver hur höga nivåer störningar bilden av streckkoden kan tåla för en specifik algoritm och ända få ut en korrekt avkodning.
Metodiken för jämförelsen är tänkt att vara en steg för steg metod för att minimera behovet av att i ett inledningsskede slippa det tidsödande arbete som krävs för att gå igenom koden för en specifik algoritm och förstå dess metodik och arbetsgång, i princip en ”black-box” testning. En sådan genomgång försvåras även ofta av faktorer som kodstruktur och mängden kommentarer i koden gjorda av i programmeraren.

Grundförutsättningen och indata i en bildigenkänningsapplikation är självklart en bild. En bild är en representation utav verkligheten och olika faktorer kan påverka avbildningen och förändra representationen i olika stor grad. Faktorer som kan påverka bilden av streckkoden är kamerafaktorer, miljöfaktorer och tryck-kod faktorer som finns beskrivna i kapitel 3. Metoden i denna D-uppsats grundar sig på alla de olika bruseffekter som kan skada, förändra och slutligen påverka avkodningen av en streckkod i en bild. Bruseffekterna som tagits upp i kapitel 3 är därför bas för att generera ett antal testbilder med olika nivåer av störningar i olika kategorier som sedan utnyttjas som referensbilder i en undersökning av algoritmens robusthet gentemot avkodning av streckkoden i bilden.

Alla brusfaktorer som tagits upp i kapitel 3 är effekter som kan störa avläsningen av en streckkod. För att undersöka en algoritms stryktålighet skapas därför utifrån bruseffekterna ett antal bilder som representerar bruseffekterna. Från att utnyttja noll procentuell förändring utav effekterna, dvs. en original bild, och sedan stegvis procentuell förändra originalbilden med en brusfaktor. På detta sätt skapas ett antal referensbilder med små brusförändringar. Därefter kan en avkodningsalgoritm testas genom att försöka avkoda varje bild ur referensgruppen och spara resultatet av vilka bilder som fått en korrekt avkodning och på så sätt skapa en tabell som kan jämföras gentemot nästa avkodningsalgoritm som testas och därefter går det att visa vilken algoritm som genomfört flest lyckade avkodningar. Vidare

finns det även möjlighet att se om en viss algoritm klarar sig bättre mot en speciell typ av brus än någon annan. Det går även i testet att få en uppfattning om en specifik 1D eller 2D streckkod klarar sig bättre mot vissa specifika störningar än någon annan. En stor uppsättning bilder med små nivåförändringar i brus gör det möjligt att enklare att särskilja på två jämbördiga algoritmer, vidare går det även att skapa bilder som kombinerar två eller flera bruseffekter i procentuella steg för att på så sätt testa ytterligare användarfall. I denna rapport används ett urval av nivåer och störningar, för en än mer exakt undersökning går det att skapa fler bilder som representerar fler nivåer av eventuella störningar i bilden.

Bilder skapas i dator för att exakt kunna veta den procentuella skillnaden i förvrängning gentemot original bilden. Referens bilderna skapas i Photoshop 7.0 [26]. De referensbilder som genereras är sådana bilder som simulerar de olika typer av effekter som påverkar en mobiltelefonkameras bildkvalitet och effekter som kan uppkomma då en 1D eller 2D streckkod fotograferas. Effekterna simuleras i Photoshop genom att addera olika filter på originalbilden och beroende på angiven nivå blir bruseffekterna mer eller mindre stora.

Photoshop 7.0 är ett professionellt program för bildredigering. Photoshop vänder sig till designers och grafiker för att skapa avancerade bilder för tryck, webben, trådlösa enheter och andra medier. Verktyget har en omfattande uppsättning av retuscherings-, målar- och rit-funktionalitet för att kunna utföra olika bildredigeringsaktiviteter i en mångsidig miljö. I Photoshop finns bl.a. verktyg som lager, färgkorrigering, mätegenskaper, skärpekontroller, filter, omformningsverktyg, funktionsmakron samt markerings och editeringskontroller.

Lager: Med hjälp av lager går det arbeta på ett element utan att påverka andra. Det går att låsa lager för att förhindra oavsiktliga ändringar, gömma dem för att få en bättre visning av det element du arbetar med och länka lager och flytta dem som en grupp.
Färgkorrigering: Används för att justera färger i en bild. t ex kommandot för automatisk färg analyserar bilden för att göra snabba och pålitliga färgjusteringar och ändra bilden permanent. Verktyget erbjuder även möjlighet att redigera färger och göra tonjusteringar.
Skärpekontroller: Ett av skärpeverktygen i Photoshop är filtret för oskarp mask. Med hjälp av oskarp mask går det att skala, rotera, korrigera färger eller utföra något annat som påverkar pixelstrukturen i en bild för att få en bra skärpa i bilden.

Filter: I Photoshop ingår mer än 95 specialeffektfilter, allt från konstnärliga effekter till rörelseoskärpa, ljuseffekter och förvrängningar. Plugin-filter från andra programvaruföretag än Adobe går också att installera och använda. När plugin-filtren har installerats visas de längst ned på Filter-menyn och fungerar på samma sätt som inbyggda filter.
Omformningsverktyg: Skala, rotera, förvräng eller skeva bilder på ett enkelt sätt. Med hjälp av 3D-omvandlingsfiltret går det att simulera tredimensionella effekter som t ex krukor och lådor. Med kommandot gör flytande erbjuder att interaktivt skjuta, dra, snörpa ihop eller blåsa upp en bild.
Funktionsmakron: Automatisera vanliga uppgifter, som t ex gruppbearbetning, genom att spela in stegen som ett funktionsmakro. Gör funktionsmakrot till en droplet för jobb som sker dagligen, ofta eller som är tidskrävande. Spara droplet-filen till tex. skrivbordet, sedan går det att enkelt använda funktionsmakrot på olika filer eller mappar med bilder genom att dra och släppa. Detta är bara några exempel på vad som går att göra i Photoshop. För en mer ingående förklaring se dokumentation för Photoshop [26].

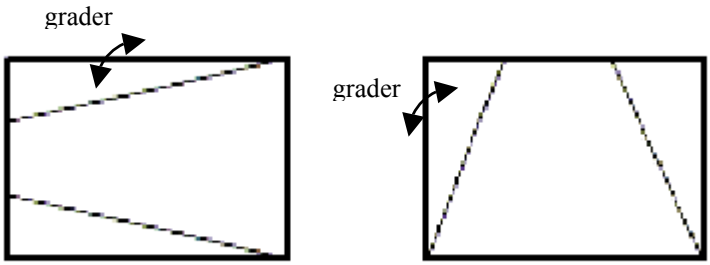
De effekter som simuleras i Photoshop är:

1. Olika upplösningsnivåer, dvs. mobilkamerans upplösning tillsammans med själva avståndet mellan streckkod och kameran. Båda dessa faktorer påverkar hur många pixlar som kan representera den 1D eller 2D streckkoden. Desto fler pixlar som representerar streckkoden desto större är chansen att kunna särskilja två olika fält. De upplösningsnivåer som simuleras börjar i 640 x 480 pixlar och rör sig stegvis neråt till 56x42 pixlar.

2. Filformat. De olika upplösningsnivåerna sparas i två olika filformat, JPG och GIF för att kontrollera komprimeringsskillnader. Påpekas bör också att komprimeringsnivån som används för alla JPG filer är satt till den minsta möjliga i Photoshop, vilket medför att så stora filer som möjligt används.

3. Perspektivförvrängningar simuleras. Både förvrängningar rakt horisontellt och rakt vertikalt simuleras (se figur 3.24). De olika nivåerna mäts i antal grader i höjd och våg

ledes. För horisontella förvrängningar skapas bilder från 4 graders vridning upp till 26 grader. Vertikala förvrängningar går från 4 till 18 grader.



Figur 4.24 - Vertikal och horisontal vridning

4. Rörelseoskärpa simuleras. Detta är effekter som uppstår vid skakig hantering av mobilkameran och vid lång slutartid. Effekterna resulterar i något som liknar dubbelexponering och ger en otydlighet i bilden i avseende på konturer och gränser. Bilder genereras med 1 pixel till 18 pixlar förskjutning, förskjutningen sker både horisontellt och vertikalt över bilden.

5. Kontraststörningar används som en kategori. Kontrast simuleras genom att förminska skillnaden mellan svart och vit, de två nivåerna närmar sig varandra. Nivåerna ändras från 10 % till 80 %.

6. Brus är den femte kategorin som testas. Bruset som används i denna simulation är gaussiskt monokromt brus, med gaussiskt brus menas att bruset är intensitets fördelat och monokromt att det endast består av gråskalor och inte hela färgskalan. Filtret som används för att simulera brus är slumpvis genererat, dvs. bruset sprids olika varje gång filtret läggs över samma bild med samma stryka. Därför för att få samma brusbild på alla bilder med angiven nivå skapas först en brusbild för varje brusnivå. Varje nivå sparas som en separat bild. De sparade brusbilderna garanterar att samma brusbild läggs på den angivna originalbilden, vilket garanterar likvärdiga testförhållanden. Brus läggs på originalbilden i nivåerna 5 % till 15 %.

7. Barrel och Pincushion effekter simuleras slutligen. Dessa effekterna skapas med hjälp av ett plugin-filter i Photoshop. Plugin filtret kallas för PanoTools [44] och går att ladda ner utan avgift via Internetreferensen. PanoTools består av flera verktyg, men det filter som används för att simulera Barrel och Pincushion effekter kallas för Correct. I Correct simuleras Barrel och Pincushion genom att utnyttja radiell förflyttning av pixlar, förflyttningen definieras av ett tredje gradens polynom och kan utföras individuell för varje färg, röd, grön och blå. Här används samma koefficienter oberoende av färg. Korrektions formeln anger relationen mellan avståndet från centrum av ursprungsbilden (rsrc) till motsvarande avstånd i bilden med förvrängningar (rdest).

   r_src = (a * r_dest^3 + b * r_dest^2 + c * r_dest + d) * r_dest

   Parametern d anger den linjära skalningen av bilden. Med d=1 och a=b=c=0 gör att bilden blir oförändrad. Andra värden på d skalar bilden med angiven mängd. För att förhindra att skala bilden bör a + b + c + d = 1. a, b och c förvränger bilden. Positiva värden flyttar punkter från centrum, vilket motsvarar Barrel effekter. Negativa värden flyttar in punkter mot centrum och det simulerar Pincushion-effekter. Förändringar som utförs enbart med a koefficienten påverkar bara pixlar längs ut i bilden, medan värden på b ger en mer likformig förvrängning [43]. Förvrängningar skapas från 10 % till 80 %.

Processen som skapar alla dessa bilder automatiseras genom att skapa ”droplets” i Photoshop. En droplet är en liten tillämpning för att utföra ett funktionsmakro på en eller flera bilder som dras till droplet-ikonen. Grunden för en droplet utgörs av ett funktionsmakro. Ett funktionsmakro är en serie med kommandon som tillämpas på en enstaka fil eller på en grupp med filer. Det går att skapa ett funktionsmakro som omfattar de flesta kommando i Photoshop, ex. ändra bildstorlek på en bild, lägga på en filter effekt på bilden, ändra bildens detaljskärpa, eller helt enkelt genom ett fil-meny kommando spara filen med önskat format [26]. För att exekvera en droplet, som i de här fallen genererar x antal bilder dras den tilltänkta originalbilden med musmarkören och släpps på dropletfilen.

De droplets som skapats för automatisk generering av testbilder följer i en lista nedan.

- 1. brus.exe - 11 bilder genereras med brus från 5 % till 15 %, i stegökningar på 1.

- 2. distortion_barrel.exe - 15 bilder genereras med barrel förvrängningar från 10 % till 80 %, i stegökningar på 5. I denna kategori visas en dialogruta mellan varje filteranvändning, vid dialog ruta ett måste filtret ställas in korrekt. Detta görs genom att ladda set-filen ”PanaTOOLS_Barrel” som finns i katalogen filter eller genom att manuellt ställa in följande värden under options: 0 för parameter c, 0 för b, 0.05 för a och 0.94 för d.

- 3. distortion_pincusion.exe - 15 bilder genereras med pincushion förvrängningar från 10 % till 80 %, i stegökningar på 5. Motsvarande inställningar gör här i dialogruta ett, ladda filen ”PanaTOOLS_Pincushion” som finns i katalogen filter eller knappa in följande värden: 0 för parameter c, 0.01 för b, 0.04 för c och 1.06 för d.

- 4. kontrast.exe - 15 bilder genereras med kontrast filter från 10 % försämring till 80 % försämring i stegökningar på 5.

- 5. perspektiv_horisontell.exe – 12 bilder genereras där förvrängningsvinkel varieras från 4 till 26 grader i steg av 2.

- 6. perspektiv_vertikal.exe - 8 bilder genereras där förvrängningsvinkel varieras från 4 till 18 grader i steg av 2.

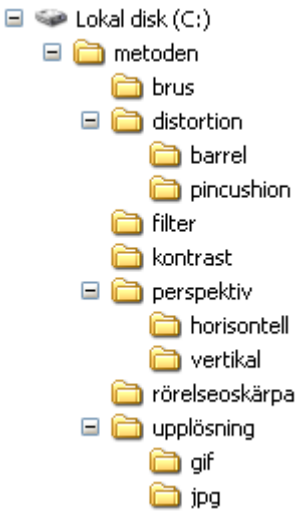
- 7. rörelseoskärpa.exe - 13 bilder genereras med oskärpa från 1 till 8 pixlar i steg på 1 och från 10 till 22 pixlar i steg på 2.

- 8. upplösning_gif.exe - 8 bilder genereras och sparas som gif. Upplösningarna som används är 640x480, 480x360, 360x270, 270x203, 203x152, 152x114, 114x56 och 56x42.

- 9. upplösning_jpg.exe - 8 bilder genereras och sparas som jpg. Upplösningarna som används är 640x480, 480x360, 360x270, 270x203, 203x152, 152x114, 114x56 och 56x42.

Men innan bilderna genereras måste en katalogstruktur skapas då mappar inte går att generera i Photshops funktionsmakron. Katalogstrukturen som behöver skapas och som även fungerar som kategorier är definierade i figur 4.25 och finns även som en bat-fil i bilaga E.

För att exekvera bat-filen öppna en DOS kommando prompt. I Windows XP välj Start - kör... I öppna fältet skriva ”cmd” och öppna för att starta DOS. Kör bat-filen genom att skriva dess namn och därefter ’Enter’. Följande katalogstruktur skapas då:



Figur 4.25 - Katalogstruktur för automatisk bildgenerering

För att generare testbilderna dras originalbilden, som är grunden för övriga bilder, och släpps över den droplet fil som representerar den kategori av bilder som skall skapas. Notera att Photoshop måste vara installerat på dator för att droplet filerna skall fungera. Bilderna skapas och sparas i sina respektive kategorier i katalogstrukturen som genererades med hjälp av bat-filen eller för hand.

Efter att bilderna har genererats går det att genomföra test på avkodningsalgoritmen. Testet utförs genom att köra avkodningsalgoritmen på varje bild i varje kategori i strukturen. Ut ur testet fås en tabell som listar varje bild i varje kategori, vidare listas avkodningsresultatet från varje bild. Testet listar om avkodningsalgoritmen har hittat någon kod, samt om resultatet är korrekt avkodat.

Eftersom varje algoritm som testas ser olika ut, anrop ser olika ut, resultat ut ur algoritmen hämtas på olika sätt och i olika format, initiering av algoritmerna ser olika ut m.m. På grund utav dessa olikheter har en grundstruktur för ett program skapats för avkodningsalgoritmer i Java, koden finns i bilaga F. I anropet till programmet skall 2 argument anges. Argument ett är sökvägen till den katalogen som utgör stam för katalogträdet som innehåller alla testbilder som beskrivits ovan. Vanligtvis är sökvägen definierad som ”c:\metoden”. Argument två i anropet anger namnet på den text fil i vilket resultatet ut från testkörningen sparas. Filen sparas i sökvägen som anges av argument ett, om filen redan finns skrivs den över. I koden för programmet anges var och hur koden lämpligen bör ändras om en ny algoritm är tänkt att testas.

Efter eventuella ändringar skall alla redigerade filer sparas och kompileras i lämplig kompilator. Notera att java måste vara installerat på datorn. Se http://java.sun.com/ för mer information om att hämta hem Java utvecklings kitt, installera och konfigurera Javas SDK (Software Developing Kit), kompilera Java-filer m.m. Vid exekveringen av testprogrammet som avkodar alla bilder i alla kategorier ses en progressbar ticka över skärmen. Efter att progressbaren stannat visar applikationen antalet kategorier samt antalet bilder som testats. När progressbaren stannat har alla resultat blivit lagrade i resultat filen och filen kan öppnas för eventuell granskning, se exempel nedan. För en mer utförlig genomgång av metodens tillvägagångssätt se bilaga N - lathunden.

Ex. utdrag från resultat filen.

Korrekt avkodning enligt algoritmen har skett då ”checksum = true” samt ”resultat = 123456789128”.

C:\metoden\rörelseoskärpa

filnamn	**checksum**	**resultat**
01 pix.jpg	true	123456789128
02 pix.jpg	true	123456789128
03 pix.jpg	true	123456789128
04 pix.jpg	true	123456789128
05 pix.jpg	true	123456789128
06 pix.jpg	true	123456789128
07 pix.jpg	true	123456789128
08 pix.jpg	true	123456789128
10 pix.jpg	false	0
12 pix.jpg	false	0
14 pix.jpg	false	0
16 pix.jpg	false	0
18 pix.jpg	false	0
20 pix.jpg	false	0
22 pix.jpg	false	0

C:\metoden\upplösning\jpg

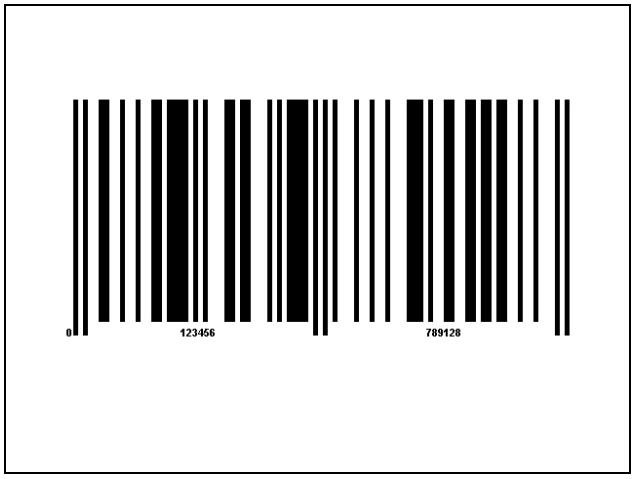
filnamn	**checksum**	**resultat**
086x65.jpg	false	0
114x86.jpg	false	0
153x114.jpg	false	0
203x152.jpg	false	0

270x203.jpg	false	0
360x270.jpg	true	123456789128
480x360.jpg	true	123456789128
640x480.jpg	true	123456789128

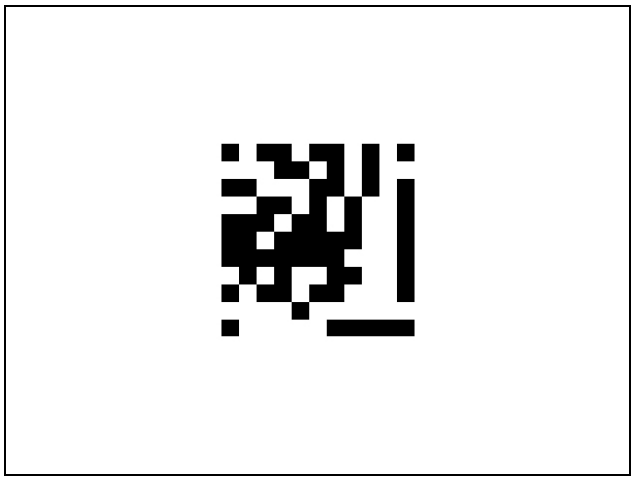
5 Resultat

Med hjälp av den framtagna metoden i denna uppsats testas tre olika avkodningsalgoritmer. Avkodare 1 är ett kommersiellt verktyg som kan testas som freeware under begränsade förhållanden, algoritmen är för streckkoden EAN13 och kallas för dBarScanJ. Avkodare 2 är en open-source algoritm för EAN13, algoritmen tillhör ett utvecklingsverktyg som kallas Papier-Mâché [8]. Avkodare nr 3 är också en open-source algoritm, den avkodar en ny typ av 2D kod som kallas för Visual Code [16], avkodare 3 går under namnet ETH. Avkodarna implementeras för att exekveras på en server (se kapitel 2.1.1, steg 1 och 4.4.2 serverlösning).

Bas- eller originalbilden som användes för att skapa referensbilder och testunderlag för EAN13 visas i figur 5.1 och bilden för Visual Code visas i figur 5.2.



Figur 5.1 - EAN13 basbild



Figur 5.2 - Visual Code basbild

Alla genererade bilder som skapas med hjälp av metoden för EAN13 visas i bilaga J och för Visual Code i bilaga K. Nedan presenteras resultaten från metoden och i kapitel 5.2 visas resultaten från experimentet.

5.1 Metoden

Siffrorna i tabellerna 5.1, 5.2, och 5.3 är data ut från metoden efter att tre olika avkodningsalgoritmer har testats utifrån metodiken beskriven i kapitel 4.2. Algoritm 1 i resultaten nedan representerar det kommersiella EAN13 verktyget, algoritm 2 är en open-source EAN13 avkodare, slutligen algoritm 3 representerar resultaten från open-source avkodaren till Visual Code. Resultaten i sin helhet, dvs. tabellerna som skapas med hjälp av kodskelettet i bilaga F presenteras i bilaga L och bilaga M. Nedan görs en sammanfattning av de resultaten. Tabellerna visar olika kategorier av bildstörningar. Inom varje kategori presenteras intervall där respektive algoritm gjorde lyckade avkodningar samt misslyckade avkodningar. Intervallen representeras av bildernas namn i varje kategori, namnen i sig är även en representation av störningsnivån i bilden.

Tabell 5.1 - Algoritm 1, dBarScanJ

Kategori	**Del**	**Intervall**		**Antal bilder**	**Algoritm 1**	
		avkodade bilder	*ej avkodade bilder*		*avkodade*	*procent*
Brus		G 05%.jpg - G 15%.jpg	-	11	11	100%
Distortion	Barrel	B 10.jpg - B 40.jpg	B 45.jpg - B 80.jpg	15	7	47%
	Pincushion	P 10.jpg - P 80.jpg	-	15	15	100%
Kontrast		10%.jpg - 50%.jpg	55 % - 80 %.jpg	15	9	60%
Perspektiv	Horisontell	H 04.jpg - H 20.jpg	H 22.jpg - H 26.jpg	12	9	75%
	Vertikal	V 04.jpg - V 12.jpg	V 14.jpg - V 18.jpg	8	5	63%
Rörelseoskärpa		01pix.jpg - 08pix.jpg	10pix.jpg - 18pix.jpg	13	8	62%
Upplösning	Jpg	640x480.jpg - 360x270.jpg	270x203.jpg - 86x65.jpg	8	3	38%
	Gif	640x480.gif - 360x270.gif	270x203.gif - 86x65.gif	8	3	38%
	totalt:			105	70	67%

Tabell 5.2 - Algoritm 2, Papier-Mâché

Kategori	**Del**	**Intervall**		**Antal bilder**	**Algoritm 2**	
		avkodade bilder	*ej avkodade bilder*		*avkodade*	*procent*
Brus		G 05%.jpg - G 15%.jpg	-	11	11	100%
Distortion	Barrel	B 10.jpg - B 80.jpg	-	15	15	100%
	Pincushion	P 10.jpg - P 80.jpg	-	15	15	100%
Kontrast		10%.jpg - 50%.jpg	55 %.jpg - 80 %.jpg	15	9	60%
Perspektiv	Horisontell	H 04.jpg - H 26.jpg	-	12	12	100%
	Vertikal	V 04.jpg - V 14.jpg	V 16.jpg - V 18.jpg	8	6	75%
Rörelseoskärpa		01pix.jpg - 05pix.jpg	06pix.jpg - 18pix.jpg	13	5	38%
Upplösning	Jpg	640x480.jpg - 270x203.jpg	203x152.jpg - 86x65.jpg	8	3	38%
	Gif	640x480.gif - 360x270.gif	270x203.gif - 86x65.gif	8	4	50%
	totalt:			105	80	76%

Tabell 5.3 - Algoritm 3, ETH

Kategori	**Del**	**Intervall**		**Antal bilder**	**Algoritm 3**	
		avkodade bilder	*ej avkodade bilder*		*avkodade*	*procent*
Brus		G 05%.jpg - G 11%.jpg	G 12%.jpg - G 15%.jpg	11	7	64%
Distortion	Barrel	B 10.jpg - B 40.jpg	B 45.jpg - B 80.jpg	15	7	47%
	Pincushion	P 10.jpg - P 70.jpg	P 75.jpg - P 80.jpg	15	13	87%
Kontrast		10%.jpg - 65%.jpg	65 % - 80 %.jpg	15	12	80%
Perspektiv	Horisontell	H 04.jpg - H 16.jpg	H 18.jpg - H 26.jpg	12	7	58%
	Vertikal	V 04.jpg - V 12.jpg	V 14.jpg - V 18.jpg	8	5	63%
Rörelseoskärpa		01pix.jpg - 18pix.jpg	-	13	13	100%
Upplösning	Jpg	640x480.jpg - 86x65.jpg	-	8	7	88%
	Gif	640x480.gif - 114x86.gif	86x65.gif	8	8	100%
	totalt:			105	79	75%

Totalt skapas med hjälp av metoden 105 bilder med olika förvrängningar. Utav dessa 105 bilder lyckades algoritm 1 att avkoda 70 bilder, vilket ger en avkodningsnivå på 67 %. Motsvarande siffror för algoritm 2 är 80 lyckade avkodningar och en nivå på 76 %. Algoritm 3 som avkodar Visual Code lyckades avkoda 79 utav 105 bilder och det ger då en avkodningsprocent på 75 %.

5.2 Experimentet

Siffrorna i diagram 5.1, 5.2 och 5.3 nedan är från experimentet som utfördes enligt kapitel 4.2. Experimentet gjordes för att om möjligt visa ett samband mellan de bruseffekter som simulerades i de datagenererade bilderna som skapas i metoden och bruseffekter från bilder tagna genom praktiska experiment. Om bilderna från experimentet visas ha liknande brusstörningar som de datagenererade bilderna så kan de datorsimulerade bilderna utnyttjas som en representation av verkligheten.

Experimentet utfördes, dvs. bilder togs fram, för tre olika kategorier:

1. Upplösning/avstånd.
2. Horisontell perspektivförvrängning.
3. Vertikal perspektivförvrängning

Resultaten i sin helhet visas i bilaga I.

5.2.1 Upplösning/Avstånd

Diagram 5.1 - Upplösning/Avstånd innehåller 3 kurvor, värdena anger antalet pixlar i bilden som representerar hela streckkoden horisontalt. Det finns endast 6 bilder var från experiment serierna Nokia och Siemens gentemot 8 bilder från den datorgenererade serien. Detta beror på att det inte gick att ta fler bilder med den ställningen som konstruerades och användes i experimenten.

a) Den gula kurvan med trekantiga punkter representerar datagenererade bilder från kategorin upplösning. Dessa bilder representerar bilder tagna i olika hög upplösning eller det går också att säga att det motsvaras av bilder tagna på olika långa avstånd mellan kamera och fotoobjekt.
b) Den rosa kurvan med fyrkantiga punkter representerar bilder ur experimentet tagna på olika avstånd med en Siemens SX1.

c) Den tredje mörkblåa kurvan med diamantformade punkter representerar bilder ur experimentet tagna med en Nokia 7610.

Y axel representerar antalet pixlar, X axeln representerar varje enskild series bild i kategorin (se tabell 5.4).

Tabell 5.4 - Diagram 5.1 bildlista

Bild nr	Nokia	Siemens	Data gen
1	Nokia (01).jpg	Siemens (01).jpg	640x480.jpg
2	Nokia (02).jpg	Siemens (02).jpg	480x360.jpg
3	Nokia (03).jpg	Siemens (03).jpg	360x270.jpg
4	Nokia (04).jpg	Siemens (04).jpg	270x203.jpg
5	Nokia (05).jpg	Siemens (05).jpg	203x152.jpg
6	Nokia (06).jpg	Siemens (06).jpg	153x114.jpg
7			114x86.jpg
8			86x65.jpg

Diagram 5.1 - Upplösning/Avstånd

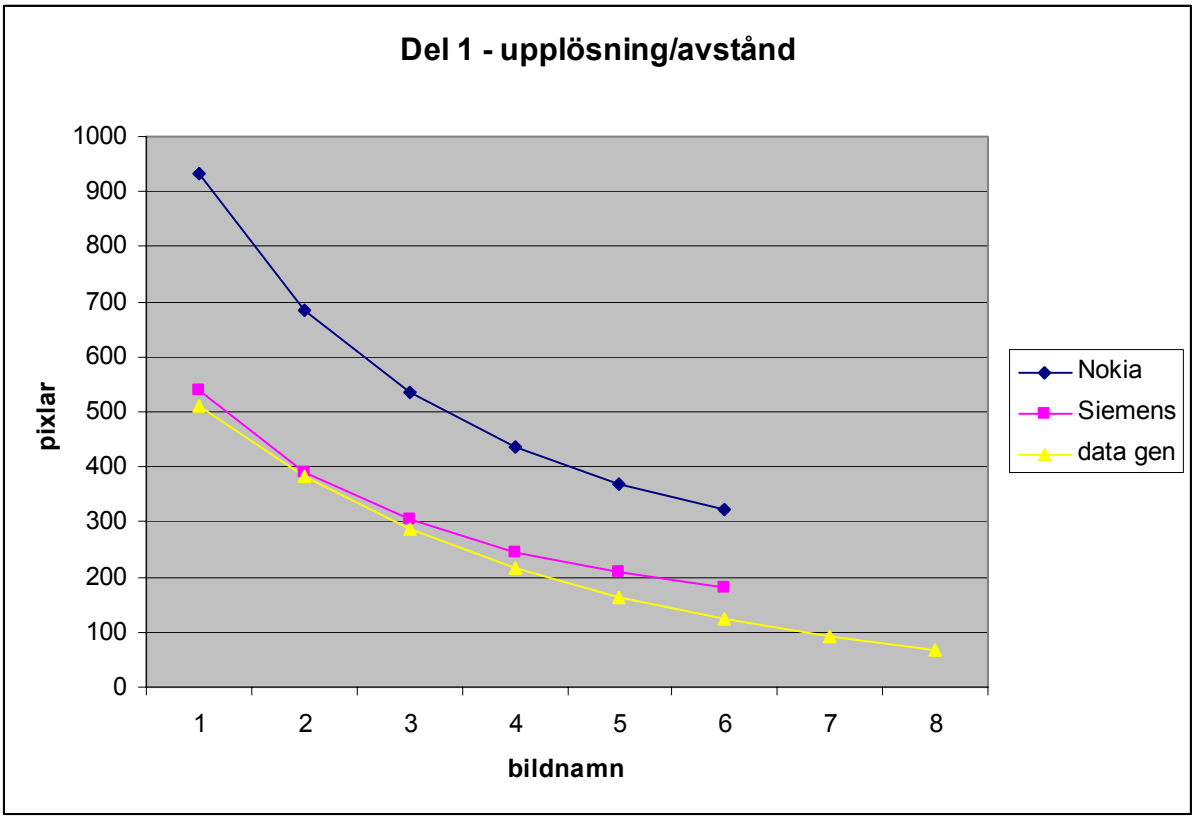


Diagram 5.1 tolkas som att antalet pixlar som representerar streckkoden i bilderna ändras i samma takt för alla tre representationer beroende på bildens storlek i pixlar. Tolkningen blir att de datorgenererade bilderna verkar ha ett samband med motsvarande effekter från de praktiska försöken.

5.2.2 Horisontell perspektivförvrängning

Diagram 5.2 visar horisontell perspektivförvrängning. Diagrammet innehåller två kurvor:

a) Kurvan med diamanter som punkter (den mörkblå kurvan) representerar bilder ur experimentet tagna med en Nokia 7610 med olika stor horisontellvinkel mellan streckkod och kamera.

b) Kurvan med fyrkanter som punkter (den rosa kurvan) representerar bilder från experimentet tagna med en Siemens SX1 med olika stor horisontellvinkel mellan streckkod och kamera.

c) Kurvan med treanglar som punkter (den gula kurvan) representerar bilder som blivit datorgenererade, dessa bilder hämtas från kategori perspektiv\horisontell.

Värdena på y-axeln visar den uppmätta vinkeln i bilden efter förvrängning har skett. På x-axeln visas varje enskild series bild i kategorin (se tabell 5.5).

Tabell 5.5 - Diagram 5.2 bildlista

Bild nr	Nokia	Siemens	Data gen
1	Nokia (09).jpg	Siemens (09).jpg	H 04.jpg
2	Nokia (10).jpg	Siemens (10).jpg	H 06.jpg
3	Nokia (11).jpg	Siemens (11).jpg	H 08.jpg
4	Nokia (12).jpg	Siemens (12).jpg	H 10.jpg
5	Nokia (13).jpg	Siemens (13).jpg	H 12.jpg
6	Nokia (14).jpg	Siemens (14).jpg	H 14.jpg
7	Nokia (15).jpg	Siemens (15).jpg	H 16.jpg
8	Nokia (16).jpg	Siemens (16).jpg	H 18.jpg
9	Nokia (17).jpg	Siemens (17).jpg	H 20.jpg
10	Nokia (18).jpg	Siemens (18).jpg	H 22.jpg
11	Nokia (19).jpg	Siemens (19).jpg	H 24.jpg
12	Nokia (20).jpg	Siemens (20).jpg	H 26.jpg

Diagram 5.2 - Horisontell perspektivförvrängning

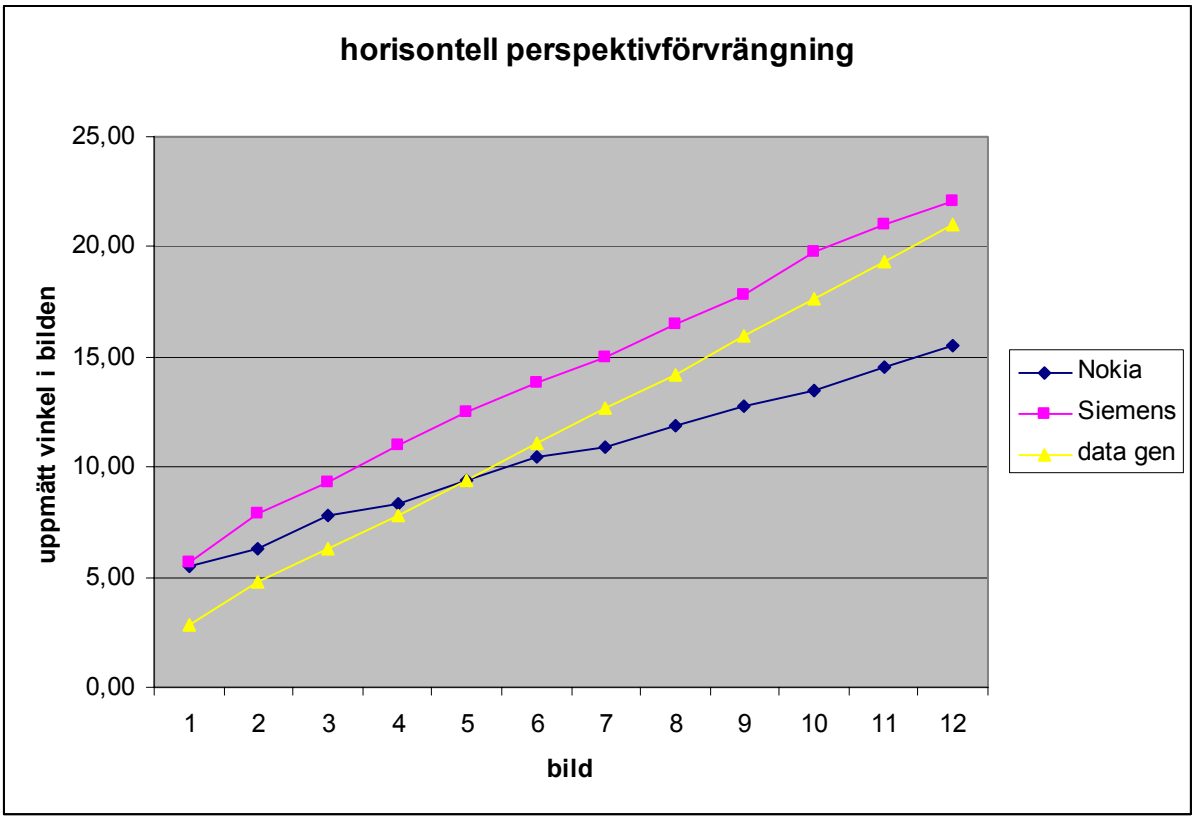


Diagram 5.2 visar en tydlig likhet mellan värdet på den förvrängda vinkeln hos bilder från den datorgenererade kategorin gentemot motsvarande vinkel i bilder från de experimentellt framtagna bilderna i kategorierna Nokia och Siemens.

5.2.3 Vertikal perspektivförvrängning

Diagram 5.3 anger data från vertikal perspektivförvrängning. Diagrammet innehåller två kurvor, kurvorna representerar data likt diagram 5.2:

a) Kurvan med diamanter som punkter (den mörkblå kurvan) representerar bilder ur experimentet tagna med en Nokia 7610 med olika stor vertikalvinkel mellan streckkod och kamera.

b) Kurvan med fyrkanter som punkter (den rosa kurvan) representerar bilder från experimentet tagna med en Siemens SX1 med olika stor vertikalvinkel mellan streckkod och kamera.

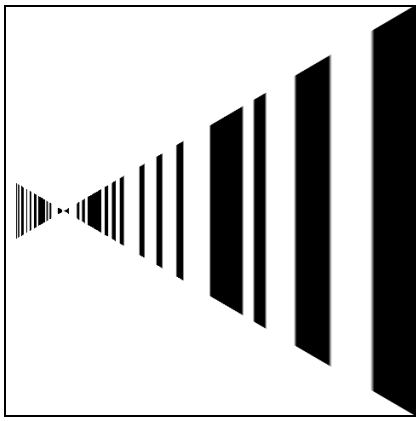
c) Kurvan med treanglar som punkter (den gula kurvan) representerar bilder som blivit datorgenererade, dessa bilder hämtas från kategori perspektiv\horisontell.

Värdena på y-axeln visar den uppmätta vinkeln i bilden efter förvrängning har skett. På x-axeln visas varje enskild series bild i kategorin (se tabell 5.6).

Tabell 5.6 - Diagram 5.3 bildlista

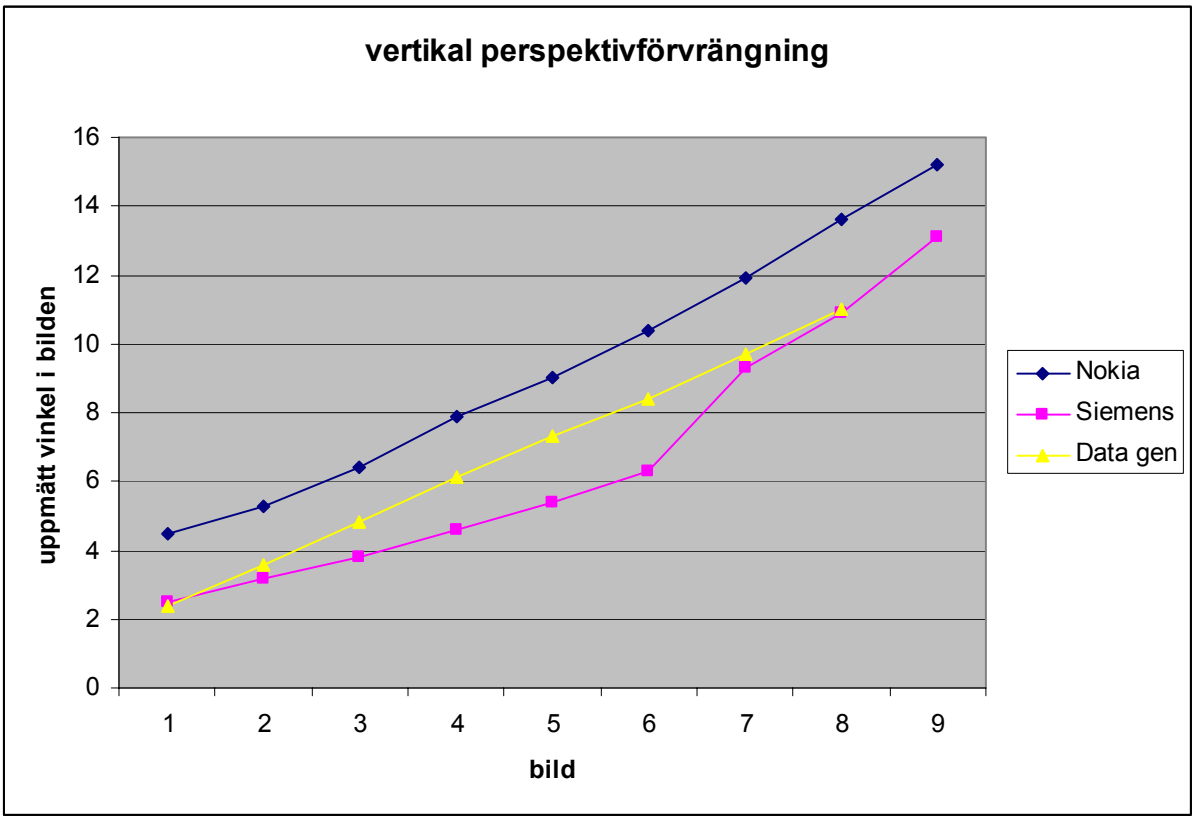
bild nr	Nokia	Siemens	Data gen
1	Nokia (21).jpg	Siemens (21).jpg	V 04.jpg
2	Nokia (22).jpg	Siemens (22).jpg	V 06.jpg
3	Nokia (23).jpg	Siemens (23).jpg	V 08.jpg
4	Nokia (24).jpg	Siemens (24).jpg	V 10.jpg
5	Nokia (25).jpg	Siemens (25).jpg	V 12.jpg
6	Nokia (26).jpg	Siemens (26).jpg	V 14.jpg
7	Nokia (27).jpg	Siemens (27).jpg	V 16.jpg
8	Nokia (28).jpg	Siemens (28).jpg	V 18.jpg
9	Nokia (29).jpg	Siemens (29).jpg	

I kolumnen för datagenererade bilder kunde inte fler bilder skapas bortom 18 grader. Gränsen från originalbild till bild med vertikal perspektivförvrängning stannade vid 18 grader på grund utav verktyget som användes i Photoshop. Gradtal ovanför 18 resulterade i bilder likt figur 5.3 nedan.



Figur 5.3 - Vertikal perspektivförvrängning ovanför 18 °

Diagram 5.3 - Vertikal perspektivförvrängning



I diagram 5.3 presenteras den uppmätta förvrängda vinkeln från de datorgenererade bilderna samt vinkeln från de experimentellt framtagna bilderna tagna med en Siemens SX1 respektive Nokia 7610. Den förvrängda vinkeln i de tre olika kategorierna följer varandra storleksmässigt. Även här finns en tydlig samband mellan de tre kategorierna vilket ger en autenticitet åt de datorgenererade bilderna.

5.3 Sammanfattning

Resultaten från experimenten gjord med de två mobiltelefonkamerorna i de tre kategorierna upplösning/avstånd, horisontell perspektivförvrängning och vertikal perspektivförvrängning har de datorgenererade bilderna i dessa kategorier kunnat kopplas ihop med verkligheten och därmed fått en äkthet. De datorgenererade bilderna i dessa kategorier kan således utnyttjas som en representation av verkligheten.

Resultaten från den framtagna metoden mot de tre olika avkodningsalgoritmer visar olikheter mellan de tre olika avkodningsalgoritmerna. Resultaten från metoden visar att 1D streckkoder och 2D streckkoder har olika fördelar, metoden bekräftar därmed bestämda fakta. Vidare visar metoden skillnader i prestationsförmåga mellan de två olika avkodarna för samma streckkod. Resultaten visar att avkodare 2 som tillhörde utvecklingsverktyget Papier-Mâché lyckats bättre med att avkoda bilderna från metoden än avkodare 1.

6 Slutsatser och Diskussion

Det finns många faktorer som påverkar och spelar in i en avbildningsprocess eller i ett bildgenereringssystem likt en mobiltelefonskameras. Det är svårt att mäta upp enskilda bruseffekter då mångfalden av brusfaktorer inte går att plocka bort på ett enkelt sätt. Önskas en specifik störningseffekt testas slinker även övriga störningskomponenter med in i bilden. Enskilda bruskomponenter går ej eller är svåra att isolera i praktiska experiment. Det är därför svårt att med säkerhet påvisa olika bruseffekters nivå och störningsgrad.

6.1 Metoden

Resultaten från kapitel 5 visar en tendens till att störningarna i kategorierna brus, distorsion (barrel, pincushion) och perspektiv verkar har liten påverkan på streckkoden för EAN13. Denna tendens med lyckade avkodningarna i kategorierna distorsion och perspektiv för EAN13 kan förklaras med hjälp av det faktum att bilderna representerar en 1D streckkod. En 1D streckkod har redundans i höjdled vilket betyder att störningar från dessa kategorier inte har samma inverkan som övriga störningar. Störningarna från kategorierna distorsion och perspektiv påverkar inte hela bilden utan endast delar av den, vissa delar utsätts för stora förvrängningar medan andra endast utsätts för mindre störningar. Detta ledar till att en 1D streckkod med redundans i höjdled ofta lyckas undkomma med en rad någonstans i koden som bara blivit påverkad till en liten grad av dessa faktorer och en lyckad avkodning ges som följd.

Kontrast, skärpa och upplösning verkar däremot vara egenskaper som har stor betydelse för en lyckad avkodning av EAN13 vilket heller inte är helt överraskande med tanke på de små byggstenarna i den 1D streckkoden. För att en lyckad avkodning ska kunna genomföras av EAN13 streckkoder är en bra kamera med hög upplösning och god skärpa och kontrast av stor betydelse. Linsförvrängningarna i systemen har liten inverkan, likaså har användarens inverkan på att ta bra kort, dvs. att ta kort på streckkoden i en rak vinkel gentemot den, ha en liten betydelse.

De faktorer som påverkar EAN13 mest verkar enligt testerna påverka Visual Code minst. De faktorer som har störst betydelse för att en lyckad avkodning ska kunna genomföras på Visual Code verkar enligt testet vara distorsion och perspektiv förvrängningar. En lyckad avkodning kan därför, om distorsion och perspektiv förvrängningarna är små, genomföras även med hjälp mindre bra kameror med tanke på upplösning, oskärpa och kontrast. För Visual code och 2D streckkoder har fotografens förmåga att ta bra bilder av streckkoden större betydelse jämfört med EAN13 och 1D streckkoder.

Figur 6.1 - EAN13 och Visual Code

I resultaten från metoden går det också att urskilja skillnader mellan de två olika avkodarna för EAN13. Algoritm 2 lyckas avkoda 10 stycken eller 9 % fler bilder än algoritm 1. Metoden verkar därför i dessa två fall kunna användas för att på ett enkelt och snabbt sätt särskilja på två avkodningsalgoritmer för samma typ av streckkoder. Metoden kan också utvecklas och enkelt förbättras till att med större exakthet särskilja på olika avkodare genom att helt enkelt ha ett större urval av bilder, dvs. fler bilder med mindre skillnad i störningsavvikelse mellan varje bild. På så sätt skulle än mindre skillnader kunna upptäckas mellan två avkodningsalgoritmers avkodningsutfall.

6.2 Experimentet

I experimentets första försök jämförs antalet pixlar som representerar streckkoden horisontalt i bilden (diagram 5.1). Datorgenererade bilder tagna i olika hög upplösning jämförs med bilder ur experimentet tagna på olika avstånd med en Siemens SX1 och en Nokia 7610. Ett samband mellan de tre olika kurvorna visas i diagram 5.1. Antalet pixlar som representerar streckkoden i bilderna tagna med olika kameror på olika avstånd tycks likt de datorgenererade

bilderna minska med samma styrka. I det andra försöket jämförs den uppmätta förvrängningsvinkeln i bilden efter att en horisontell förvrängning har skett (diagram 5.2), dels genom simulering eller datorgenererade bilder samt från bilderna från det fysiska experimentet. Kurvorna och värdena från jämförelsen visas överensstämma och även här finns det ett samband mellan verklighet och simulation. I det tredje och sista försöket jämförs den vertikala förvrängningsvinkeln i bilderna (diagram 5.3), dels från experimentet och dels från de datorgenererade bilderna och likt ovan går det även här urskilja ett samband mellan datorgenererade bilder och bilder tagna i experimentet.

De datorgenererade bilderna från kategorierna upplösning/avstånd, horisontell perspektivförvrängning och vertikal perspektivförvrängning har genom experiment genomgått en undersökning för att påvisa dess likhet med motsvarande bildförvrängningar från verkligheten. Försöken visar att ett samband finns och verifierar metodens riktighet i dessa kategorier, datorgenererade bilder i dessa kategorier har därmed en länk till den fysiska världen. Experimenten visade att de datorgenererade bilder framtagna med metoden kan användas som modell för verkliga bildförvrängningar.

6.3 Vilket kodsystem lämpar sig bäst för mobilkamera

Vid streckkodsavläsning med hjälp av dagens mobiltelefonkameror, som än så länge då denna rapport påbörjades har en begränsad upplösning går det påstå att utvecklingen av streckkoder för mobiltelefoner är i ungefär samma skede som vid utvecklingen av de allra första typerna av streckkoder, låg avläsningsupplösning är en gemensam begränsade faktor då som nu. Med en låg upplösning i instrumentet för avläsning är behovet av en streckkod som lätt kan avläsas större än mängden av information som det går att lagra. Därmed är de streckkoder som designats för en enkel avkodning med en hög avkodningsprocent även de som med fördel skulle kunna avkodas med hjälp av en mobilkamera med låg upplösning.

En 1D streckkodstandard som bara använder sig av kodningselement med två olika storlekar (se kapitel 3), dvs. som endast använder sig av ett smal till bred förhållande är att föredra. Ett sådant streckkodsystem är enklare att avläsa då längden på elementen blir enklare att särskilja

även i låga upplösningar. Ett tidigt system som tagits fram för enkel avkodning så som Codabar (se kapitel 3.2.1) skulle därför kunna vara fördelaktigt vid avkodning med mobilkamera jämfört med EAN13 som använder sig av ett system med upp till 3 olika bredder på streckelementen.

Ett system med hög densitet kan också ha fördelar och vara enkel att avkoda. Detta genom att man på samma yta som en låg densitets system kan ha plats för en streckkod med en högre X dimension, vilket gör att det blir enklare att skilja svarta och vita fält från varandra. Det är det som gör att 2D streckkoder är enklare att avkoda med låg upplöslig utrustning. Enskilda kodningselement är oftast större i 2D streckkoder än motsvarande kodningselement i 1D streckkoder.

6.4 Framtida utveckling

Tvådimensionella streckkoder verkar enligt resultaten vara att föredra om syftet för användandet är en interaktion med mobiltelefoner. Det är dock inte helt självklart vilken typ av 2D streckkod som är bäst lämpad för denna uppgift. I undersökningarna ovan testades med metoden endast en 2D streckkod då endast en open-source avkodare fanns tillgänglig vid det försökens tidpunkt. Det hade varit intressant att jämför Visual Code med andra 2D streckkoder exempelvis Aztec koden (se kapitel 3.2.2) för att genom test försöka avgöra vilken specifikation som är mest lämpad för avkodning med hjälp av en mobiltelefonkamera. Det är också intressant att se att de flesta streckkoder på marknaden idag inte är specifikt framtagna för mobiltelefoner och dess teknik. Så frågan är om den optimal streckkoden för mobilt användande ännu har tagits fram.

Visual Code (se figur 6.3) [16] tillsammans med TRIP Code (se figur 6.2) [11] [19] är dock två 2D streckkodsstandarder som är bland de första försöken till att designa en streckkod för det mobila användandet. TRIP Code är dessutom framtagen ur ett visuellt tilldragande form- eller designtänkande. TRIP Code har ett runt format likt bullseye i Maxi Code (se kapitel 3.2.2) och kan anses som lite mer tilltalande eller uppmärksammande jämfört med andra

framtagna 2D streckkoder. Den har dock jämförelsevis med de flesta 2D streckkoder en tämligen låg datadensitet.

Figur 6.2 - TRIP Code

Visual Code är framtagen för att motstå bl.a. barrel effekterna från mobiltelefonkameror, detta bl.a. med hjälp av de två avlånga guidepaneler som finns i symbolens nedre högra hörn. Dessa två 2D streckkoder kan vara framtidens streckkod för mobila sammanhang.

Figur 6.3 - Visual Code

6.5 Sammanfattning

Metoden som tagits fram i denna rapport har enligt resultaten gett en möjlighet att på ett icke-komplicerat sätt testa avkodningsalgoritmers avkodningskapacitet och dess möjlighet att motverka bruseffekternas inverkan samt avgöra vilken algoritm som kan anses vara bättre än den andra. Resultaten ut från metoden stärks genom de fakta som togs upp i rapportens inledande bakgrundbeskrivning (kapitel 3) angående de olika streckkodernas för och nackdelar. Metodens resultat visar att faktorer som brus, distorsion och perspektiv har mindre påverkan på den 1D streckkoden EAN13 samt att upplösning och kontrast har minst effekt på den 2D streckkoden Visual Code.

7 Referenser

Tryckta referenser

[1] Bailey, D., *Image Capture Modelling for High Resolution Reconstruction*. : Institute of Information Sciences and Technology, Massey University, New Zealand, 1998.

[2] Bailey, D., *Super-resolution of bar codes*. : Institute of Information Sciences and Technology Massey University, New Zealand : Journal of Electronic Imagine 10, pp 213-220, 2001.

[3] Barnes, D., Bradshaw, J., Day, L., Schott, T., Wilson, R., *Two Dimensional Bar Coding.* : Technical Report - Purdue University, 1999.

[4] Bengtsson, E., *Digital Imaging Systems - Lecture 2, Digital cameras, CMOS and CCD sensors.* : Uppsala Universitet, SLU, Centre for Image Analysis, 2004.

[5] Deitel, H., M., Deitel, P., J., *Java : how to program.* : Upper Saddle River, N.J. : Pearson Education International, 2003.

[6] Gfeller, B., *Recognition of 2-dimensional visual codes with the Nokia 7650.* : Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, 2003.

[7] Gonzalez, R., C., Woods, R., E., *Digital image processing.* : Upper Saddle River, N.J. : Prentice Hall, 2002

[8] Klemmer, S., Li, J., Lin, J., Landay, J., *Papier-Mâché: Toolkit Support for Tangible Input.* : Group for User Interface Research, Computer Science Division, University of California and DUB Group, Computer Science & Engineering, University of Washington, 2003.

[9] Larsson, A., Andersson, M., *Introduktion till feldetekterande och felkorrigerande koder.* : Rapport, Högskolan Karlskrona Ronneby, 1997.

[10] Liao, H-Y., Liu, S-J., Chen, L-H., Tyan, H-R., *A Bar-code Recognition System Using Backpropagation Neural Networks.* : Engineering Applications of Artificial Intelligence, nr 8, pp 81 - 90, 1995.

[11] Lóez de Ipiña, D., Mendonc, P., Hopper, A., *TRIP: A Low-Cost Vision-Based Location. System for Ubiquitous Computing.* : Laboratory for Communications Engineering, University of Cambridge, UK. Fallside Laboratory, University of Cambridge, UK. AT&T Laboratories Cambridge, UK. : Springer London Ltd, Personal and Ubiquitous Computing 6:206–219, 2002.

[12] Norstedt, S., *2D printcode recognition.* : Master's theses in mathematical sciences. : ISSN 1404-5342 ; 2002:E43 : Lund Univ., 2002.

[13] Orwell, George., *1984.* : Skönlitteratur - Höganäs : Bra böcker, 1984. - ISBN 99-0429695-2.

[14] Reed, I., S., Solomon, G., *Polynomial codes over certain finite fields.* Journal of the Society for Industrial and Applied Mathematics, 8:300-304, 1960.

[15] Rohs, M., *Real-World Interaction with Camera-Phones*. : Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland, 2004.

[16] Rohs, M., Gfeller, B., *Using Camera-Equipped Mobile Phones for Interacting with Real-World Objects.* : Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, Switzerland : Advances in Pervasive Computing, Austrian Computer Society (OCG), pp. 265-271, Vienna, Austria, 2004.

[17] Shah, S., S., Yaqub, S., Suleman, F., *Self-correcting codes conquer noise, part 2.* : Reed-Solomon codecs. : EDN, pp 107, issue 3/15/2001.

[18] Shannon, C. E., *A Mathematical Theory of Communication.* : Bell System Technical Journal, vol 27, pp 379-423, 623-656, 1948.

[19] Tuddenham, P., Vukovic, M., *Ubiquitous Service Discovery using TRIP* : University of Cambridge Computer Laboratory, 2003

[20] UMTS Forum, *Mobile evolution - shaping the future.* : UMTS Forum, United Kingdom : White paper, 2003.

[21] Wagner, N., R., *The Laws of Cryptography with Java Code.* : Electronic published book. : University of Texas, San Antonio, 2003.

[22] Yu, B., Jain, A., K., *A Robust and fast skew detection algoritm for generic documents.* : Department of Computer Science, Michigan State University. U.S.A. : Elsevier Science Ltd, Pattern Recognition, Vol. 29, No. 10, pp. 1599-1629, 1996.

Otryckta referenser

[23] 4i2i Communications Ltd., http://www.4i2i.com/reed_solomon_codes.htm, 2005-03-02.

[24] Acuity CiMatrix, http://www.rvsi.com/, 2005-03-02.

[25] adis - automatic data entry and identification systems, http://www.ades.ch/PDF-Files/Informationen zum Aztec Code.pdf, 2005-05-09.

[26] Adobe Systems Incorporated, http://www.adobe.com/, 2005-05-09.

[27] AIM Global, the Association for Automatic Identification and Mobility, http://www.aimglobal.org/technologies/barcode/, 2005-02-25.

[28] American National Standards Institute, http://www.ansi.org/, 2005-03-02.

[29] Aztec Barcode Symbology Specification Rev 3.0, http://www.ritservice.ru/products/aps/aztece.asp, 2005-05-09.

[30] BarCode 1, a web of information about barcode, http://www.adams1.com/, 2005-02-25.

[31] BARCODE Island, http://www.barcodeisland.com/, 2005-02-28.

[32] Denso Wave, http://www.denso-id.com/, 2005-03-02.

[33] EAN International, http://www.ean-int.org/, 2005-02-28.

[34] Extended Binary Coded Decimal Interchange Code, http://www.ibm.com, 2005-03-02.

[35] Hypermedia Image Processing Reference, Fisher, B., Perkins, S., Walker, A., Wolfart, E., Department of Artificial Intelligence, University of Edinburgh, UK. http://www.cee.hw.ac.uk/hipr/html/threshld.html, 2005-05-09.

[36] Intermec Technologies Corporation, http://www.intermec.com/, 2005-03-02.

[37] International Symbology Specification, http://www.iso.org/, 2005-03-02.

[38] KCSI, The Advantages of Barcoding, http://www.kcsi.ca/barcoding_adv.html, 2005-05-09.

[39] Morovia Corporation, http://morovia.com/education/symbology/, 2005-02-28.

[40] MTB MobilTeleBranschen, http://www.mtb.se/, 2005-02-25.

[41] Nippon Barcode Co.,Ltd. , http://www.n-barcode.com/en/index-en.html, 2005-02-28.

[42] Nokia Mobile Phones, http://www.nokia.se/, 2005-05-18.

[43] Panorama Tools documentation, Helmut Dersch, http://home.no.net/dmaurer/~dersch/Index.htm, 2005-05-09.

[44] Panorama Tools, http://home.no.net/dmaurer/~dersch/PanoTools.zip, 2005-05-09.

[45] Samsung, CeBIT 2005, http://www.samsung.com/common/microsite/exhibition/cebit2005/press/press19.html, 2005-04-11.

[46] Siemens Communications consumer site, http://www.my-siemens.com, 2005-05-18.

[47] Sony Ericsson Mobile Communications AB, http://www.sonyericsson.com/, 2005-05-18.

[48] Symbol Technologies, http://www.symbol.com/, 2005-03-02.

[49] The Barcode Software Center, http://www.mecsw.com/info/info.html, 2005-02-28.

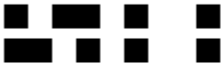
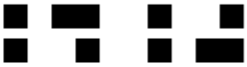
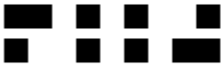
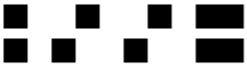
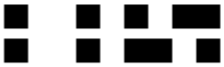
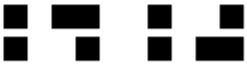
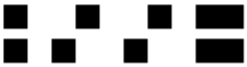
[50] Uniform Code Council, Inc., http://www.uc-council.org/, 2005-02-28.

[51] United Parcel Service, http://www.ups.com, 2005-03-02.

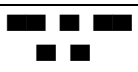
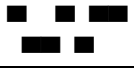
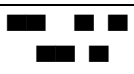
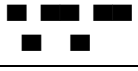
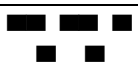
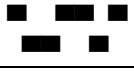
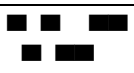
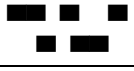
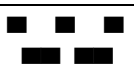
[52] Åbjörnsson, C., Södergren, C., Svensson, J., Soukka, L. Error Correction. Sammanfattning ur Poynton, C., A. A Technical Introduction to Digital Video. http://www.itn.liu.se/~dagha/TNM045-2001/kap 9w_03 error correction.pdf

Bilagor

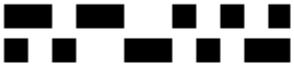
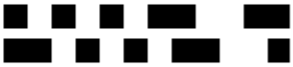
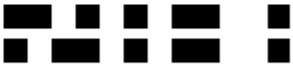
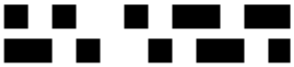
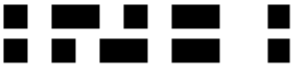
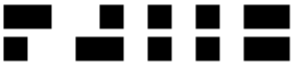
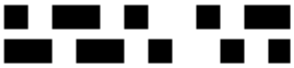
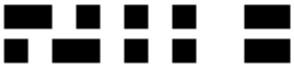
Bilaga A - Codabar

Char.	Pattern	Bars	Spaces	Char.	Pattern	Bars	Spaces
0		0001	001	:		1011	000
1		0010	001	/		1101	000
2		0001	010	.		1110	000
3		1000	100	+		0111	000
4		0100	001	a		0100	011
5		1000	001	b		0001	110
6		0001	100	c		0001	011
7		0010	100	d		0010	011
8		0100	100	t		0100	011
9		1000	010	n		0001	110
-		0010	010	*		0001	011
$		0100	010	e		0010	011

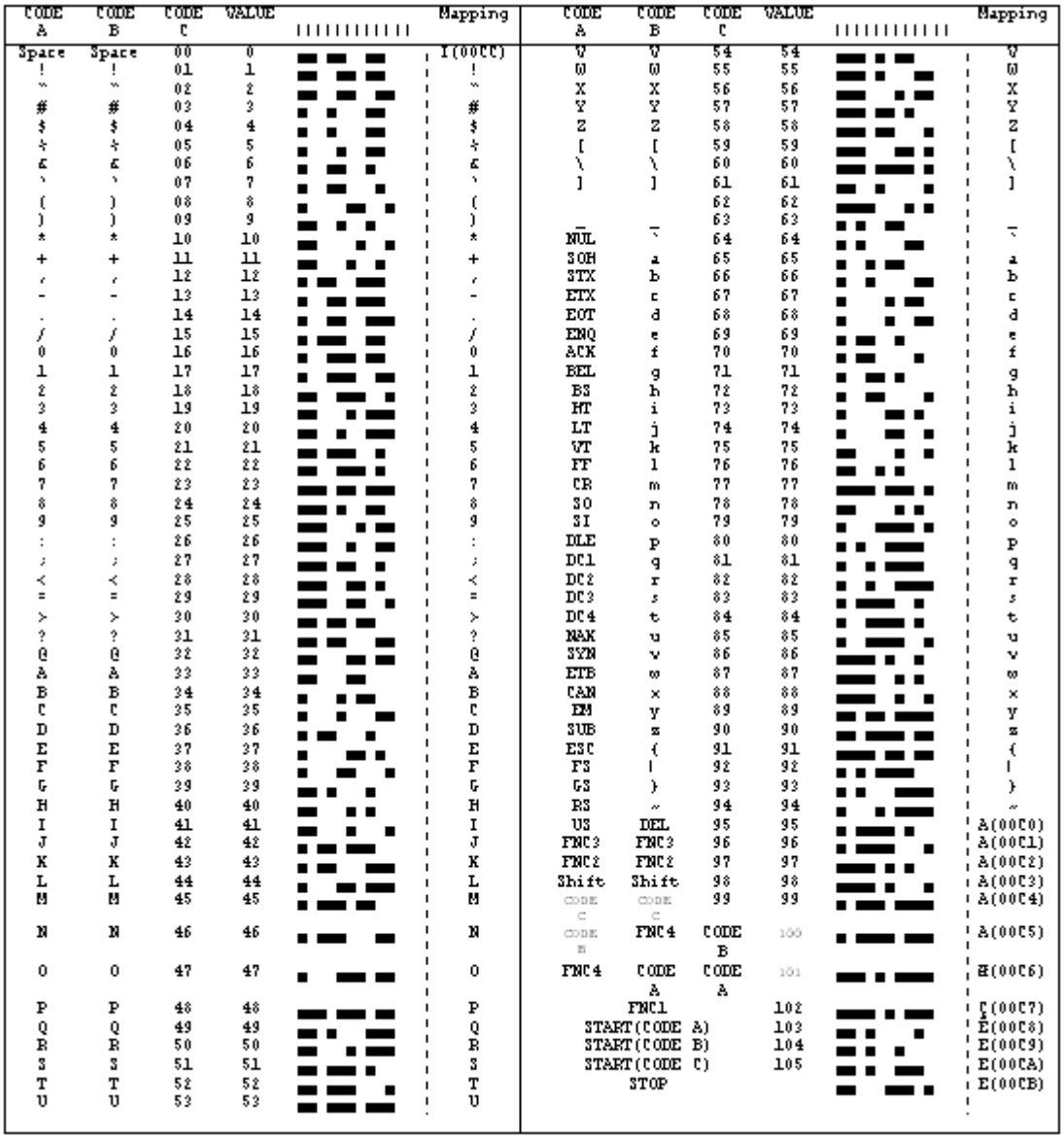
Bilaga B - Interleaved 2/5

Tecken	Binärt	Pattern	
0	00110		NNWWN
1	10001		WNNNW
2	01001		NWNNW
3	11000		WWNNN
4	00101		NNWNW
5	10100		WNWNN
6	01100		NWWNN
7	00011		NNNWW
8	10010		WNNWN
9	01010		NWNWN

Bilaga C - Code 39

Char.	Pattern	Bars	Spaces	Char.	Pattern	Bars	Spaces
1		10001	0100	M		11000	0001
2		01001	0100	N		00101	0001
3		11000	0100	O		10100	0001
4		00101	0100	P		01100	0001
5		10100	0100	Q		00011	0001
6		01100	0100	R		10010	0001
7		00011	0100	S		01010	0001
8		10010	0100	T		00110	0001
9		01010	0100	U		10001	1000
0		00110	0100	V		01001	1000
A		10001	0010	W		11000	1000
B		01001	0010	X		00101	1000
C		11000	0010	Y		10100	1000
D		00101	0010	Z		01100	1000
E		10100	0010	-		00011	1000
F		01100	0010	.		10010	1000
G		00011	0010	Space		01010	1000
H		10010	0010	*		00110	1000
I		01010	0010	$		00000	1110
J		00110	0010	/		00000	1101
K		10001	0001	+		00000	1011
L		01001	0001	%		00000	0111

Bilaga D - Code 128



CODE A	CODE B	CODE C	VALUE	Mapping
Space	Space	00	0	I(00CC)
!	!	01	1	!
"	"	02	2	"
#	#	03	3	#
$	$	04	4	$
%	%	05	5	%
&	&	06	6	&
'	'	07	7	'
(	)	08	8	(
)	)	09	9	)
*	*	10	10	*
+	+	11	11	+
,	,	12	12	,
-	-	13	13	-
.	.	14	14	.
/	/	15	15	/
0	0	16	16	0
1	1	17	17	1
2	2	18	18	2
3	3	19	19	3
4	4	20	20	4
5	5	21	21	5
6	6	22	22	6
7	7	23	23	7
8	8	24	24	8
9	9	25	25	9
:	:	26	26	:
;	;	27	27	;
<	<	28	28	<
=	=	29	29	=
>	>	30	30	>
?	?	31	31	?
@	@	32	32	@
A	A	33	33	A
B	B	34	34	B
C	C	35	35	C
D	D	36	36	D
E	E	37	37	E
F	F	38	38	F
G	G	39	39	G
H	H	40	40	H
I	I	41	41	I
J	J	42	42	J
K	K	43	43	K
L	L	44	44	L
M	M	45	45	M
N	N	46	46	N
O	O	47	47	O
P	P	48	48	P
Q	Q	49	49	Q
R	R	50	50	R
S	S	51	51	S
T	T	52	52	T
U	U	53	53	U

CODE A	CODE B	CODE C	VALUE	Mapping
V	V	54	54	V
W	W	55	55	W
X	X	56	56	X
Y	Y	57	57	Y
Z	Z	58	58	Z
[	[	59	59	[
\	\	60	60	\
]	]	61	61	]
		62	62	
_	^	63	63	^
NUL	`	64	64	`
SOH	a	65	65	a
STX	b	66	66	b
ETX	c	67	67	c
EOT	d	68	68	d
ENQ	e	69	69	e
ACK	f	70	70	f
BEL	g	71	71	g
BS	h	72	72	h
HT	i	73	73	i
LT	j	74	74	j
VT	k	75	75	k
FF	l	76	76	l
CR	m	77	77	m
SO	n	78	78	n
SI	o	79	79	o
DLE	p	80	80	p
DC1	q	81	81	q
DC2	r	82	82	r
DC3	s	83	83	s
DC4	t	84	84	t
NAK	u	85	85	u
SYN	v	86	86	v
ETB	w	87	87	w
CAN	x	88	88	x
EM	y	89	89	y
SUB	z	90	90	z
ESC	{	91	91	{
FS	|	92	92	|
GS	}	93	93	}
RS	~	94	94	~
US	DEL	95	95	A(00C0)
FNC3	FNC3	96	96	A(00C1)
FNC2	FNC2	97	97	A(00C2)
Shift	Shift	98	98	A(00C3)
CODE C	CODE C	99	99	A(00C4)
CODE B	FNC4	CODE B	100	A(00C5)
FNC4	CODE A	CODE A	101	H(00C6)
FNC1			102	[(00C7)
START(CODE A)			103	E(00C8)
START(CODE B)			104	E(00C9)
START(CODE C)			105	E(00CA)
STOP				E(00CB)

Bilaga E - Bat fil för generering av katalogstruktur

Observera att tecknet för ’ö’ och ’ä’ i DOS skrivs med ” respektive „ i Windows.

```
@ECHO OFF
c:
cd\
md metoden
cd metoden
md brus
md distortion
md filter
md kontrast
md perspektiv
md r”relseosk„rpa
md uppl”sning
cd distortion
md barrel
md pincushion
cd..\perspektiv
md horisontell
md vertikal
cd..\uppl”sning
md gif
md jpg
cd\
```

Bilaga F - Javaprogram

Javaprogram skelettstruktur för avkodning av valda datorgenererade bilder med valbar algoritm.

```
/*  import decoding algoritm functions */

import com.briscan.ean13.EanDecoder;

/* ------------------------------------------- */

import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.FilenameFilter;
import java.io.PrintWriter;
import java.io.IOException;
import java.io.ObjectOutputStream;

public class Stanford {
      public static File imageFile;
      public static String filend;
      public static int antal_kategorier;
      public static int totalt_Filer;
      public static PrintWriter utFil;
      public static ObjectOutputStream output;

      public static void output(String ord)
      {     utFil.println(ord);}

      public static void closefile()
      {     utFil.close(); }

      public static void openfile(String path ,String name)
      {
            File nfil = new File( path + "//" + name  );
            try   {
                  utFil = new PrintWriter (new BufferedWriter(new FileWriter (nfil)));
            }
            catch (IOException e)
            {     }
      }

     //      Process directories under dir
   public static void visitAllDirs(File dir)
   {
       if (dir.isDirectory())
       {
            FilenameFilter filter = new FilenameFilter() {
             public boolean accept(File dir, String name) {
                return (name.endsWith(".jpg") || name.endsWith(".gif"));
             }
          };

          String[] fillist = dir.list(filter);            // fillist[i] holds all files in the category.
          String path = dir.getAbsolutePath();
```

```
if(path.endsWith("metoden") || path.endsWith("filter") || path.endsWith("droplet"))
{
}
else
{
  if( !(fillist.length == 0))
  {
      antal_kategorier++;              // inc number of categories.
      totalt_Filer += fillist.length;  // inc number of files.

      output("\n" + path);             // write category to result file
      output("filnamn" + "\t" + "checksum" + "\t" + "resultat");          // write headlines in result file.

      for (int i=0; i<fillist.length; i++)
      {
          /*   INIT SAVE VARIABLES   */
          /*   change if needed   */
          int svar;
          String resultat;
          boolean check;
          /* ----------------------------------   */

          imageFile = new File(path + "\\" + fillist[i]);    // set path and name of file to read.
          BufferedImage bi = null;

          try {      bi = ImageIO.read(imageFile);  }       // read imagefile.
          catch (java.io.IOException e) {
              System.out.println(e);
              System.exit(1);
          }


          /*   DECODER STUFF HERE   */
          /*   change if needed    */
          /*   the image with the barcode is stored in the variable 'bi' as a BufferedImage */

          barc.decode(bi);         // sending BufferedImage 'bi' to decoder.

          check = EanDecoder.eanParsed.cchecksum;

          StringBuffer sBuf = new StringBuffer();
          for (int j = 0; j < 13; ++j)
          {
                  sBuf.append((char) (EanDecoder.eanParsed.code[j] + '0'));
          }
          resultat = sBuf.toString();       // get result


          /*   SAVE RESULT TO FILE   */
          output(fillist[i] + "\t" + check + "\t" + resultat);

          /* ----------------------------------   */

          /* ----------------------------------   */
          System.out.print(".");
      }
  }
```

```
        }

          String[] children = dir.list();

        for (int i=0; i<children.length; i++)
        {
           visitAllDirs(new File(dir, children[i]));
        }
    }
  }


    /* INIT DECODER NAME */
    public static EanDecoder barc;
    /* ----------------------------    */

    public static void main(String[] args)
    {
        /* INIT DECODER HERE */
        barc = new EanDecoder();
        /* ------------------------------- */

        openfile(args[0], args[1]);        // open file for result storage
        File dir = new File( args[0] );
        visitAllDirs(dir);                 // begin testing

        System.out.println(antal_kategorier);
        System.out.println(totalt_Filer);

        closefile();
    }
}
```

Bilaga G - Aztec

	Upper		Lower		Mixed		Punct		Digit	
Value	Char	ASCII	Char	ASCII	Char	ASCII	Char	ASCII	Char	ASCII
0	PS		PS		PS		FLG(n)	***	PS	
1	SP	32	SP	32	SP	32	CR	13	SP	32
2	A	65	a	97	SOH	1	CR LF	13,10	0	48
3	B	66	b	98	STX	2	. SP	46,32	1	49
4	C	67	c	99	ETX	3	, SP	44,32	2	50
5	D	68	d	100	EOT	4	: SP	58,32	3	51
6	E	69	e	101	ENQ	5	!	33	4	52
7	F	70	f	102	ACK	6	"	34	5	53
8	G	71	g	103	BEL	7	#	35	6	54
9	H	72	h	104	BS	8	$	36	7	55
10	I	73	i	105	HT	9	%	37	8	56
11	J	74	j	106	LF	10	&	38	9	57
12	K	75	k	107	VT	11	'	39	,	44
13	L	76	l	108	FF	12	(	40	.	46
14	M	77	m	109	CR	13	)	41	UL	
15	N	78	n	110	ESC	27	*	42	US	
16	O	79	o	111	FS	28	+	43		
17	P	80	p	112	GS	29	,	44		
18	Q	81	q	113	RS	30	-	45		
19	R	82	r	114	uS	31	.	46		
20	S	83	s	115	@	64	/	47		
21	T	84	t	116	\	92	:	58		
22	U	85	u	117	^	94	;	59		
23	V	86	v	118		95	<	60		
24	W	87	w	119	`	96	=	61		
25	X	88	x	120	|	124	>	62		
26	Y	89	y	121	~	126	?	63		
27	Z	90	z	122	DEL	127	[	91		
28	LL		US		LL		]	93		
29	ML		ML		UL		{	123		
30	DL		DL		PL		}	125		
31	BS		BS		BS		UL			

Bilaga H - Bilder från Experimentet

Siemens Sx1

Bildstorlek (upplösning) 640 x 480.

Avstånd

Siemens(01).jpg, 15 cm avstånd

Siemens(02).jpg, 20 cm avstånd

Siemens(03).jpg, 25 cm avstånd

Siemens(04).jpg, 30 cm avstånd

Siemens(05).jpg, 35 cm avstånd

Siemens(06).jpg, 40 cm avstånd

Horisontell förvrängning, Y
Avstånd mellan kamera och streckkod 15 cm.


Siemens (09).jpg, 10°


Siemens (10).jpg, 15°


Siemens (11).jpg, 20°


Siemens (12).jpg, 25°


Siemens (13).jpg, 30°


Siemens (14).jpg, 35°


Siemens (15).jpg, 40°


Siemens (16).jpg, 45°


Siemens (17).jpg, 50°


Siemens (18).jpg, 55°


Siemens (19).jpg, 60°


Siemens (20).jpg, 65°

Vertikal förvrängning, Y
Avstånd mellan kamera och streckkod 15 cm.


Siemens (21).jpg, 10°

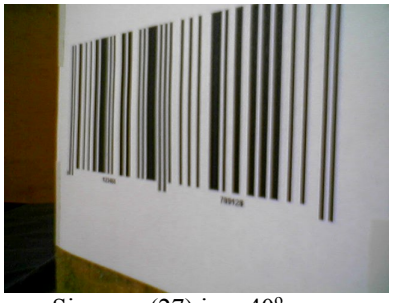

Siemens (22).jpg, 15°

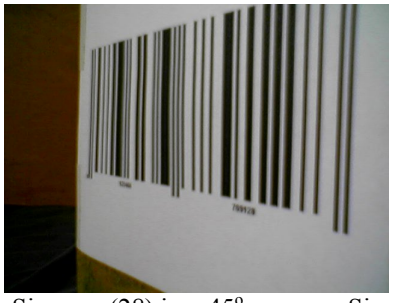

Siemens (23).jpg, 20°

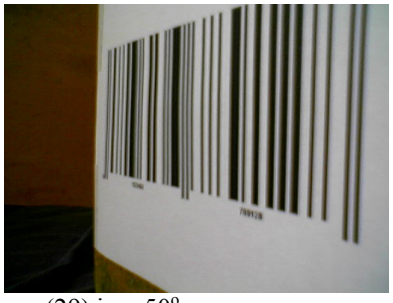

Siemens (24).jpg, 25°


Siemens (25).jpg, 30°


Siemens (26).jpg, 35°


Siemens (27).jpg, 40°

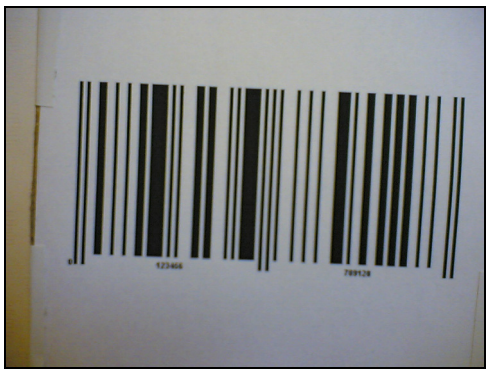

Siemens (28).jpg, 45°


Siemens (29).jpg, 50°

Nokia 7610

Bildstorlek (upplösning) 1152 x 864.

Avstånd



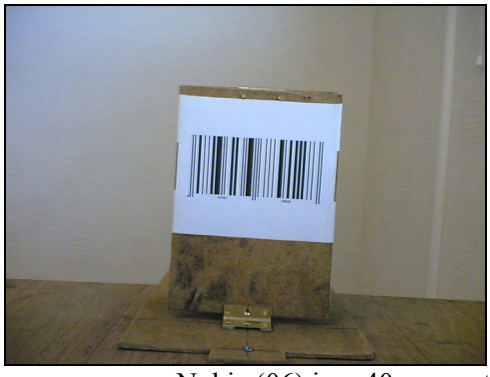
Nokia (01).jpg, 15 cm avstånd

Nokia (02).jpg, 20 cm avstånd

Nokia (03).jpg, 25 cm avstånd

Nokia (04).jpg, 30 cm avstånd

Nokia (05).jpg, 35 cm avstånd



Nokia (06).jpg, 40 cm avstånd

Horisontell förvrängning, Y
Avstånd mellan kamera och streckkod 15 cm.


Nokia (09).jpg, 10°


Nokia (10).jpg, 15°


Nokia (11).jpg, 20°


Nokia (12).jpg, 25°


Nokia (13).jpg, 30°


Nokia (14).jpg, 35°


Nokia (15).jpg, 40°


Nokia (16).jpg, 45°


Nokia (17).jpg, 50°


Nokia (18).jpg, 55°


Nokia (19).jpg, 60°


Nokia (20).jpg, 65°

Vertikal förvrängning, Y
Avstånd mellan kamera och streckkod 15 cm.

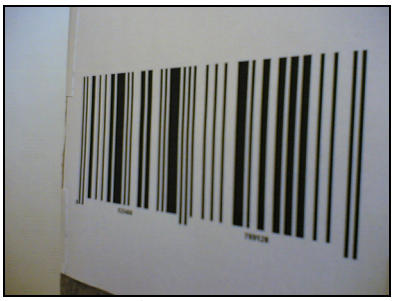

Nokia (21).jpg, 10°

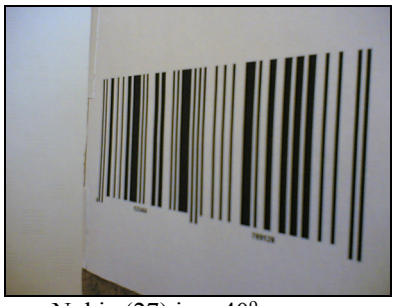

Nokia (22).jpg, 15°

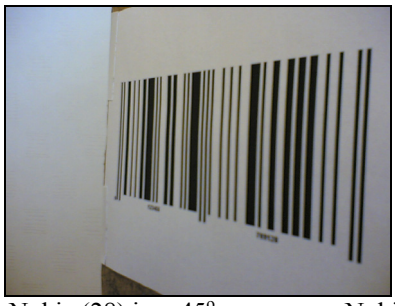

Nokia (23).jpg, 20°


Nokia (24).jpg, 25°


Nokia (25).jpg, 30°


Nokia (26).jpg, 35°


Nokia (27).jpg, 40°


Nokia (28).jpg, 45°


Nokia (29).jpg, 50°

Bilaga I - Resultat - jämförelse experiment mot simulation

Del 1 - Upplösning/avstånd

Figur 1 - Antal pixlar

Experiment bilder

Tabell 1

Nokia	bildnamn	avstånd	antal pixlar som representerar streckkoden horisontalt
Avstånd/Upplösning			
1152 x 864	Nokia (01).jpg	15 cm	933 pixlar
	Nokia (02).jpg	20 cm	683 pixlar
	Nokia (03).jpg	25 cm	534 pixlar
	Nokia (04).jpg	30 cm	435 pixlar
	Nokia (05).jpg	35 cm	369 pixlar
	Nokia (06).jpg	40 cm	322 pixlar

Tabell 2

Siemens	bildnamn	avstånd	antal pixlar som representerar streckkoden horisontalt
Avstånd/Upplösning			
640 x 480	Siemens (01).jpg	15 cm	540 pixlar
	Siemens (02).jpg	20 cm	391 pixlar
	Siemens (03).jpg	25 cm	305 pixlar
	Siemens (04).jpg	30 cm	246 pixlar
	Siemens (05).jpg	35 cm	209 pixlar
	Siemens (06).jpg	40 cm	181 pixlar

Datorgenererade bilder

Tabell 3

Data genererade	bildnamn	antal pixlar som representerar streckkoden horisontalt
Upplösning/avstånd		
	640x480.jpg	509 pixlar
	480x360.jpg	383 pixlar
	360x270.jpg	288 pixlar
	270x203.jpg	215 pixlar
	203x152.jpg	163 pixlar
	153x114.jpg	123 pixlar
	114x86.jpg	92 pixlar
	086x65.jpg	69 pixlar

Del 2 - Horisontell perspektivförvrängning

Figur 2 - Vinkel

Experiment bilder

Tabell 4

Nokia	bildnamn	Y vinkelinställning på ställningen	Uppmätt vinkelförändring för streckkoden i bilden
perspektivförvrägning	Nokia (09).jpg	10 grader	5.5 grader
horisontell	Nokia (10).jpg	15 grader	6.3 grader
upplösning 1152 x 864	Nokia (11).jpg	20 grader	7.8 grader
avstånd 15 cm	Nokia (12).jpg	25 grader	8.3 grader
	Nokia (13).jpg	30 grader	9.4 grader
	Nokia (14).jpg	35 grader	10.5 grader
	Nokia (15).jpg	40 grader	10.9 grader
	Nokia (16).jpg	45 grader	11.9 grader
	Nokia (17).jpg	50 grader	12.8 grader
	Nokia (18).jpg	55 grader	13.5 grader
	Nokia (19).jpg	60 grader	14.5 grader

	Nokia (20).jpg	65 grader	15.5 grader

Tabell 5

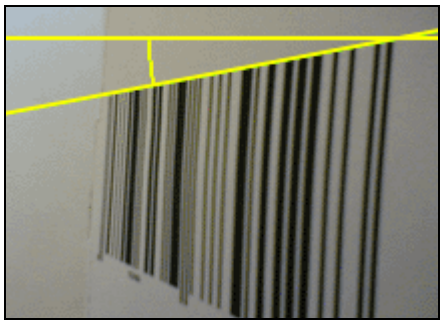
Siemens	bildnamn	Y vinkelinställning på ställningen	Uppmätt vinkelförändring för streckkoden i bilden
perspektivförvrägning	Siemens (09).jpg	10 grader	5,7 grader
horisontell	Siemens (10).jpg	15 grader	7,9 grader
upplösning 640 x 480	Siemens (11).jpg	20 grader	9,3 grader
avstånd 15 cm	Siemens (12).jpg	25 grader	11,0 grader
	Siemens (13).jpg	30 grader	12,5 grader
	Siemens (14).jpg	35 grader	13,8 grader
	Siemens (15).jpg	40 grader	15,0 grader
	Siemens (16).jpg	45 grader	16,5 grader
	Siemens (17).jpg	50 grader	17,8 grader
	Siemens (18).jpg	55 grader	19,8 grader
	Siemens (19).jpg	60 grader	21,0 grader
	Siemens (20).jpg	65 grader	22,1 grader

Datorgenererade bilder

Tabell 6

Perspektivförvrägning	bildnamn		Uppmätt vinkelförändring för streckkoden
horisontell	H 04.jpg		2.8 grader
	H 06.jpg		4.8 grader
	H 08.jpg		6.3 grader
	H 10.jpg		7.8 grader
	H 12.jpg		9.4 grader
	H 14.jpg		11.1 grader
	H 16.jpg		12.7 grader
	H 18.jpg		14.2 grader
	H 20.jpg		16.0 grader
	H 22.jpg		17.6 grader
	H 24.jpg		19.3 grader
	H 26.jpg		21.0 grader

Del 3 - Vertikal perspektivförvrängning



Figur 3 - Vinkel

Experiment bilder

Tabell 7

Nokia	bildnamn	X vinkelinställning på ställningen	Uppmätt vinkelförändring för streckkoden
perspektivförvrägning	Nokia (21).jpg	10 grader	4.5 grader
vertikal	Nokia (22).jpg	15 grader	5.3 grader
upplösning 1152 x 864	Nokia (23).jpg	20 grader	6.4 grader
avstånd 15 cm	Nokia (24).jpg	25 grader	7.9 grader
	Nokia (25).jpg	30 grader	9.0 grader
	Nokia (26).jpg	35 grader	10.4 grader
	Nokia (27).jpg	40 grader	11.9 grader
	Nokia (28).jpg	45 grader	13.6 grader
	Nokia (29).jpg	50 grader	15.2 grader

Tabell 8

Siemens	bildnamn	X vinkelinställning på ställningen	Uppmätt vinkelförändring för streckkoden
perspektivförvrägning	Siemens (21).jpg	10 grader	2,5 grader
vertikal	Siemens (22).jpg	15 grader	3,2 grader
upplösning 640 x 480	Siemens (23).jpg	20 grader	3,8 grader
avstånd 15 cm	Siemens (24).jpg	25 grader	4,6 grader
	Siemens (25).jpg	30 grader	5,4 grader
	Siemens (26).jpg	35 grader	6,3 grader
	Siemens (27).jpg	40 grader	9,3 grader
	Siemens (28).jpg	45 grader	10,9 grader
	Siemens (29).jpg	50 grader	13,1 grader

Datorgenererade bilder

Tabell 9

perspektivförvrägning	bildnamn		Uppmätt vinkelförändring för streckkoden
vertikal	V 04.jpg		2.4 grader
	V 06.jpg		3.6 grader
	V 08.jpg		4.8 grader
	V 10.jpg		6.1 grader
	V 12.jpg		7.3 grader
	V 14.jpg		8.4 grader
	V 16.jpg		9.7 grader
	V 18.jpg		11.0 grader

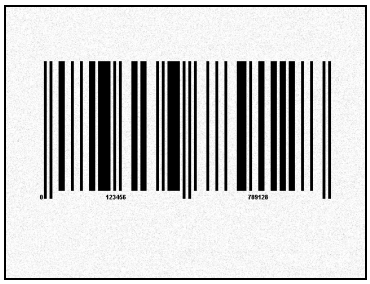
Bilaga J - Datorgenererade EAN13 bilder från metoden

Brus


G 05 %.jpg

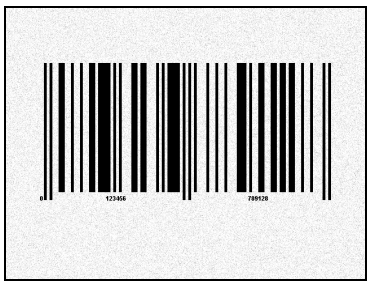

G 06 %.jpg

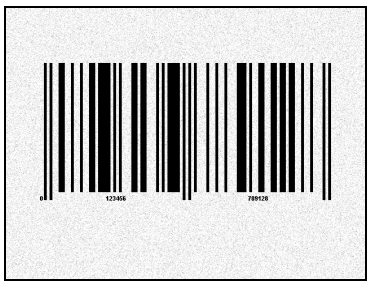

G 07 %.jpg

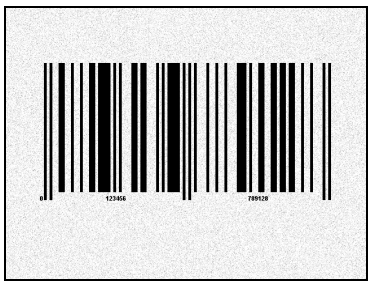

G 08 %.jpg

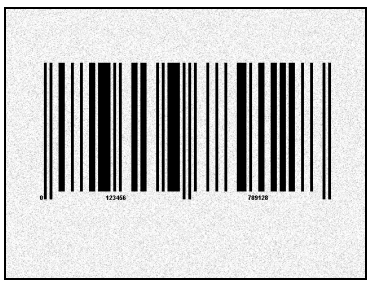

G 09 %.jpg

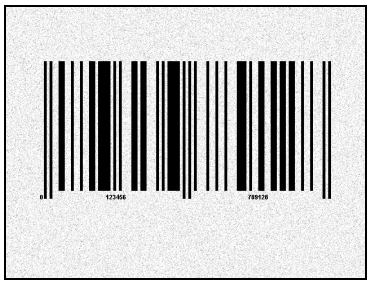

G 10 %.jpg


G 11 %.jpg

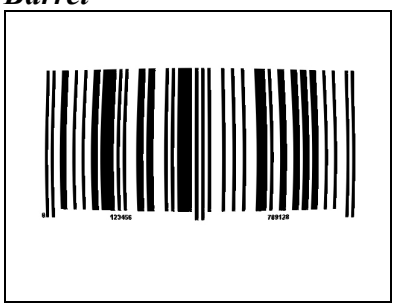

G 12 %.jpg

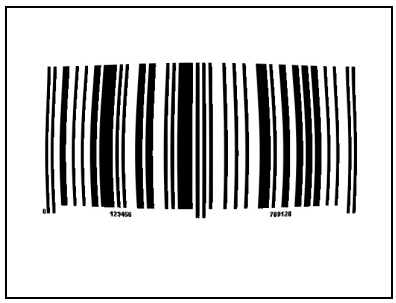

G 13 %.jpg

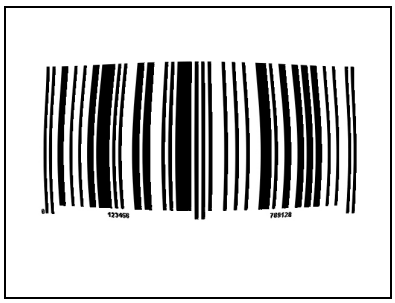

G 14 %.jpg

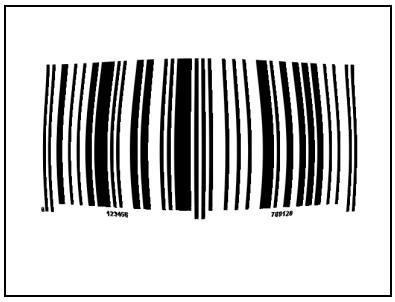

G 15 %.jpg

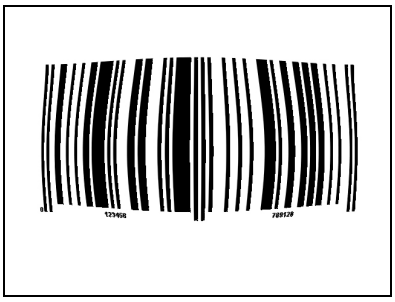
Barrel

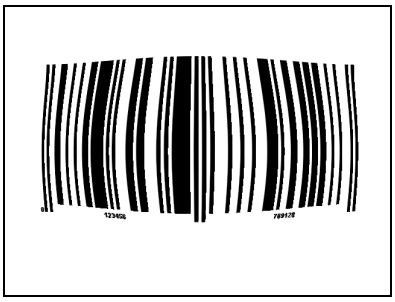

B 10.jpg

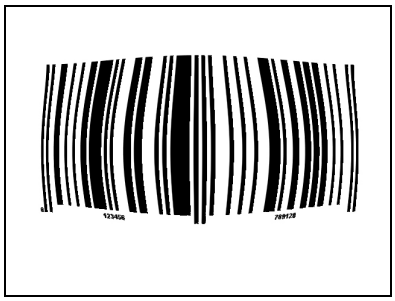

B 15.jpg

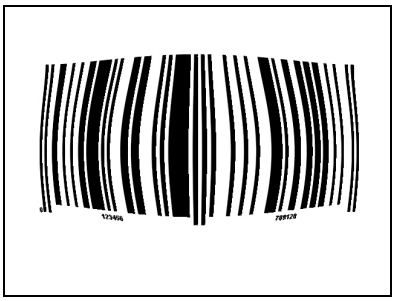

B 20.jpg

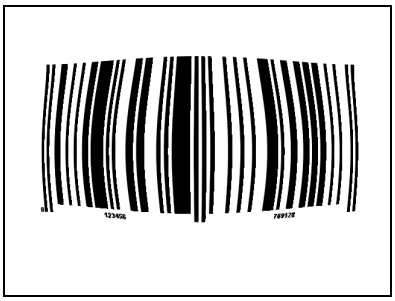

B 25.jpg

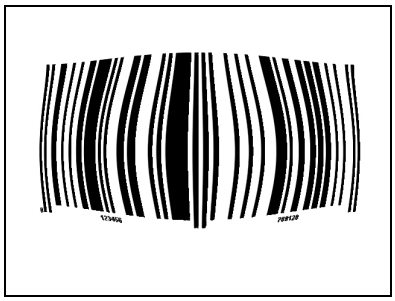

B 30.jpg

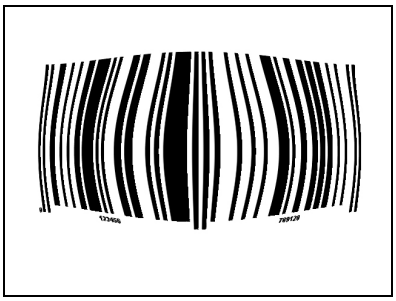

B 35.jpg

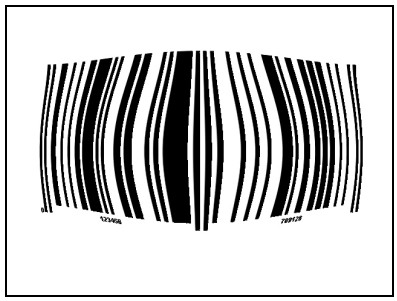

B 40.jpg

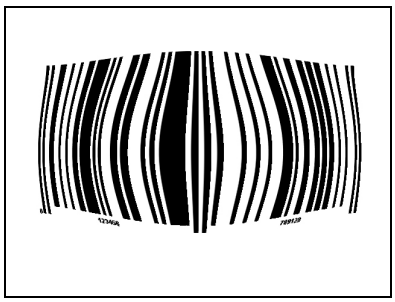

B 45.jpg

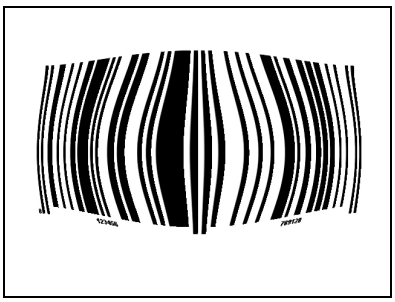

B 50.jpg

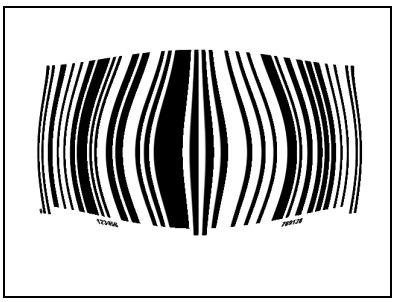

B 55.jpg


B 60.jpg


B 65.jpg

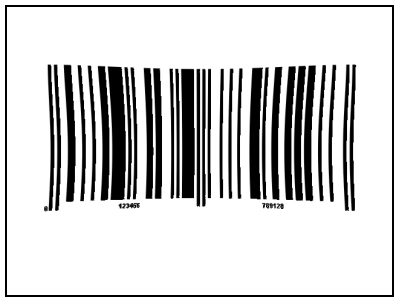

B 70.jpg

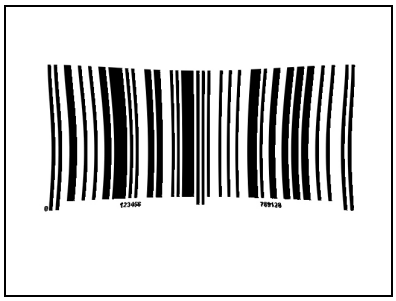

B 75.jpg

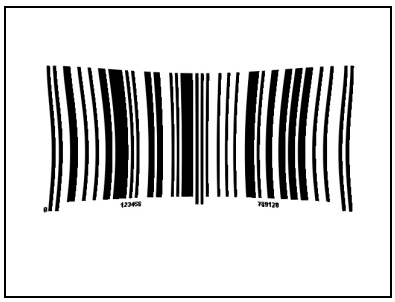

B 80.jpg

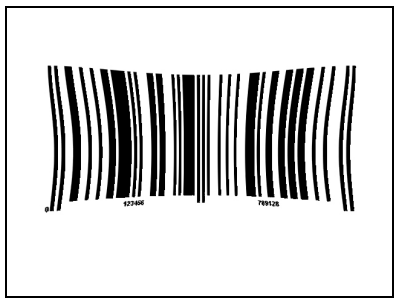
Pincushion

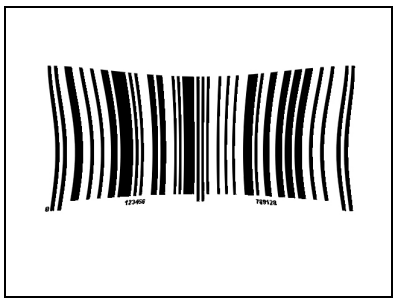

P 10.jpg

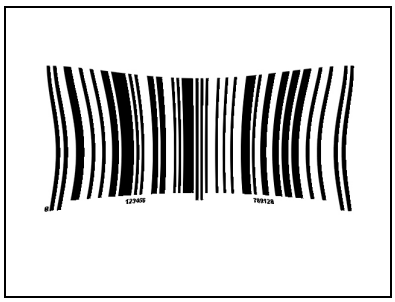

P 15.jpg

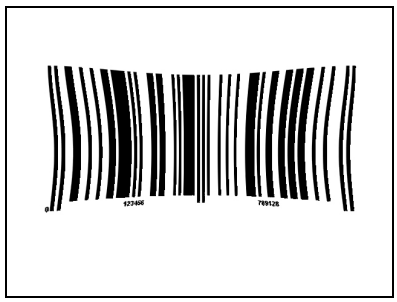

P 20.jpg

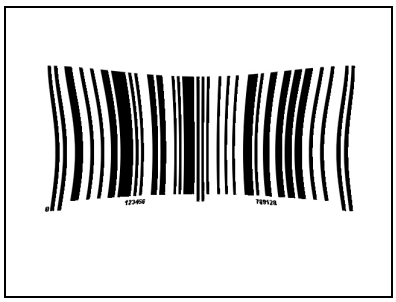

P 25.jpg

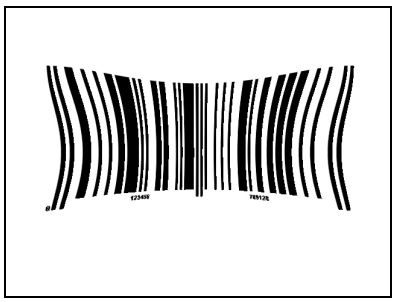

P 30.jpg

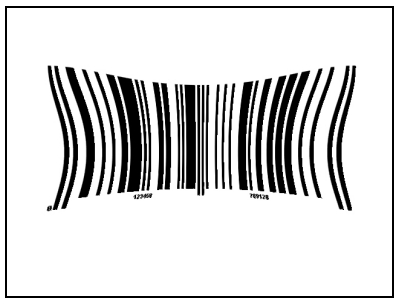

P 35.jpg

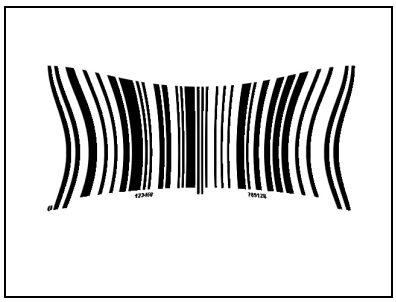

P 40.jpg

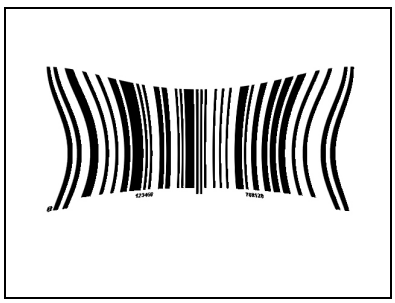

P 45.jpg

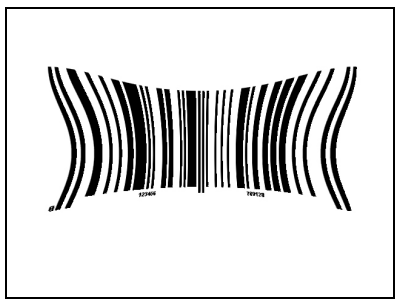

P 50.jpg

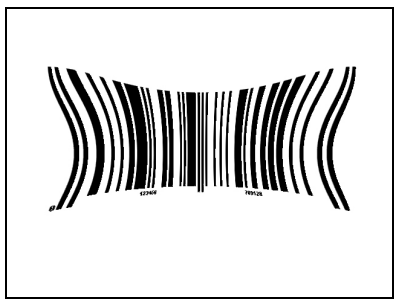

P 55.jpg


P 60.jpg


P 65.jpg


P 70.jpg


P 75.jpg


P 80.jpg

Kontrast


10 %.jpg


15 %.jpg


20 %.jpg


25 %.jpg


30 %.jpg


35 %.jpg


40 %.jpg


45 %.jpg


50 %.jpg


55 %.jpg


60 %.jpg

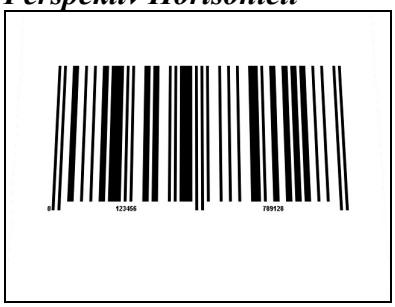

65 %.jpg

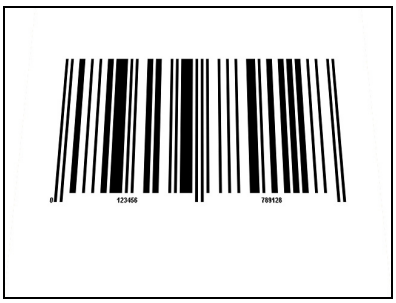

70 %.jpg

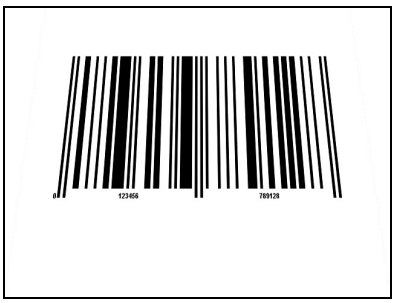

75 %.jpg

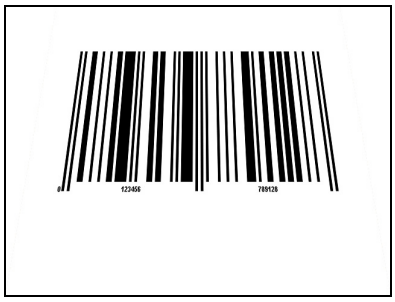

80 %.jpg

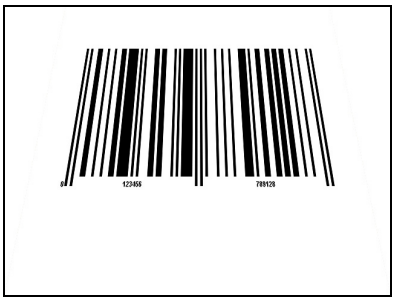
Perspektiv Horisontell

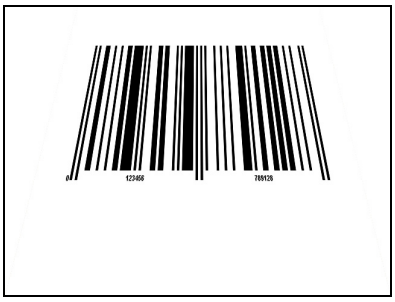

H 04.jpg

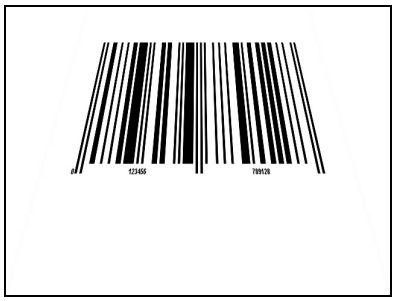

H 06.jpg

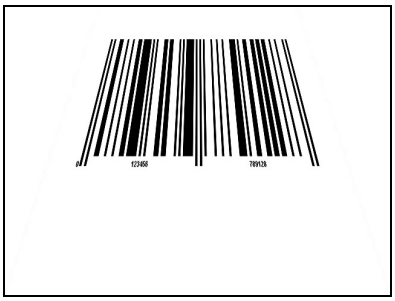

H 08.jpg

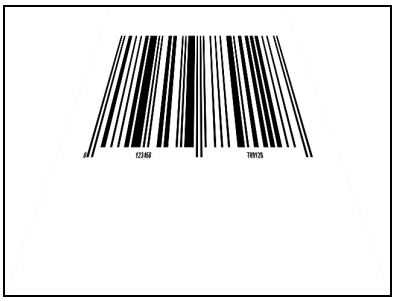

H 10.jpg

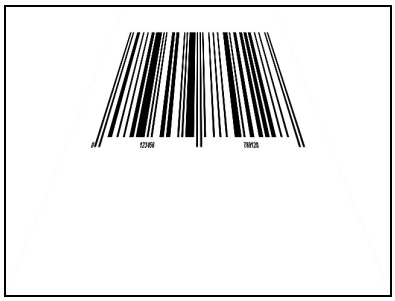

H 12.jpg

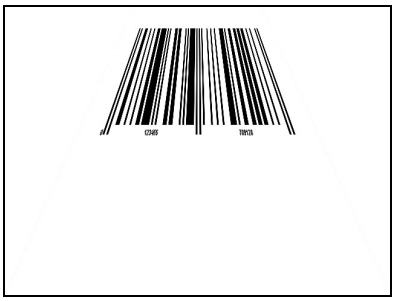

H 14.jpg

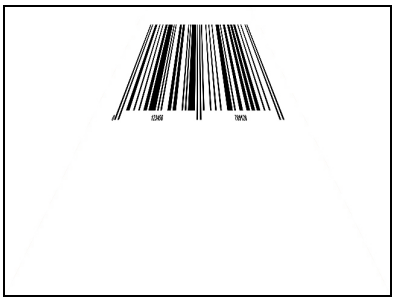

H 16.jpg


H 18.jpg


H 20.jpg

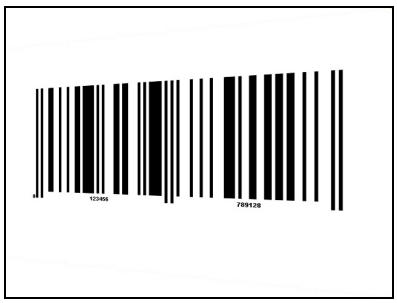

H 22.jpg

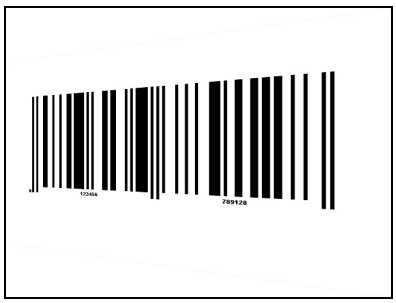

H 24.jpg

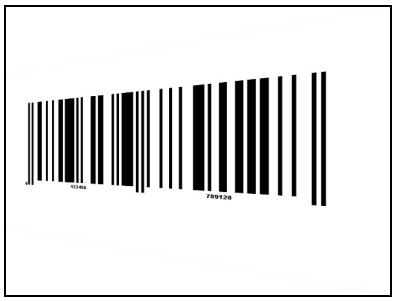

H 26.jpg

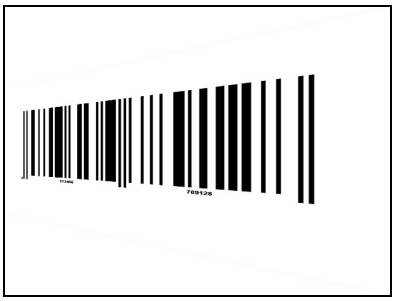
Perspektiv Vertikal

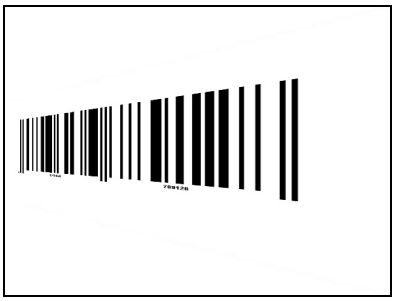

V 04.jpg

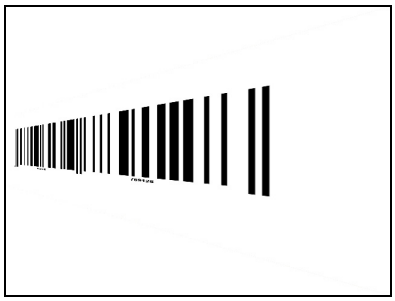

V 06.jpg

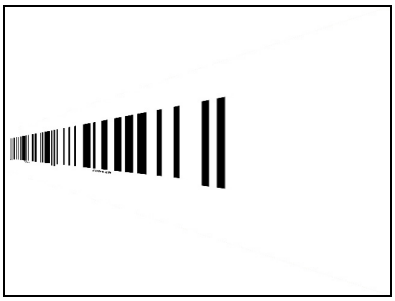

V 08.jpg


V 10.jpg


V 12.jpg


V 14.jpg


V 16.jpg


V 18.jpg

Rörelseoskärpa


01 pix.jpg


02 pix.jpg


03 pix.jpg


04 pix.jpg


05 pix.jpg


06 pix.jpg


07 pix.jpg


08 pix.jpg


10 pix.jpg


12 pix.jpg


14 pix.jpg

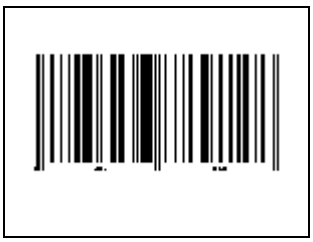

16 pix.jpg


18 pix.jpg

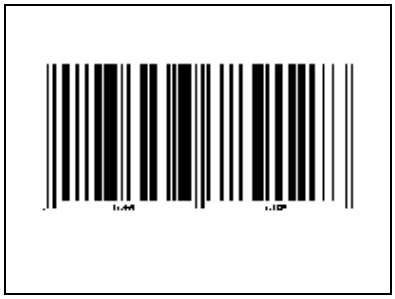
Upplösning - gif

086x65.gif

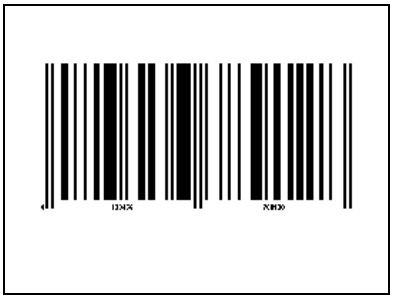
114x86.gif



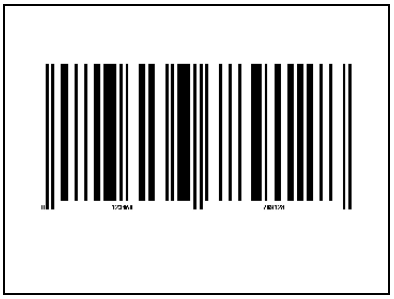
152x114.gif



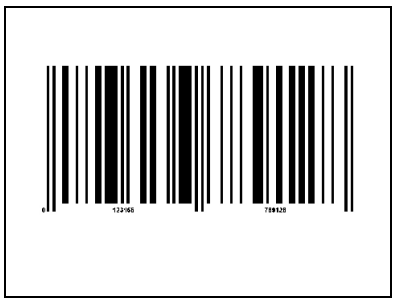
203x152.gif



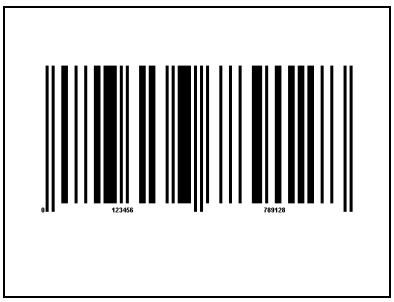
270x203.gif



360x270.gif



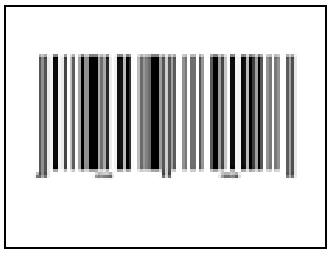
480x360.gif



640x480.gif

Upplösning - jpg

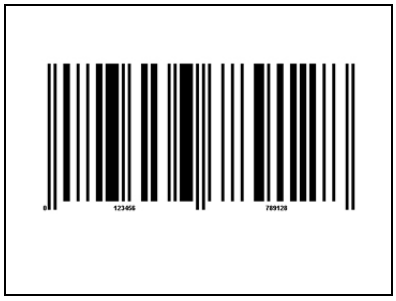

086x65.jpg

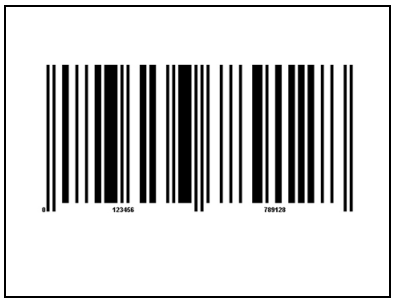

114x86.jpg

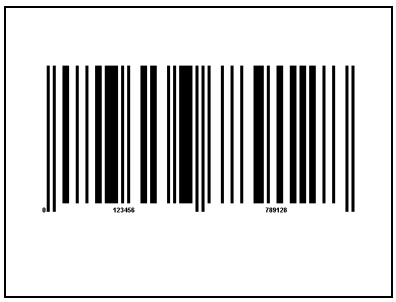

153x114.jpg


203x152.jpg


270x203.jpg

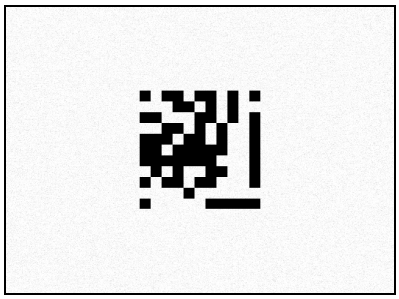

360x270.jpg

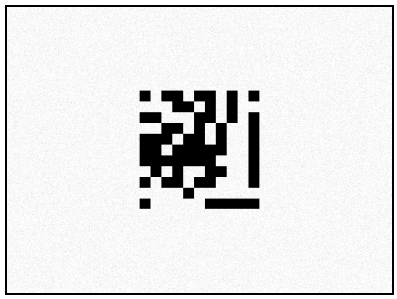

480x360.jpg

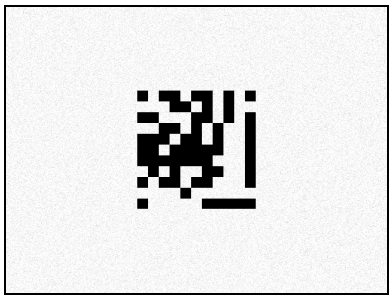

640x480.jpg

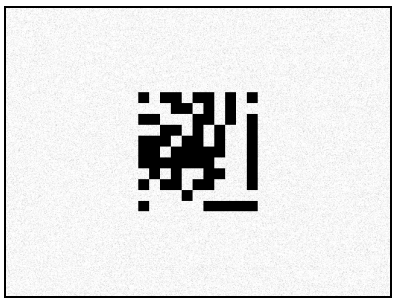
Bilaga K - Datorgenererade Visual Code bilder från metoden

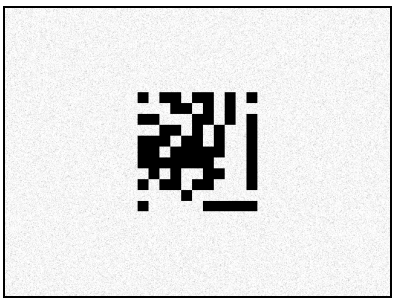
Brus

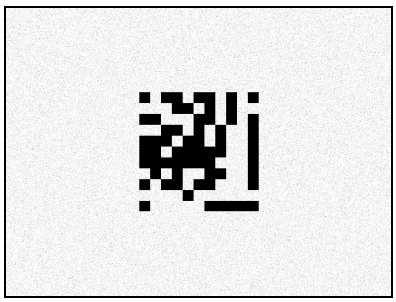

G 05 %.jpg

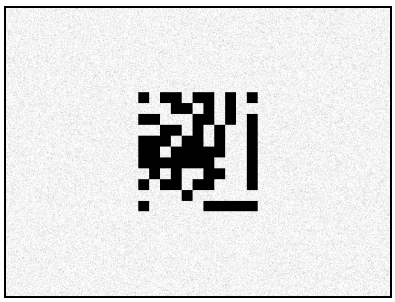

G 06 %.jpg

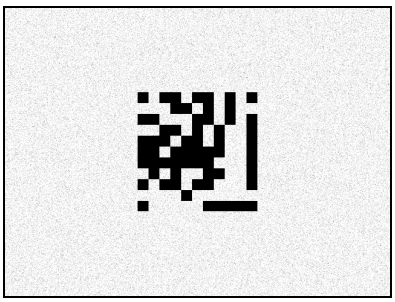

G 07 %.jpg

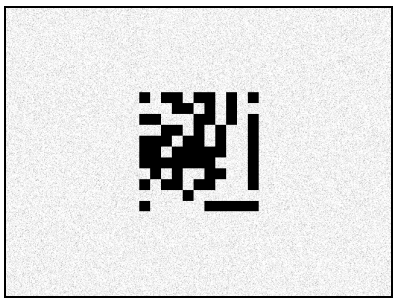

G 08 %.jpg

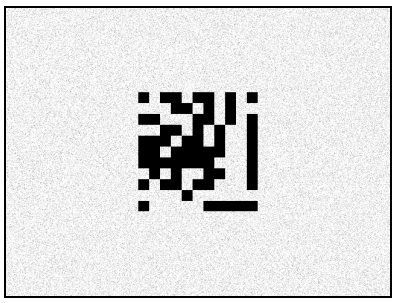

G 09 %.jpg

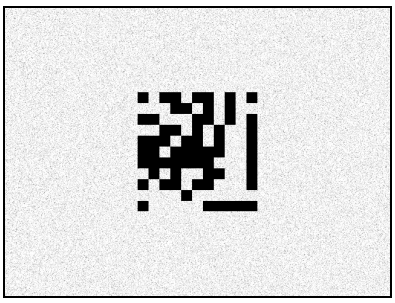

G 10 %.jpg


G 11 %.jpg

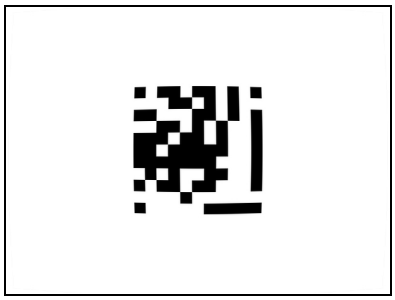

G 12 %.jpg

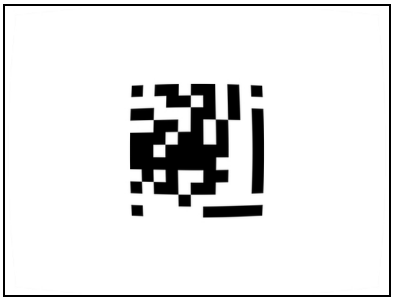

G 13 %.jpg

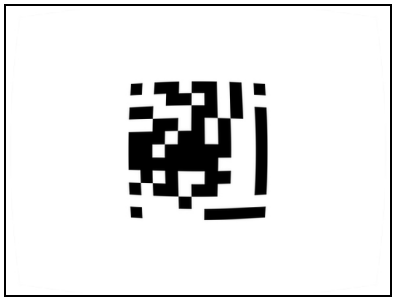

G 14 %.jpg

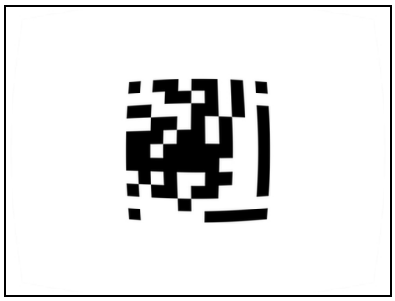

G 15 %.jpg

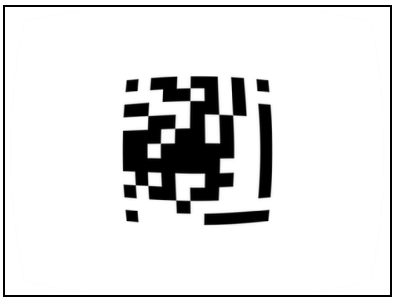
Barrel

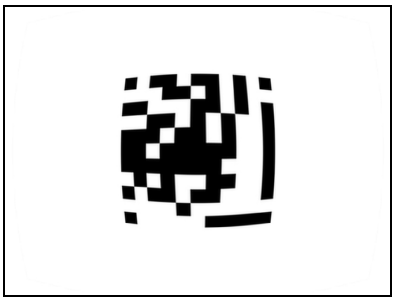

B 10.jpg

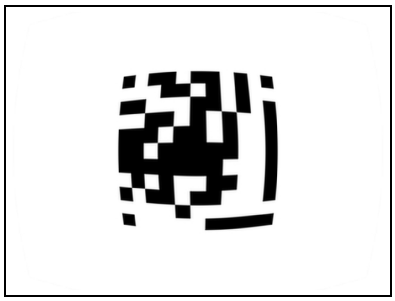

B 15.jpg

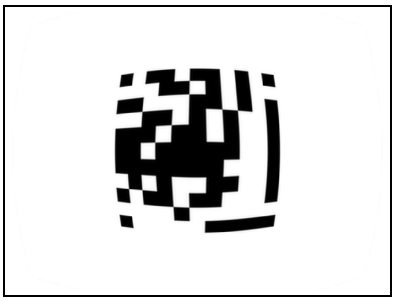

B 20.jpg

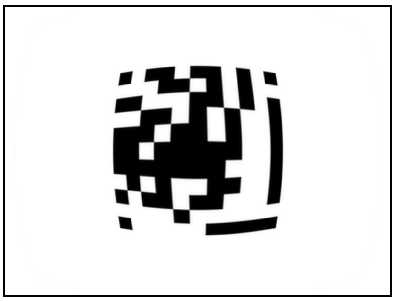

B 25.jpg

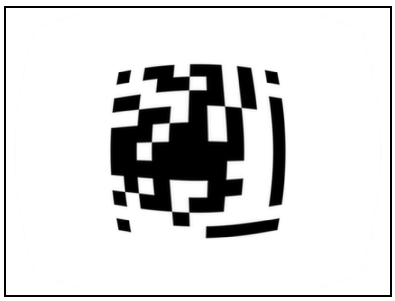

B 30.jpg

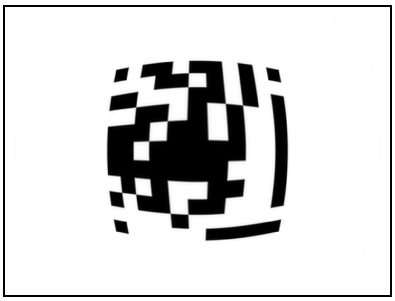

B 35.jpg

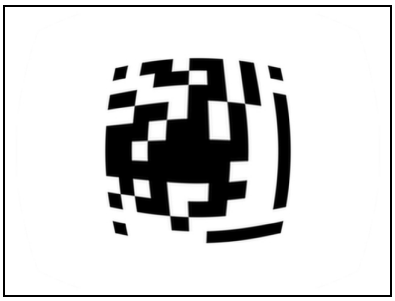

B 40.jpg

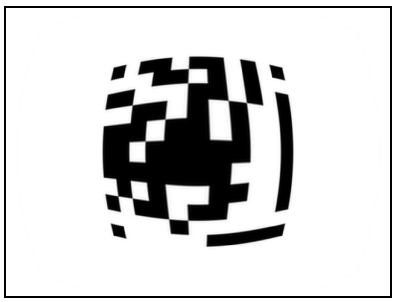

B 45.jpg

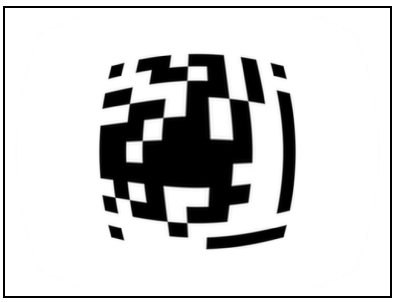

B 50.jpg

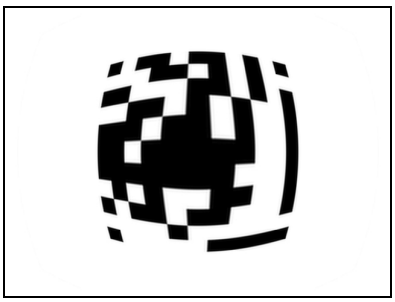

B 55.jpg


B 60.jpg

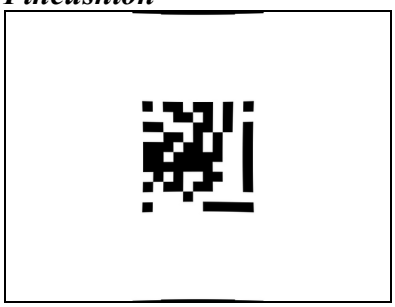

B 65.jpg

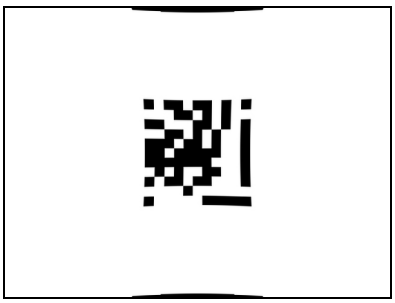

B 70.jpg

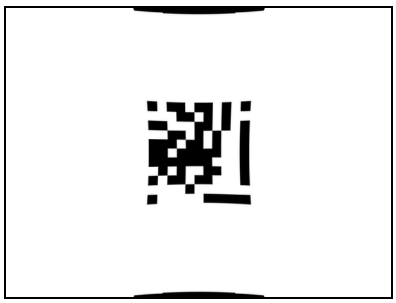

B 75.jpg

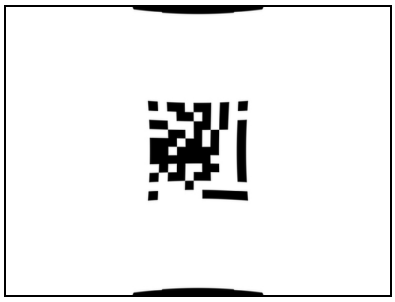

B 80.jpg

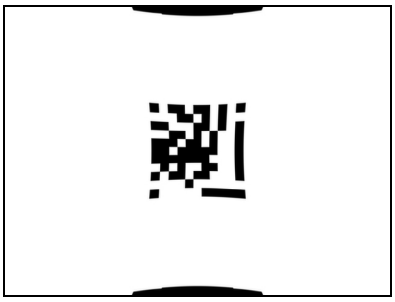
Pincushion

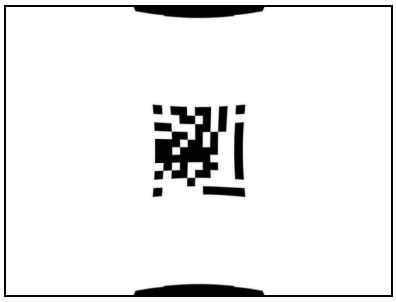

P 10.jpg

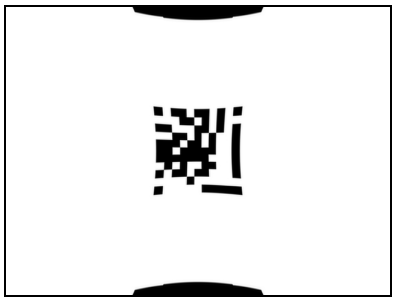

P 15.jpg

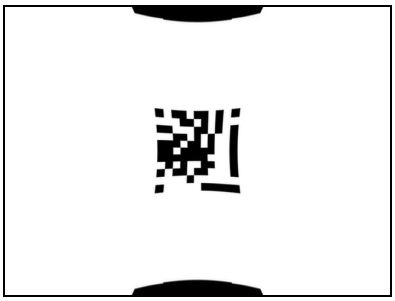

P 20.jpg

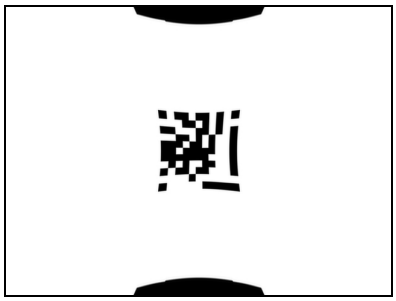

P 25.jpg

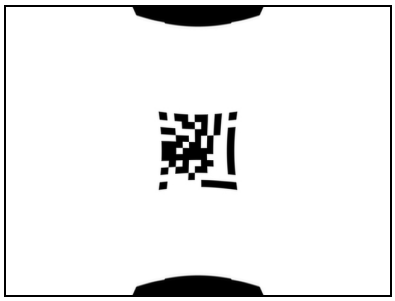

P 30.jpg

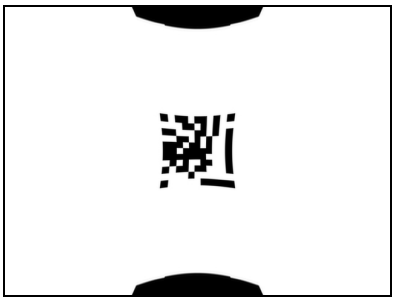

P 35.jpg

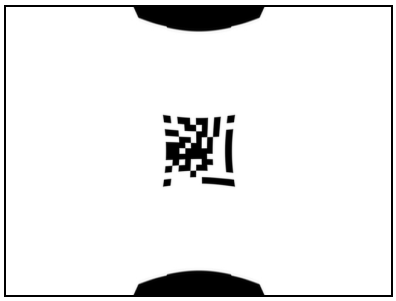

P 40.jpg

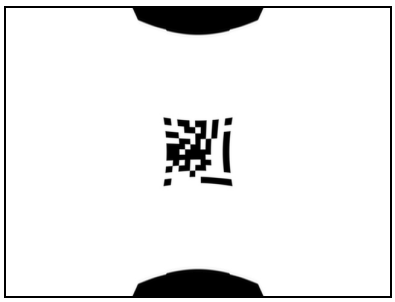

P 45.jpg

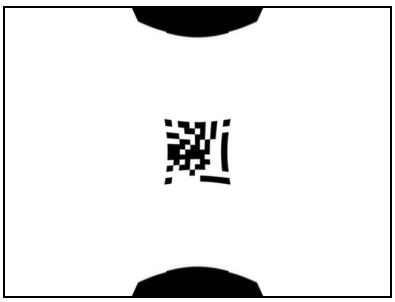

P 50.jpg

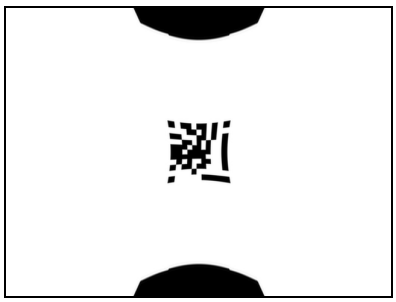

P 55.jpg


P 60.jpg

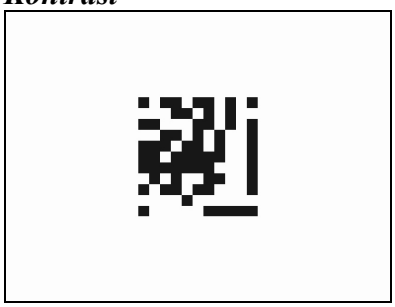

P 65.jpg

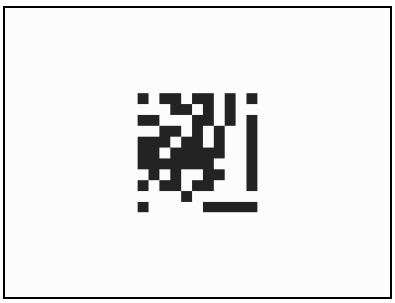

P 70.jpg

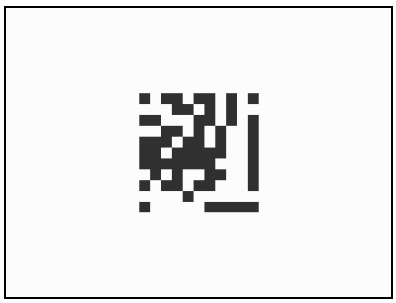

P 75.jpg

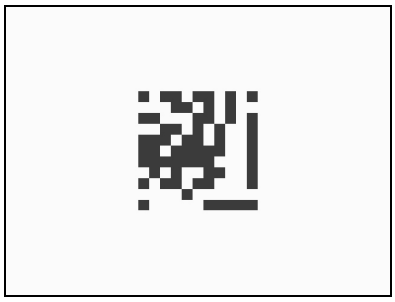

P 80.jpg

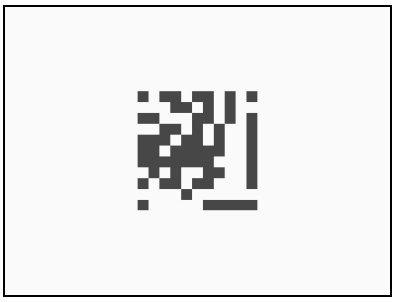
Kontrast

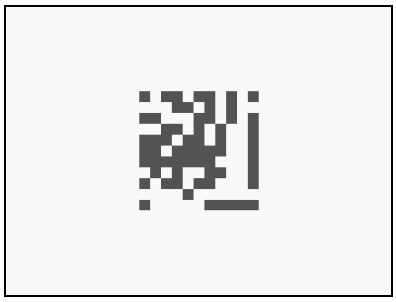

10 %.jpg

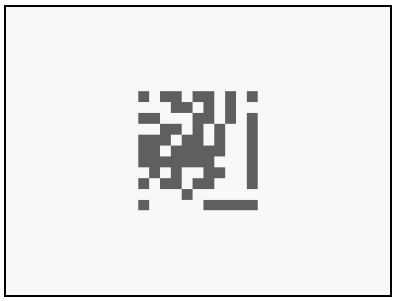

15 %.jpg

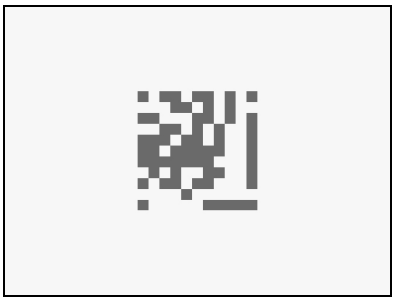

20 %.jpg

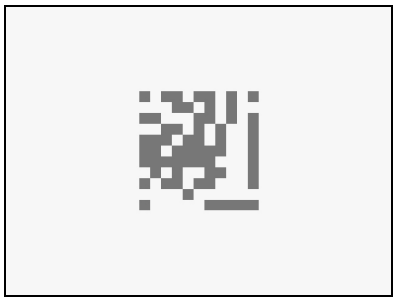

25 %.jpg

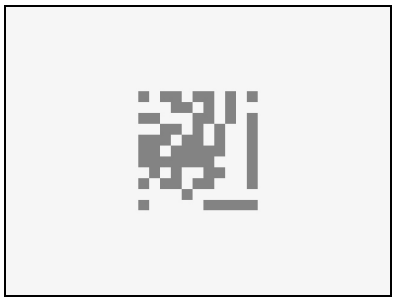

30 %.jpg

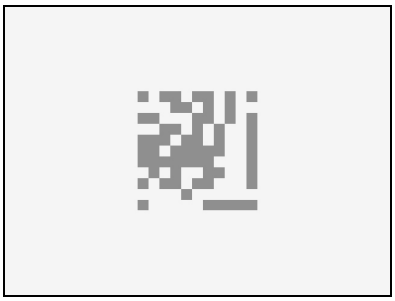

35 %.jpg

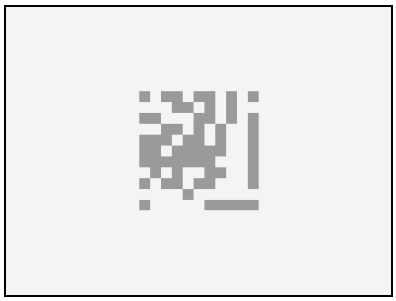

40 %.jpg

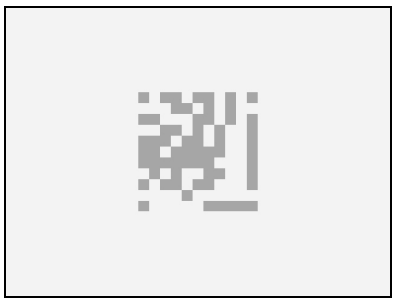

45 %.jpg

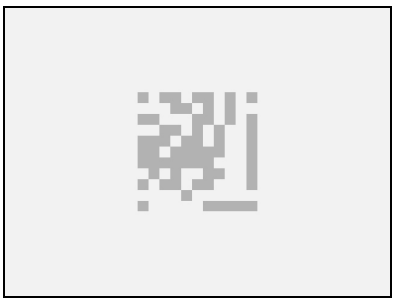

50 %.jpg

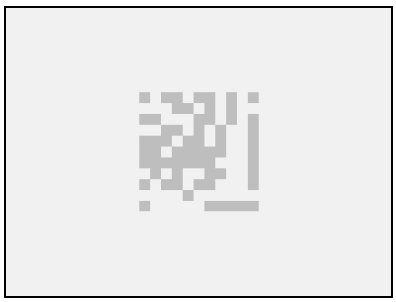

55 %.jpg


60 %.jpg

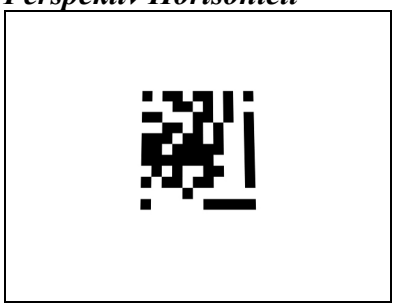

65 %.jpg

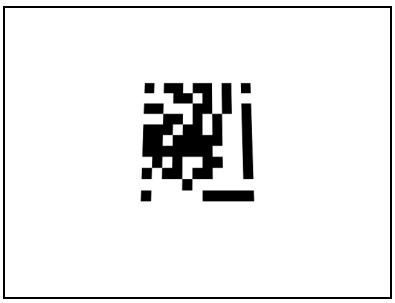

70 %.jpg

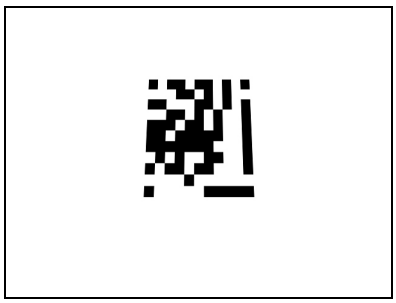

75 %.jpg

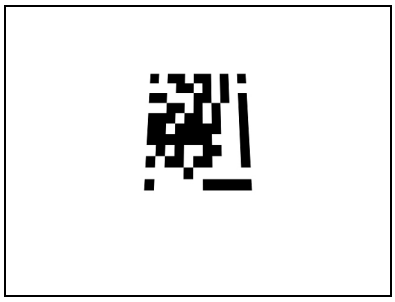

80 %.jpg

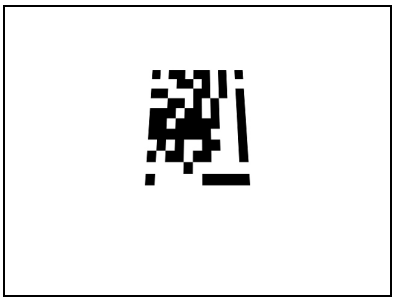
Perspektiv Horisontell

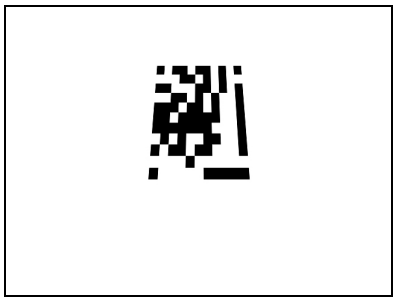

H 04.jpg

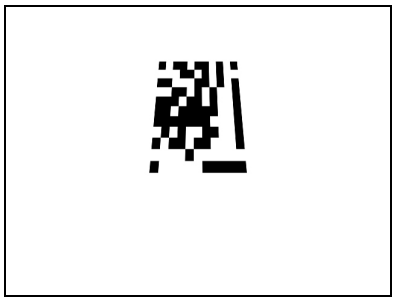

H 06.jpg

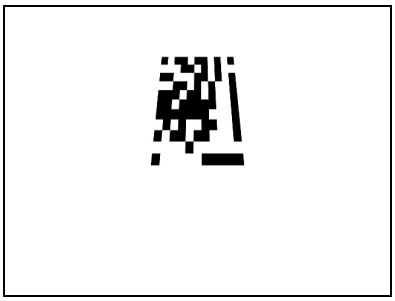

H 08.jpg

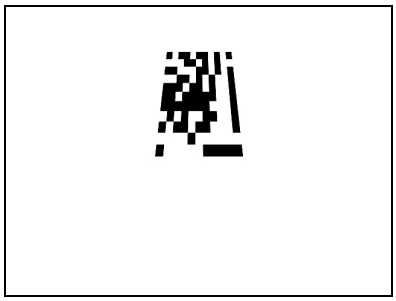

H 10.jpg

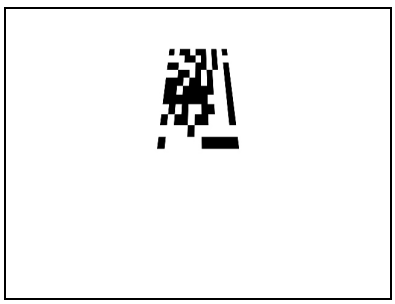

H 12.jpg

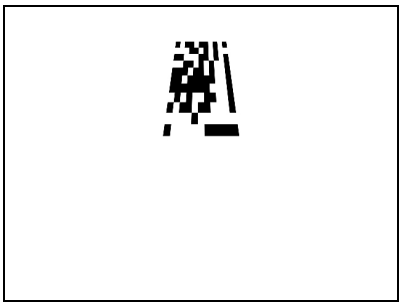

H 14.jpg

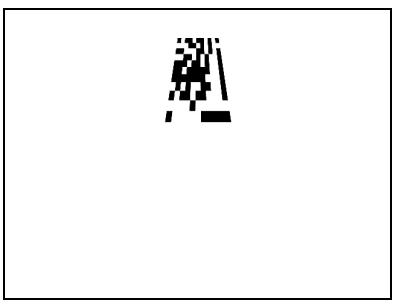

H 16.jpg


H 18.jpg

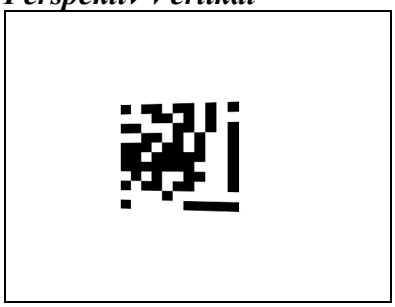

H 20.jpg

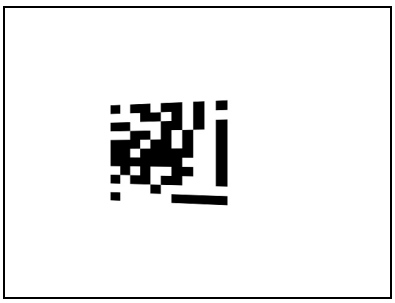

H 22.jpg

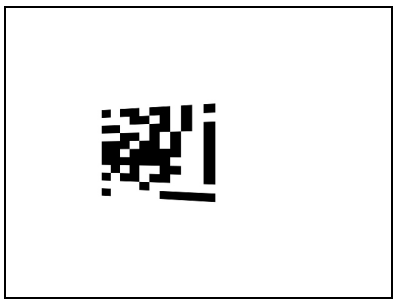

H 24.jpg

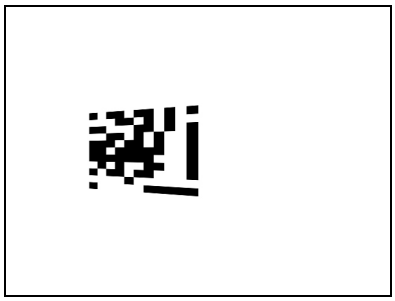

H 26.jpg

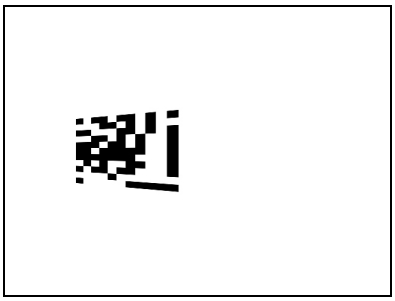
Perspektiv Vertikal

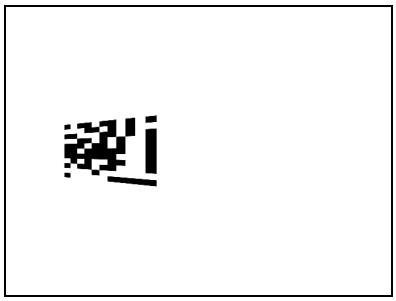

V 04.jpg

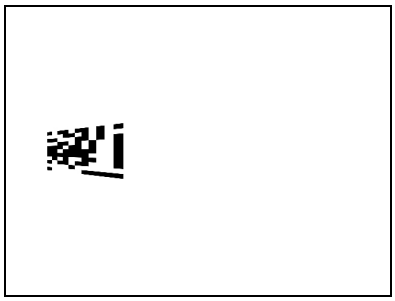

V 06.jpg

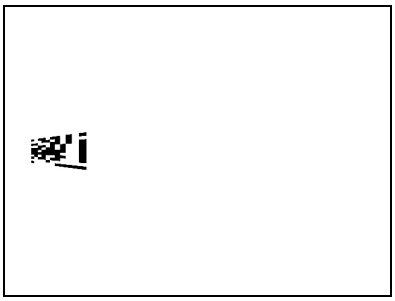

V 08.jpg


V 10.jpg

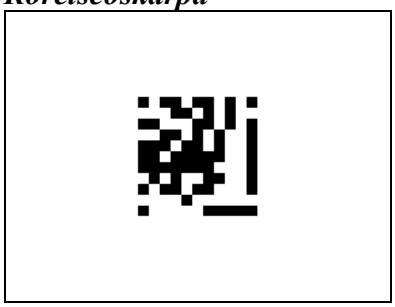

V 12.jpg

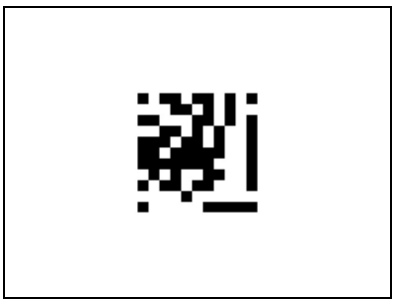

V 14.jpg

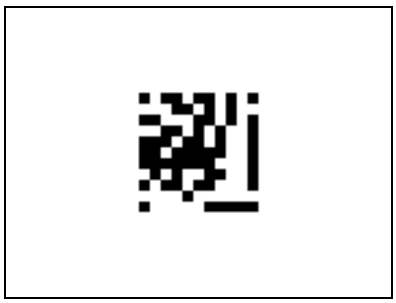

V 16.jpg

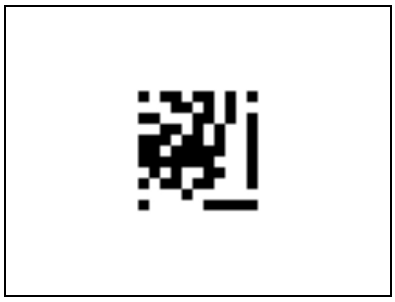

V 18.jpg

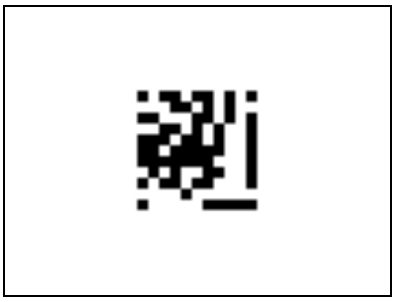
Rörelseoskärpa

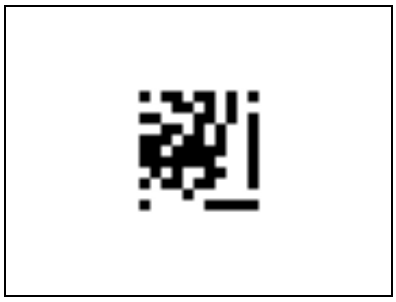

01 pix.jpg

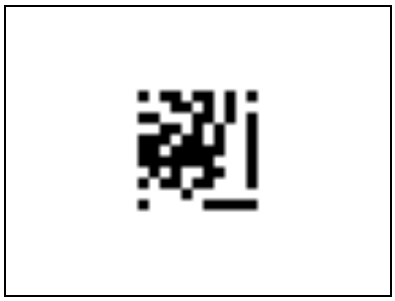

02 pix.jpg


03 pix.jpg


04 pix.jpg


05 pix.jpg


06 pix.jpg


07 pix.jpg


08 pix.jpg


10 pix.jpg


12 pix.jpg


14 pix.jpg

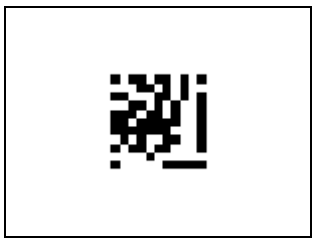

16 pix.jpg

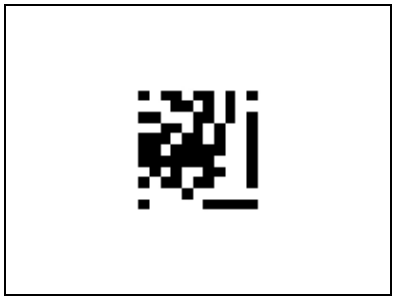

18 pix.jpg

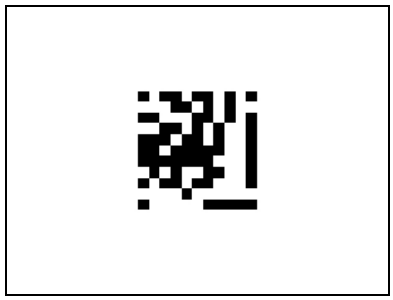
Upplösning - gif

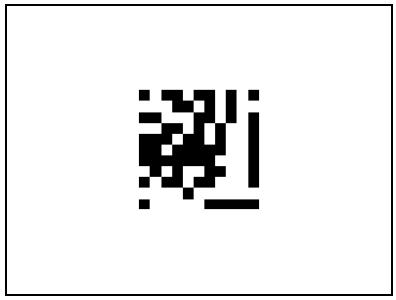

086x65.gif

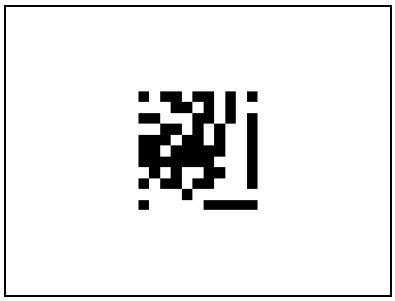

114x86.gif

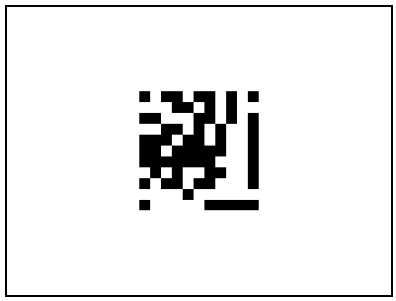

152x114.gif


203x152.gif


270x203.gif

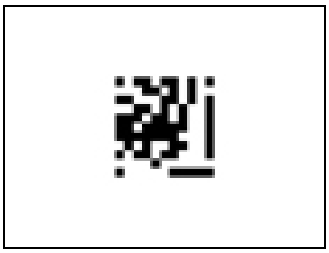

360x270.gif

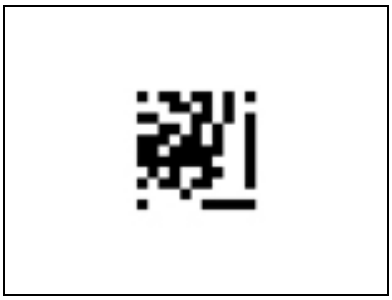

480x360.gif

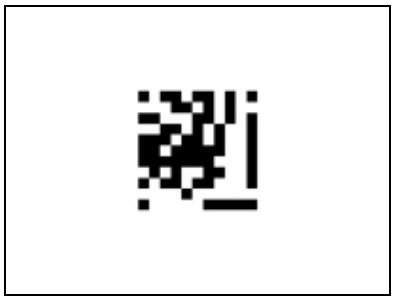

640x480.gif

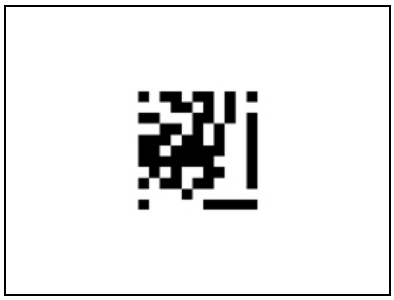
Upplösning - jpg

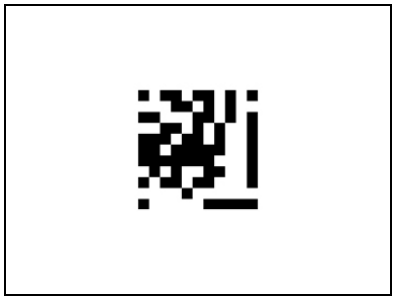

086x65.jpg

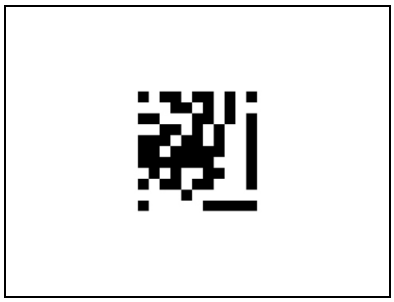

114x86.jpg

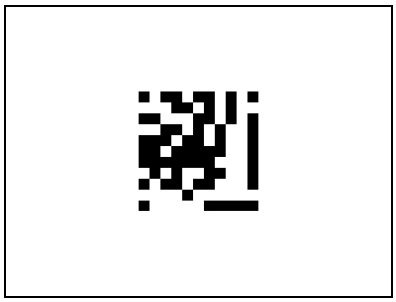

153x114.jpg


203x152.jpg


270x203.jpg


360x270.jpg


480x360.jpg


640x480.jpg

Bilaga L - Resultat metoden EAN 13

Korrekt avkodning enligt algoritmen har skett då ”checksum = true” samt ”resultat = 123456789128”.

Algoritm 1 - dBarScanJ

Tabell 1

C:\metoden\brus		
filnamn	**checksum**	**resultat**
G 05 %.jpg	true	123456789128
G 06 %.jpg	true	123456789128
G 07 %.jpg	true	123456789128
G 08 %.jpg	true	123456789128
G 09 %.jpg	true	123456789128
G 10 %.jpg	true	123456789128
G 11 %.jpg	true	123456789128
G 12 %.jpg	true	123456789128
G 13 %.jpg	true	123456789128
G 14 %.jpg	true	123456789128
G 15 %.jpg	true	123456789128

C:\metoden\distortion\barrel		
filnamn	**checksum**	**resultat**
B 10.jpg	true	123456789128
B 15.jpg	true	123456789128
B 20.jpg	true	123456789128
B 25.jpg	true	123456789128
B 30.jpg	true	123456789128
B 35.jpg	true	123456789128
B 40.jpg	true	123456789128
B 45.jpg	false	0
B 50.jpg	false	0
B 55.jpg	false	0
B 60.jpg	false	0
B 65.jpg	false	0
B 70.jpg	false	0
B 75.jpg	false	0
B 80.jpg	false	0

C:\metoden\distortion\pincushion		
filnamn	**checksum**	**resultat**
P 10.jpg	true	123456789128
P 15.jpg	true	123456789128
P 20.jpg	true	123456789128
P 25.jpg	true	123456789128
P 30.jpg	true	123456789128
P 35.jpg	true	123456789128
P 40.jpg	true	123456789128

P 45.jpg	true	123456789128
P 50.jpg	true	123456789128
P 55.jpg	true	123456789128
P 60.jpg	true	123456789128
P 65.jpg	true	123456789128
P 70.jpg	true	123456789128
P 75.jpg	true	123456789128
P 80.jpg	true	123456789128

C:\metoden\kontrast		
filnamn	**checksum**	**resultat**
10 %.jpg	true	123456789128
15 %.jpg	true	123456789128
20 %.jpg	true	123456789128
25 %.jpg	true	123456789128
30 %.jpg	true	123456789128
35 %.jpg	true	123456789128
40 %.jpg	true	123456789128
45 %.jpg	true	123456789128
50 %.jpg	true	123456789128
55 %.jpg	false	0
60 %.jpg	false	0
65 %.jpg	false	0
70 %.jpg	false	0
75 %.jpg	false	0
80 %.jpg	false	0

C:\metoden\perspektiv\horisontell		
filnamn	**checksum**	**resultat**
H 04.jpg	true	123456789128
H 06.jpg	true	123456789128
H 08.jpg	true	123456789128
H 10.jpg	true	123456789128
H 12.jpg	true	123456789128
H 14.jpg	true	123456789128
H 16.jpg	true	123456789128
H 18.jpg	true	123456789128
H 20.jpg	true	123456789128
H 22.jpg	false	0
H 24.jpg	false	0
H 26.jpg	false	0

C:\metoden\perspektiv\vertikal		
filnamn	**checksum**	**resultat**
V 04.jpg	true	123456789128
V 06.jpg	true	123456789128
V 08.jpg	true	123456789128
V 10.jpg	true	123456789128
V 12.jpg	true	123456789128
V 14.jpg	false	0

V 16.jpg	false	0
V 18.jpg	false	0

C:\metoden\rörelseoskärpa		
filnamn	**checksum**	**resultat**
01 pix.jpg	true	123456789128
02 pix.jpg	true	123456789128
03 pix.jpg	true	123456789128
04 pix.jpg	true	123456789128
05 pix.jpg	true	123456789128
06 pix.jpg	true	123456789128
07 pix.jpg	true	123456789128
08 pix.jpg	true	123456789128
10 pix.jpg	false	0
12 pix.jpg	false	0
14 pix.jpg	false	0
16 pix.jpg	false	0
18 pix.jpg	false	0

C:\metoden\upplösning\gif		
filnamn	**checksum**	**resultat**
086x65.gif	false	0
114x86.gif	false	0
152x114.gif	false	0
203x152.gif	false	0
270x203.gif	true	123456789129
360x270.gif	true	123456789128
480x360.gif	true	123456789128
640x480.gif	true	123456789128

C:\metoden\upplösning\jpg		
filnamn	**checksum**	**resultat**
086x65.jpg	false	0
114x86.jpg	false	0
153x114.jpg	false	0
203x152.jpg	false	0
270x203.jpg	false	0
360x270.jpg	true	123456789128
480x360.jpg	true	123456789128
640x480.jpg	true	123456789128

Algoritm 2 - Papier-Mâché

Tabell 2

C:\metoden\brus		
filnamn	**checksum**	**resultat**
G 05 %.jpg	true	123456789128
G 06 %.jpg	true	123456789128
G 07 %.jpg	true	123456789128
G 08 %.jpg	true	123456789128
G 09 %.jpg	true	123456789128
G 10 %.jpg	true	123456789128
G 11 %.jpg	true	123456789128
G 12 %.jpg	true	123456789128
G 13 %.jpg	true	123456789128
G 14 %.jpg	true	123456789128
G 15 %.jpg	true	123456789128

C:\metoden\distortion\barrel		
filnamn	**checksum**	**resultat**
B 10.jpg	true	123456789128
B 15.jpg	true	123456789128
B 20.jpg	true	123456789128
B 25.jpg	true	123456789128
B 30.jpg	true	123456789128
B 35.jpg	true	123456789128
B 40.jpg	true	123456789128
B 45.jpg	true	123456789128
B 50.jpg	true	123456789128
B 55.jpg	true	123456789128
B 60.jpg	true	123456789128
B 65.jpg	true	123456789128
B 70.jpg	true	123456789128
B 75.jpg	true	123456789128
B 80.jpg	true	123456789128

C:\metoden\distortion\pincushion		
filnamn	**checksum**	**resultat**
P 10.jpg	true	123456789128
P 15.jpg	true	123456789128
P 20.jpg	true	123456789128
P 25.jpg	true	123456789128
P 30.jpg	true	123456789128
P 35.jpg	true	123456789128
P 40.jpg	true	123456789128
P 45.jpg	true	123456789128
P 50.jpg	true	123456789128
P 55.jpg	true	123456789128
P 60.jpg	true	123456789128
P 65.jpg	true	123456789128
P 70.jpg	true	123456789128

P 75.jpg	true	123456789128
P 80.jpg	true	123456789128

C:\metoden\kontrast		
filnamn	**checksum**	**resultat**
10 %.jpg	true	123456789128
15 %.jpg	true	123456789128
20 %.jpg	true	123456789128
25 %.jpg	true	123456789128
30 %.jpg	true	123456789128
35 %.jpg	true	123456789128
40 %.jpg	true	123456789128
45 %.jpg	true	123456789128
50 %.jpg	true	123456789128
55 %.jpg	false	0
60 %.jpg	false	0
65 %.jpg	false	0
70 %.jpg	false	0
75 %.jpg	false	0
80 %.jpg	false	0

C:\metoden\perspektiv\horisontell		
filnamn	**checksum**	**resultat**
H 04.jpg	true	123456789128
H 06.jpg	true	123456789128
H 08.jpg	true	123456789128
H 10.jpg	true	123456789128
H 12.jpg	true	123456789128
H 14.jpg	true	123456789128
H 16.jpg	true	123456789128
H 18.jpg	true	123456789128
H 20.jpg	true	123456789128
H 22.jpg	true	123456789128
H 24.jpg	true	123456789128
H 26.jpg	true	123456789128

C:\metoden\perspektiv\vertikal		
filnamn	**checksum**	**resultat**
V 04.jpg	true	123456789128
V 06.jpg	true	123456789128
V 08.jpg	true	123456789128
V 10.jpg	true	123456789128
V 12.jpg	true	123456789128
V 14.jpg	true	123456789128
V 16.jpg	false	0
V 18.jpg	false	0

C:\metoden\rörelseoskärpa		
filnamn	**checksum**	**resultat**
01 pix.jpg	true	123456789128
02 pix.jpg	true	123456789128
03 pix.jpg	true	123456789128
04 pix.jpg	true	123456789128
05 pix.jpg	true	123456789128
06 pix.jpg	false	0
07 pix.jpg	false	0
08 pix.jpg	false	0
10 pix.jpg	false	0
12 pix.jpg	false	0
14 pix.jpg	false	0
16 pix.jpg	false	0
18 pix.jpg	false	0

C:\metoden\upplösning\gif		
filnamn	**checksum**	**resultat**
086x65.gif	false	0
114x86.gif	false	0
152x114.gif	false	0
203x152.gif	false	0
270x203.gif	false	0
360x270.gif	true	123456789128
480x360.gif	true	123456789128
640x480.gif	true	123456789128

C:\metoden\upplösning\jpg		
filnamn	**checksum**	**resultat**
086x65.jpg	false	0
114x86.jpg	false	0
153x114.jpg	false	0
203x152.jpg	false	0
270x203.jpg	true	123456789128
360x270.jpg	true	123456789128
480x360.jpg	true	123456789128
640x480.jpg	true	123456789128

Sammanfattning

Algoritm 1 - dBarScanJ

Tabell 3

Kategori	Del	Intervall		Antal bilder	Algoritm 1	
		avkodade bilder	*ej avkodade bilder*		*avkodade*	*procent*
Brus		G 05%.jpg - G 15%.jpg	-	11	11	100%
Distortion	Barrel	B 10.jpg - B 40.jpg	B 45.jpg - B 80.jpg	15	7	47%
	Pincushion	P 10.jpg - P 80.jpg	-	15	15	100%
Kontrast		10%.jpg - 50%.jpg	55 % - 80 %.jpg	15	9	60%
Perspektiv	Horisontell	H 04.jpg - H 20.jpg	H 22.jpg - H 26.jpg	12	9	75%
	Vertikal	V 04.jpg - V 12.jpg	V 14.jpg - V 18.jpg	8	5	63%
Rörelseoskärpa		01pix.jpg - 08pix.jpg	10pix.jpg - 18pix.jpg	13	8	62%
Upplösning	Jpg	640x480.jpg - 360x270.jpg	270x203.jpg - 86x65.jpg	8	3	38%
	Gif	640x480.gif - 360x270.gif	270x203.gif - 86x65.gif	8	3	38%
	totalt:			105	70	67%

Algoritm 2 - Papier-Mâché

Tabell 4

Kategori	Del	Intervall		Antal bilder	Algoritm 2	
		avkodade bilder	*ej avkodade bilder*		*avkodade*	*procent*
Brus		G 05%.jpg - G 15%.jpg	-	11	11	100%
Distortion	Barrel	B 10.jpg - B 80.jpg	-	15	15	100%
	Pincushion	P 10.jpg - P 80.jpg	-	15	15	100%
Kontrast		10%.jpg - 50%.jpg	55 %.jpg - 80 %.jpg	15	9	60%
Perspektiv	Horisontell	H 04.jpg - H 26.jpg	-	12	12	100%
	Vertikal	V 04.jpg - V 14.jpg	V 16.jpg - V 18.jpg	8	6	75%
Rörelseoskärpa		01pix.jpg - 05pix.jpg	06pix.jpg - 18pix.jpg	13	5	38%
Upplösning	Jpg	640x480.jpg - 270x203.jpg	203x152.jpg - 86x65.jpg	8	3	38%
	Gif	640x480.gif - 360x270.gif	270x203.gif - 86x65.gif	8	4	50%
	totalt:			105	80	76%

Bilaga M - Resultat metoden Visual Code

Korrekt avkodning enligt algoritmen har skett då ”checksum = true” samt ”resultat = 9199990000000000000000”.

Algoritm 3 - ETH

Tabell 1

C:\metoden\brus		
filnamn	**checksum**	**resultat**
G 05 %.jpg	true	9199990000000000000000
G 06 %.jpg	true	9199990000000000000000
G 07 %.jpg	true	9199990000000000000000
G 08 %.jpg	true	9199990000000000000000
G 09 %.jpg	true	9199990000000000000000
G 10 %.jpg	true	9199990000000000000000
G 11 %.jpg	true	9199990000000000000000
G 12 %.jpg	false	787866347537688000000000
G 13 %.jpg	false	0
G 14 %.jpg	false	0
G 15 %.jpg	false	636750620086959000000000

C:\metoden\distortion\barrel		
filnamn	**checksum**	**resultat**
B 10.jpg	true	9199990000000000000000
B 15.jpg	true	9199990000000000000000
B 20.jpg	true	9199990000000000000000
B 25.jpg	true	9199990000000000000000
B 30.jpg	true	9199990000000000000000
B 35.jpg	true	9199990000000000000000
B 40.jpg	true	9199990000000000000000
B 45.jpg	false	5510232830970840000000000
B 50.jpg	false	4896325332312970000000000
B 55.jpg	false	4896473482726310000000000
B 60.jpg	false	5198704937630250000000000
B 65.jpg	false	5274115227403590000000000
B 70.jpg	false	5274114074490880000000000
B 75.jpg	false	0
B 80.jpg	false	0

C:\metoden\distortion\pincushion		
filnamn	**checksum**	**resultat**
P 10.jpg	true	9199990000000000000000
P 15.jpg	true	9199990000000000000000
P 20.jpg	true	9199990000000000000000
P 25.jpg	true	9199990000000000000000
P 30.jpg	true	9199990000000000000000
P 35.jpg	true	9199990000000000000000
P 40.jpg	true	9199990000000000000000
P 45.jpg	true	9199990000000000000000

P 50.jpg	true	9199990000000000000000
P 55.jpg	true	9199990000000000000000
P 60.jpg	true	9199990000000000000000
P 65.jpg	true	9199990000000000000000
P 70.jpg	true	9199990000000000000000
P 75.jpg	false	6521440240912530000000000
P 80.jpg	false	5312514421297900000000000

C:\metoden\kontrast		
filnamn	**checksum**	**resultat**
10 %.jpg	true	9199990000000000000000
15 %.jpg	true	9199990000000000000000
20 %.jpg	true	9199990000000000000000
25 %.jpg	true	9199990000000000000000
30 %.jpg	true	9199990000000000000000
35 %.jpg	true	9199990000000000000000
40 %.jpg	true	9199990000000000000000
45 %.jpg	true	9199990000000000000000
50 %.jpg	true	9199990000000000000000
55 %.jpg	true	9199990000000000000000
60 %.jpg	true	9199990000000000000000
65 %.jpg	true	9199990000000000000000
70 %.jpg	false	636750593064262000000000
75 %.jpg	false	0
80 %.jpg	false	0

C:\metoden\perspektiv\horisontell		
filnamn	**checksum**	**resultat**
H 04.jpg	true	9199990000000000000000
H 06.jpg	true	9199990000000000000000
H 08.jpg	true	9199990000000000000000
H 10.jpg	true	9199990000000000000000
H 12.jpg	true	9199990000000000000000
H 14.jpg	true	9199990000000000000000
H 16.jpg	true	9199990000000000000000
H 18.jpg	false	6367506189687730000000000
H 20.jpg	false	0
H 22.jpg	false	0
H 24.jpg	false	0
H 26.jpg	false	0

C:\metoden\perspektiv\vertikal		
filnamn	**checksum**	**resultat**
V 04.jpg	true	9199990000000000000000
V 06.jpg	true	9199990000000000000000
V 08.jpg	true	9199990000000000000000
V 10.jpg	true	9199990000000000000000
V 12.jpg	true	9199990000000000000000
V 14.jpg	false	0

V 16.jpg	false	0
V 18.jpg	false	0

C:\metoden\rörelseoskärpa		
filnamn	**checksum**	**resultat**
01 pix.jpg	true	9199990000000000000000
02 pix.jpg	true	9199990000000000000000
03 pix.jpg	true	9199990000000000000000
04 pix.jpg	true	9199990000000000000000
05 pix.jpg	true	9199990000000000000000
06 pix.jpg	true	9199990000000000000000
07 pix.jpg	true	9199990000000000000000
08 pix.jpg	true	9199990000000000000000
10 pix.jpg	true	9199990000000000000000
12 pix.jpg	true	9199990000000000000000
14 pix.jpg	true	9199990000000000000000
16 pix.jpg	true	9199990000000000000000
18 pix.jpg	true	9199990000000000000000

C:\metoden\upplösning\gif		
filnamn	**checksum**	**resultat**
086x65.gif	false	0
114x86.gif	true	9199990000000000000000
152x114.gif	true	9199990000000000000000
203x152.gif	true	9199990000000000000000
270x203.gif	true	9199990000000000000000
360x270.gif	true	9199990000000000000000
480x360.gif	true	9199990000000000000000
640x480.gif	true	9199990000000000000000

C:\metoden\upplösning\jpg		
filnamn	**checksum**	**resultat**
086x65.jpg	true	9199990000000000000000
114x86.jpg	true	9199990000000000000000
153x114.jpg	true	9199990000000000000000
203x152.jpg	true	9199990000000000000000
270x203.jpg	true	9199990000000000000000
360x270.jpg	true	9199990000000000000000
480x360.jpg	true	9199990000000000000000
640x480.jpg	true	9199990000000000000000

Sammanfattning

Algoritm 3 - ETH

Tabell 2

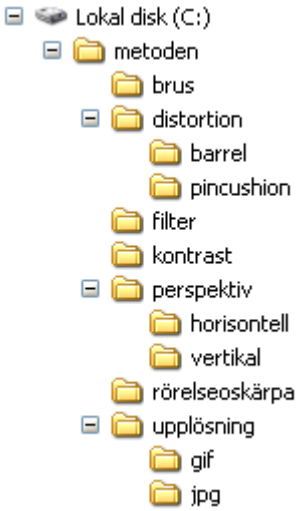
Kategori	Del	Intervall		Antal bilder	Algoritm 3	
		avkodade bilder	*ej avkodade bilder*		*avkodade*	*procent*
Brus		G 05%.jpg - G 11%.jpg	G 12%.jpg - G 15%.jpg	11	7	64%
Distortion	Barrel	B 10.jpg - B 40.jpg	B 45.jpg - B 80.jpg	15	7	47%
	Pincushion	P 10.jpg - P 70.jpg	P 75.jpg - P 80.jpg	15	13	87%
Kontrast		10%.jpg - 65%.jpg	65 % - 80 %.jpg	15	12	80%
Perspektiv	Horisontell	H 04.jpg - H 16.jpg	H 18.jpg - H 26.jpg	12	7	58%
	Vertikal	V 04.jpg - V 12.jpg	V 14.jpg - V 18.jpg	8	5	63%
Rörelseoskärpa		01pix.jpg - 18pix.jpg	-	13	13	100%
Upplösning	Jpg	640x480.jpg - 86x65.jpg	-	8	7	88%
	Gif	640x480.gif - 114x86.gif	86x65.gif	8	8	100%
	totalt:			105	79	75%

Bilaga N - Lathund för metoden

Denna lathund är en guide för att utföra metoden som presenteras i rapporten ovan. Lathunden är uppdelad i 6 antal steg som bör utföras i sekvens ordning med start i steg 1.

Steg 1:

Steg 1 utgörs av att skapa en katalogstruktur. Katalogerna används för att separera de kategorier av bilder som skapas i steg 3 av lathunden. Katalogstrukturen kan dels skapas manuellt, den skall då skapas så att den exakt efterliknar strukturen i figuren nedan.

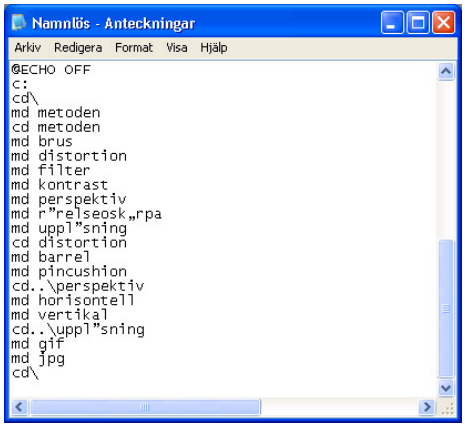


Figur 1

Katalogstrukturen kan även skapas automatiskt genom att exekvera ett makro eller en bat-fil i Windows som finns i bilaga E. Bat-filen finns att hämta på cd-skivan som följer med rapporten under root:\metoden\bat\ eller via hemsidan http://home.swipnet.se/~w-155814. För att skapa bat-filen med hjälp av bilaga E utför steg 1a, annars hoppa till steg 1b.

Steg 1a

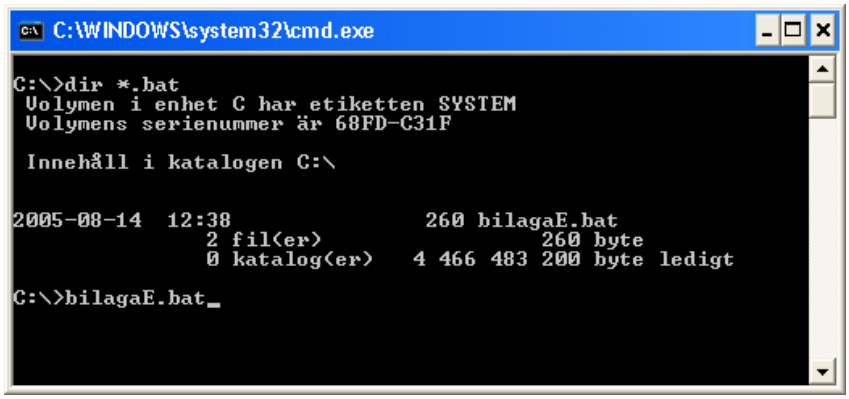
Öppna programmet anteckningar eller liknande textredigeringsprogram. Kopiera texten Från bilaga E in i anteckningar och spara filen med ändelsen bat. Ex: ”bilagaE.bat”. Resultatet bör likna bilden nedan.



Figur 2

Steg 1b

För att exekvera bat-filen öppna en DOS kommando prompt. I Windows XP välj Start - kör... I öppna fältet skriva ”cmd” och öppna för att starta DOS. Kör bat-filen genom att skriva dess namn och tryck därefter ’Enter’. Katalogstruktur enligt ovan skapas då på C:. Bat filen kan exekveras från vilket ställe på datorn som helst.

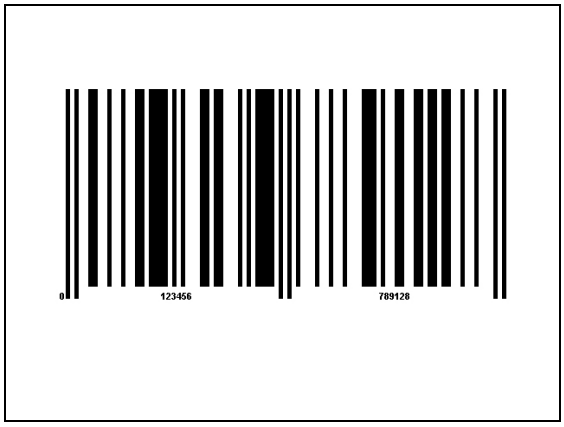


Figur 3

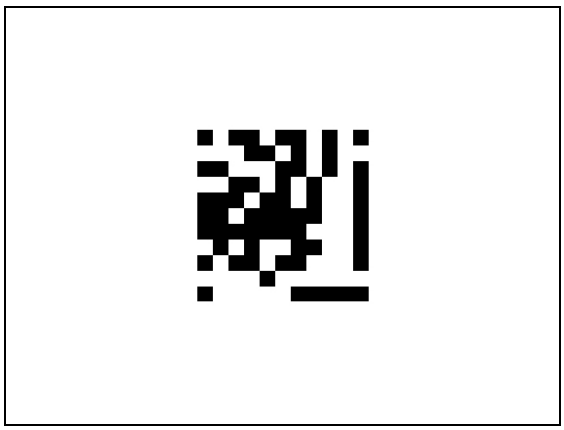
Steg 2

För att generera alla kategorier av bilder behövs en originalbild, en ursprunglig bild som övriga bilder använder som bas. Denna bild bestämmer du som användare själv beroende på typ av streckkod som skall testas. Exempel bild finns på cd-skivan under root:\metoden\ursprungligbild\ eller via http://home.swipnet.se/~w-155814.

Två exempel bilder, EAN13 och Visual Code.



Figur 4



Figur 5

Steg 3

Efter att katalogstrukturen har genererats på C: och en originalbild har valts ut är nästa steg att generera bilder. Bilderna skapas med hjälp av droplets (se kapitel 4), observera att droplets endast funderar om Photoshop finns installerat på datorn.

Bilderna skapas i nio olika kategorier med hjälp av nio olika droplets. Bilderna kan genereras i vilken ordning som helst, dropletfilerna är dock numrerade från 1 till 9 och bilderna skapas med fördel i den ordningen. Dropletfilerna listas nedan. Dropletfilerna som används i denna metoden finns att hitta på cd-skivan under root:\metoden\droplets\ eller via http://home.swipnet.se/~w-155814.

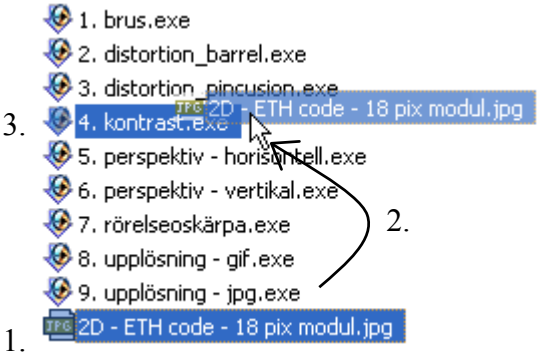
Figur 6

Dropletfilerna bör ligga i samma katalog som originalbilden, detta för att förenkla det bildskapande förloppet. För att skapa bilderna gör följande:

Steg 3a

För att generera bilderna i kategorin *brus*, klicka på och markera originalfilen - den bild som övriga bilder använder som ursprung – bilden som togs fram vid steg 2. Dra och släpp sedan den markerade filen över dropletfilen som heter **'1. brus.exe'**. När bildfilen släpps över dropletfilen startas Photoshop om det inte redan körs. I Photoshop utförs flera olika steg för att generera bilderna. I detta läge behövs ingen aktiv medverkan förutsagt att alla steg i denna guide har gjorts. 11 olika bilder genereras i kategorin brus och bilderna sparas i katalogen 'brus' under 'c:\metoden'. Katalogen måste finnas för att ett lyckat resultat ska nås, detta gäller för alla kategorier nedan.

Demonstrationsexempel.
1. markera originalbilden, i detta fallet "2D - ETH code - 18 pix modul.jpg".
2. Dra den markerade filen till rätt dropletfil, i detta fallet "4. kontrast.exe".
3. Släpp den markerade originalbilden på dropletfilen.
4. Efter att bildfilen släppt över dropletfilen startar den automatiska genereringen av bilderna i kategorin och sparas i motsvarande katalog i c:\metoden.



Figur 7

Generera bilder för kategorin *distortion - barrel* genom att klicka på och markera original filen samt därefter dra och släppa filen över dropleten som heter **'2. distortion - barrel.exe'**. 15 bilder genereras då i kategorin och bilderna sparas i katalogen 'c:\metoden\distortion\barrel'.

Bilder till de övriga kategorierna genereras på liknande sätt som för de två första.

Generera bilder för kategorin *distortion - pincusion* med hjälp av droplet **'3. distortion - pincusion.exe'**. 15 bilder genereras i kategorin och bilderna sparas i katalogen 'c:\metoden\distortion\pincushion'.

För kategorin *kontrast*, dra och släpp originalbilden på droplet **'4. kontrast.exe'**. 15 bilder skapas och läggs i katalogen 'c:\metoden\kontrasten'.

För kategorin *perspektiv - horisontell* använd dropletfilen '**5. perspektiv - horisontell.exe**'. 12 bilder sparas i katalogen 'c:\metoden\perspektiv\horisontell'.

Bilderna till kategorin *perspektiv - vertikal* skapas med hjälp av dropletfilen **'6. perspektiv - vertikal.exe'**. 8 bilder genereras i denna kategori och sparas i katalogen 'c:\metoden\perspektiv\vertikal'.

Nästa kategori att skapa bilder för är *rörelseoskärpa*. Dra och släpp samma originalbild som för kategorier ovan på dropletfilen **'7. rörelseoskärpa.exe'** för att generera bilder till kategorin. 13 bilder genereras som placeras i 'c:\metoden\rörelseoskärpa'.

Kategori *upplösning - gif* är näst sista kategori att generera bilder för. Klicka på och markera originalfilen, dra och släpp därefter filen över dropleten som heter **'8. upplösning – gif.exe'**. 8 bilder skapas i katalogen 'c:\metoden\upplösning\gif'.

Sista kategorin att generera bilder för är *upplösning - jpg*. Dropletfilen som skall användas är **'9. upplösning - jpg.exe'**. 8 bilder skapas och läggs i katalogen 'c:\metoden\upplösning\jpg'.

Steg 4

Efter att bilderna genererats går det att genomföra test på avkodningsalgoritmen/avkodningsalgoritmerna. Testet utförs genom att köra avkodningsalgoritmen på varje bild i alla kategorier. Om alla bilder genererats enligt ovan bör det finnas 105 bilder att testa algoritmen på.

För att utföra testet så smidigt som möjligt används kodskelettet nedan. En förutsättning för att använda kodskelettet är att avkodaren är gjord i java. Kodstrukturen kan självklart användas som mall för att skapa motsvarande struktur till ett annat språk. I kodskelettet måste vissa modifieringar göras för att anpassa koden till den specifika avkodaren som skall testas. De förändringar som behöver göras är t.ex. initiering och inställningar av avkodaren samt anrop till avkodaren och presentation av resultaten. Kodstrukturen finns i bilaga F, via cd-skivan eller på hemsidan http://home.swipnet.se/~w-155814. Förändringarna av koden görs enligt nedan:

Steg 4a

För att få tillgång till den aktuella avkodaren som skall användas gör de importeringar som behövs för att få avkodaren att fungera efter kommentaren:
“ /* IMPORT DECODING ALGORITM CLASS */ ”.

Exempel.

```
/*  import decoding algoritm functions */

import com.briscan.ean13.EanDecoder;

/* ---------------------------------------- */
```

Steg 4b

Nedan visas koden som initierar de variabler som används för visning av resultaten ut från avkodaren. Ändra typ och antal med mera för att tillgodogöra den aktuella avkodaren. Ändringarna skall göras efter kommentaren ” /* INIT SAVE VARIABLES */ ”.

Exempel.

```
/* INIT SAVE VARIABLES    */

int svar;
String resultat;
boolean check;

/* ---------------------------------- */
```

Steg 4c

Deklarera avkodaren efter kommentaren ” /* INIT DECODER NAME */ ”.

Exempel.

```
/* INIT DECODER NAME */

                public static EanDecoder barc;

/* ------------------------------ */
```

Steg 4d

Initiera avkodaren efter kommentaren ” /* INIT DECODER HERE */ ”.

Exempel.

```
/* INIT DECODER HERE */

        barc = new EanDecoder();

/* ------------------------------ */
```

Steg 4e

Efter kommentaren ”/* DECODER STUFF HERE */” gör de förändringar som krävs för att ange vilken bild som skall avkodas, anrop för att utföra avkodningen, anrop för att hämta resultaten från avkodningen samt övriga anrop som eventuellt måste göras för att avkodaren skall fungera.

Exempel.

```
/*  DECODER STUFF HERE */
/*  change if needed          */
/*  the image with the barcode is stored in the variable ‘bi’ as a BufferedImage */

        barc.decode(bi);              // sending BufferedImage ‘bi’ to decoder.

        check = EanDecoder.eanParsed.cchecksum;          // get checksum result

        StringBuffer sBuf = new StringBuffer();

        for (int j = 0; j < 13; ++j)
        {
                sBuf.append((char) (EanDecoder.eanParsed.code[j] + '0'));
        }

        resultat = sBuf.toString();  // get result
```

Steg 4f

Efter kommentaren ”/* SAVE RESULT TO FILE */” ändra de variabler som innehåller resultat från avkodaren. Variablerna skall anges i anropet till funktionen ”output” som utför skrivning till resultat filen. Det som bör var med är bilden namn som finns i String ’fillist[i]’ och en variabel som lagrar resultatet ut från avkodaren, vanligtvis ’resultat’.

Exempel.

```
/*  SAVE RESULT TO FILE    */

                output(fillist[i] + "\t" + check + "\t" + resultat);

/* ---------------------------------- */
```

Steg 5

Efter eventuella ändringar kompilera alla filer i en lämplig kompilator. Om javafilen som innehåller koden med grundstrukturen i bilaga F heter ’koden.java’ kompileras och körs denna fil enligt exemplet nedan. Notera att java måste vara installerat på datorn. Se http://java.sun.com/ för mer information om att hämta hem Java utvecklings kitt, installera och konfigurera Javas SDK (Software Developing Kit), kompilera Java-filer m.m.

Ex. för att kompilera javafilen ’koden.java’ skriv
i DOS kommando prompt:

javac koden.java

För att köra programmet skriv:

java koden ”c:\metoden” ”resultat.txt”

Vid exekveringen av testprogrammet som avkodar alla bilder i alla kategorier ses en progressbar ticka över skärmen. Efter att progressbaren stannat visar applikationen antalet kategorier samt antalet bilder som testats och resultaten blir lagrade i en resultat fil.

Steg 6

I anropet till programmet från kodstrukturen skapad enligt ovan skall två argument anges. Argument ett är sökvägen till den katalogen som utgör stam för katalogträdet som innehåller alla testbilder. Vanligtvis är sökvägen definierad som ”c:\metoden”. Argument två i anropet anger namnet på den text fil i vilket resultatet ut från testkörningen sparas. Filen sparas i sökvägen som anges av argument ett, om filen redan finns skrivs den över.

Ut ur testet fås en tabell som listar varje bild i varje kategori, vidare listas avkodningsresultatet från varje bild. Resultatfilen listar om avkodningsalgoritmen har hittat någon kod i den aktuella bilden samt om resultatet är korrekt avkodat eller inte.

Exempel.

Korrekt avkodning enligt algoritmen har skett då ”checksum = true” samt ”resultat = 123456789128”.

Utdrag från en resultat fil ses nedan.

C:\metoden\rörelseoskärpa

filnamn	**checksum**	**resultat**
01 pix.jpg	true	123456789128
02 pix.jpg	true	123456789128
03 pix.jpg	true	123456789128
04 pix.jpg	true	123456789128
05 pix.jpg	true	123456789128
06 pix.jpg	true	123456789128
07 pix.jpg	true	123456789128
08 pix.jpg	true	123456789128
10 pix.jpg	false	0
12 pix.jpg	false	0
14 pix.jpg	false	0
16 pix.jpg	false	0
18 pix.jpg	false	0
20 pix.jpg	false	0
22 pix.jpg	false	0

C:\metoden\upplösning\jpg

filnamn	**checksum**	**resultat**
086x65.jpg	false	0
114x86.jpg	false	0
153x114.jpg	false	0
203x152.jpg	false	0
270x203.jpg	false	0
360x270.jpg	true	123456789128
480x360.jpg	true	123456789128
640x480.jpg	true	123456789128