CHALMERS
UNIVERSITY OF TECHNOLOGY

Igenkänning och klassificering av trafikskyltar

med hjälp av maskininlärning

Kandidatarbete inom civilingenjörsprogrammet

HELMER FUNDIN
ANTON JOHANNESSON
PERHAM SHAMS

Institutionen för Signal och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2015

KANDIDATARBETE SSYX02-15-02

Igenkänning och klassificering av trafikskyltar

med hjälp av maskininlärning

HELMER FUNDIN
ANTON JOHANNESSON
PERHAM SHAMS

Institutionen för Signal och system
Chalmers Tekniska Högskola
Göteborg, Sverige 2015

Igenkänning och klassificering av trafikskyltar
med hjälp av maskininlärning
HELMER FUNDIN, ANTON JOHANNESSON, PERHAM SHAMS

Handledare: Yixiao Yun, Signaler och system
Examinator: Irene Gu, Signaler och system

Kandidatarbete: SSYX02-15-02
Institution för Signal och system
Chalmers tekniska högskola
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Göteborg, Sverige, 2015

Sammandrag

Bildigenkänning och klassificering av videodata har länge existerat och använts i olika syften. När bilindustrin nu visat intresse för tekniken har det dock uppstått problem då det inte finns någon tillräckligt noggrann metod framtagen för klassificering av vägskyltar. Tekniken är bland annat avsedd att användas i självkörande bilar och kräver därför extremt hög noggrannhet för att säkerställa en säker trafikmiljö. I denna rapport redogörs för ett omfattande arbete med mål att uppnå en så hög klassificeringsnoggrannhet som möjligt av trafikskyltar. Resultatet redogör träffsäkerheten för de två kännetecknensmetoder som valts att användas, HOG och Gabor, i de SVM-system som tränats samt den sannolikhetssummering dem emellan som konstruerats för att maximera den totala träffsäkerheten. Relativt goda resultat har uppnåtts, med en trolig förbättringsfaktor i en ökad urvalsmängd av bilder.

Innehållsförteckning

1. Inledning

Inledningskapitlet ger en introduktion till rapporten. Här berättas det om dagens teknik som använder sig av igenkänning och varför det finns ett behov för att utveckla system för att känna igen trafikskyltar. För att klargöra arbetsmetdiken och målet med projektet sätts ett antal delproblem och avgränsningar upp.

1.1 Bakgrund

Maskininlärning från videoupptagning eller bilder är ett relativt nytt forskningsområde som det idag forskas aktivt kring. Genom ökad förståelse för dess stora potential och många användningsområden har många forskningsgrupper och företag intresserat sig för området och bara det senaste årtiondet har markanta förbättringar gjorts kring tekniken. De mest kända användningsområdena som använder sig av maskininlärning och igenkänning är områden såsom ansikts- och irisigenkänning, fingeravtrycksmatchning och så vidare och nu på senare år dyker allt fler användningsområden upp. En av dessa är automatisk tolkning och analys av omgivningen i trafiken som är ett viktigt steg för effektiv analys av trafik och infrastruktur. Vägmärkesanalys kan användas inom flera områden, t.ex. vägmärkesigenkänning, trafiksäkerhet, trafiktäthetsmätningar, trafikövervakning och aktiv kontroll av vägtrafik. Anledningen till att dessa applikationer är så viktiga är på grund av skyltars centrala roll i trafiken.

Vägmärken har främst till uppgift att i trafiken visa var faror kan finnas, hjälpa till att navigera så att föraren kan ha så stor fokus på vägen som möjligt och varna föraren om var svårigheter och hinder kan uppkomma. Ett underlättande system möjliggör för föraren att till exempel i efterhand kunna se på sin instrumentbräda vilket skylt han precis passerat, eller vilken hastighetsbegränsning som råder på vägen, vilket minska risken för trafikolyckor.

Redan idag utvecklas system för att bland annat användas inom ADAS (Advanced Driver Assistance Systems) [17], som är det snabbast växande segmentet inom fordonselektronik. Här används vägmärkesanalys för att öka uppmärksamheten i trafiken genom att till exempel hjälpa föraren se i god tid vilka skyltar som gäller framöver och vilka skyltar denne eventuellt missat att observera. Speciellt relevant är utvecklingen av denna teknik till autonoma bilar som behöver ett system som måste vara relativt felfritt för att kunna fungera i trafiken samt minimera risken för olyckor. Eftersom autonoma bilar ser ut att vara framtidens fordon är forskningen kring TSR (Traffic Sign Recognition) essentiell för den framtida fordonsindustrin. Uppkomsten av nya och omfattande databaser med tusentals bilder av vägmärken har gjort det lättare för ingenjörer och forskare att testa olika system, till exempel German Benchmark Dataset [1].

Bristerna i dagens befintliga system gäller träffsäkerheten på systemen. Det är oerhört svårt och komplicerat att utveckla system som med 100 % kan känna igen och klassificera alla skyltar, under olika omständigheter. Skyltarna eller visionen av skyltarna kan påverkas av diverse störningar som

systemen måste kunna ta hänsyn till under klassificeringen. Här behövs ytterligare tester och utveckling av området för att få så felfria och snabbt processande system som möjligt.

1.2 Syfte

Syftet med det underliggande arbetet till denna rapport är att ta fram ett system som klassificerar bilder på vägmärken med en så hög noggrannhet som möjligt. Systemet består av MATLAB-algoritmer som bearbetar bilder och klassificerar dessa mot en intränad maskin. Systemets noggrannhet kommer sedan att jämföras med existerande systems och utvärderas därefter.

1.3 Problemformulering

Rapporten tar huvudsakligen upp problemen med igenkänning av trafikskyltar. Systemet har som uppgift att kunna identifiera skyltar från bilder som kan ha en mängd andra objekt i sig, såsom en gatuvy. En kamera på fordonet tar en ström av bilder som skickas vidare för igenkänning.

Igenkänningen delas in i tre delproblem. Det första delproblemet rör införskaffandet av en databas med referensbilder och en databas med testbilder. Här tas beslut om eventuellt vilka existerande databaser som ska användas. Dessa bilder behövs för träning samt testning vid klassifikationsprocessen (SVM) [2].

När en databas med referensbilder och testbilder har införskaffats så måste beslut tas gällande vilka kännetecken på skyltarna som är av intresse, delproblem två. Exempel på kännetecken är hörn, kanter, färger, kontraster och så vidare. Detta är en av de viktigaste aspekterna med uppgiften och de val som görs här kommer i stor grad att påverka slutresultatet.

Efter att valet av kännetecken är genomfört så ska en klassifikationsprocess utformas för att hitta det effektivaste systemet för att analysera testbilderna. Syftet med detta är att snabbt kunna göra jämförelser mellan tusentals referensbilder och testbilderna som skickas in och hitta en god träffsäkerhet för systemet.

1.4 Avgränsningar

Under projektets gång kommer enbart tolv olika skyltar att användas och analyseras. Detta val är baserat på att det är för tidskrävande att göra en analys och klassifikation om antalet skyltklasser är för stort, vilket hamnar i konflikt med den uppsatta tidsramen, som sträcker sig från den 23 januari till den 19 maj.

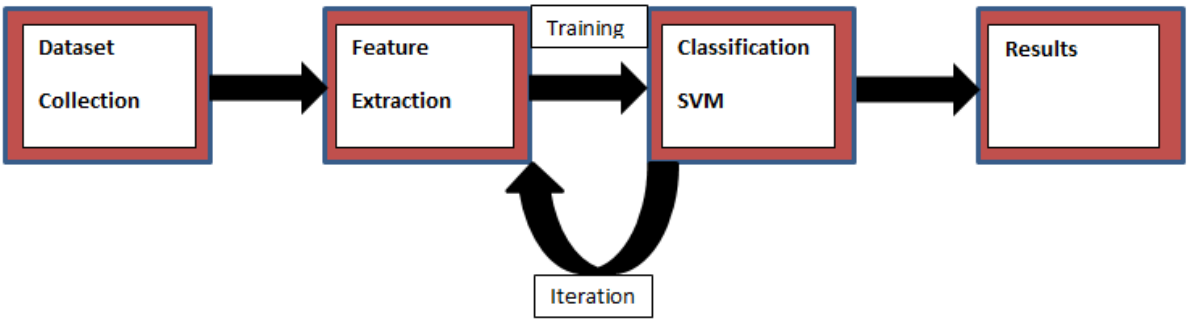
De skyltar som valts ut till att ingå i databasen är de vanligaste internationellt igenkännbara skyltarna och sådana som anses vara viktiga, såsom stoppskylt, väjningspliktsskylt och hastighetsbegränsningsskyltar. Algoritmen som sedan ligger till grund för klassificeringen ska endast analysera stillbilder. Arbete rörande analysering och klassificering av rörliga bilder, video, kommer alltså inte att genomföras. Stillbilder är lättare att analysera, och systemet kan senare uppdateras till att också kunna hantera rörliga bilder.

Det finns tre aspekter som vanligtvis är intressanta för den här typen av algoritmer: klassifikationsratio, falsklarmsratio och hur tidseffektiv algoritmen är. Tidsaspekten kommer däremot inte beaktas i detta arbete då programmeringen kommer att utföras i MATLAB. MATLAB är inte lika tidseffektivt som många andra programmeringsspråk men är anpassat för just utveckling av nya algoritmer då det är mycket användarvänligt och erbjuder användaren att lättare ändra i systemet. Om klassifikationsration samt falsklarmsration är tillfredsställande så finns dock möjligheten att översätta algoritmerna till ett tidseffektivare programmeringsspråk.

1.5 Metod

Som nämnts i problemformuleringen (1.3) består det första delproblemet i att samla in en databas med urvalsbilder. Olika tillvägagångssätt kommer att utvärderas inklusive möjligheten att själva samla in en databas med egna bilder med eventuellt stöd av existerande databaser.

Nästa moment, känneteckensextrahering, går ut på att med hjälp av existerande algoritmer hitta specifika kännetecken hos bilderna [3]. För att kunna optimera den så kallade kännetecken-extraheringsprocessen måste algoritmerna analyseras, eventuellt modifieras och kombineras. Allt för att i sista delmomentet av arbetet kunna klassificera de olika skyltarna, den så kallade klassificeringen. Dessa två delmoment är sammankopplade och kommer till stor del att överlappa varandra. Eftersom klassificeringen bygger på de existerande algoritmerna för igenkänning av kännetecken, kommer kontinuerlig iteration mellan momenten vara nödvändigt.



Figur 1.1: Systemets indelning

Klassificeringen kommer genomföras med hjälp av en så kallad Support Vector Machine (SVM). SVM analyserar data och känner igen mönster och används oftast för klassificering och regressionsanalys [2]. Med hjälp av testbilder ska systemet kunna klassificera skyltar baserat på referensbilder i databasen.

2. Teori

Teorikapitlet beskriver de viktigaste teoretiska aspekterna kring de metoder som används för att skapa systemet. Här tillkommer förklaringar och ett stort antal referenser från en mängd forskare som är väl insatta inom olika områden av bildhantering. Till en början fås en inblick i vad ett kännetecken är för någonting och en fördjupning till de kännetecken sextraheringsmetoder som väljs. Därefter tillkommer ett kapitel som handlar om maskininlärning och en metod kallad SVM. Avslutningvis beskrivs förbearbetningsprocessens och vissa matematiska applikationer teoretiska bakgrund.

2.1 Kännetecken

Grunden för att kunna identifiera och klassificera bilder ligger i kännetecken sextrahering. Kännetecken sextrahering går ut på att hitta kännetecken hos bilden som går att identifiera och kvantifiera. Det finns flera olika existerande matematiska metoder för att extrahera kännetecken ur en bild. För att systemet ska kunna få den låga felmarginal som strävas efter är valet av metoder för kännetecken sextrahering essentiell. Därför måste samtliga existerande metoder undersökas för att hitta den mest effektiva kombinationen av metoder.

Ett kännetecken kan till exempel vara formen av ett hörn. Algoritmen i det fallet är uppbyggd så att alla hörn i en bild hittas, och dess positioner sparas i en variabel. Andra kännetecken sextraheringsmetoder (t.ex. HOG) använder en gråskala för att identifiera alla kanter i en bild. Vilken av metoderna som fungerar bäst varierar beroende på bild och motiv. En kombination av olika metoder är också ett alternativ. Samtliga kännetecken för samtliga bilder används sedan till klassificeringen med hjälp av ett SVM-system. I den resterande delen av avsnittet (2.1) redogörs för de olika kännetecken sextraheringsmetoder som projektet undersökt.

2.1.1 Kernels

Kernels används i huvudsak till att förstärka, alternativt släta ut, kanter i en bild. Det som görs är att en liten matris, ofta 3x3 pixlar stor, multipliceras med en del av bilden som har samma dimensioner som kerneln. Summan av den resulterande matrisen placeras sedan i mitten av matrisen som extraherats från bilden. Detta resulterar i att kanter antingen blir förstärkta eller utslätade vilket i sin tur leder till att bilden blir enklare att analysera för vissa av de kännetecken som finns att tillgå.

$G_y =$

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

$G_x =$

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

Figur 2.1: Exempel på två kernelmatriser av typen Sobel.

$$I = Bild(i-1{:}i+1, j-1{:}j+1) =$$

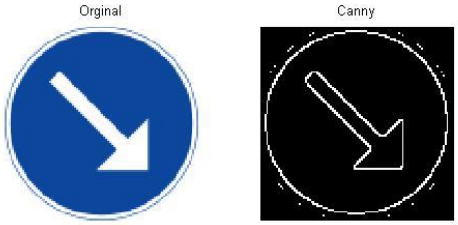
255	0	255
255	72	156
120	255	236

Figur 2.2: Pixelvärden från bilden I.

$$G(i,j) = \sqrt{\sum(G_y * I)^2 + \sum(G_x * I)^2} = 255 \tag{2.1}$$

Figur 2.3: Kernels i figur 2.1 appliceras på pixeln markerad med grönt i bilden I i figur2.2.

I figuren 2.3 illustreras beräkningar för en specifik kernel, där G_y är y-kerneln, G_x är x-kerneln och I är en matris med bildens pixelvärde. Pixelns värde är från början 72, vilket avser ett relativt mörkt värde på gråskalan. Dock så är det generella värdet i omgivningen ljus vilket medför att efter applicering av den utformade kerneln så kommer det nya värdet bli 255 vilket är vitt. På sådant vis kan alltså kanterna på bilden förstärkas.



Figur 2.4: Före och efter applicering av en Canny-kernel på en påbudsskylt i MATLAB.

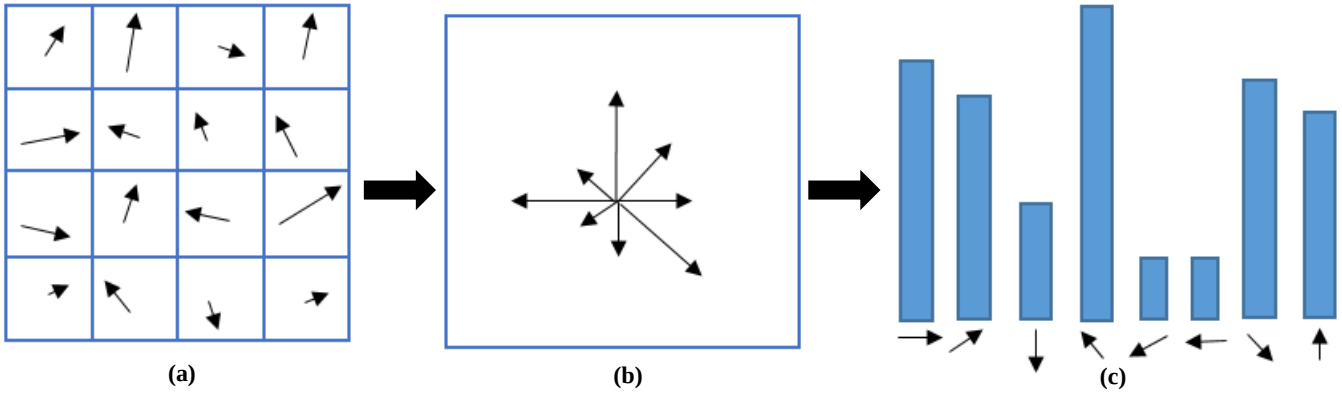
2.1.2 Threshold

Thresholding är den enklaste metoden av bildsegmentering. Genom att ändra en bild så att den visas i gråskala, kan thresholding användas för att skapa binära bilder. Principen är lättast att förklara med ett enkelt exempel. Varje bild består av ett visst antal pixlar. Varje pixel i en bild ersätts mot en svart pixel ifall bildintensiteten i punkten är lägre än en fix bestämd konstant, och byts ut mot en vit pixel om bildintensiteten är högre. På detta sätt kan ett föremål utskiljas i en bild skiljt från bakgrunden. I figur 2.5 kan thresholding ses applicerad på en stoppskylt. De delar av skylten och bakgrunden som har en hög intensitet, exempelvis skyltens kanter och text har blivit vit, medan de delar med lägre intensitet, exempelvis skyltens röda delar, har blivit svart. Med hjälp av denna metod kan andra kännetecken i bilden bli tydligare och därmed kan det förenkla känneteckensextraheringen när andra metoder används.

Figur 2.5: Exempel på thresholding på en bild med en stoppskylt.

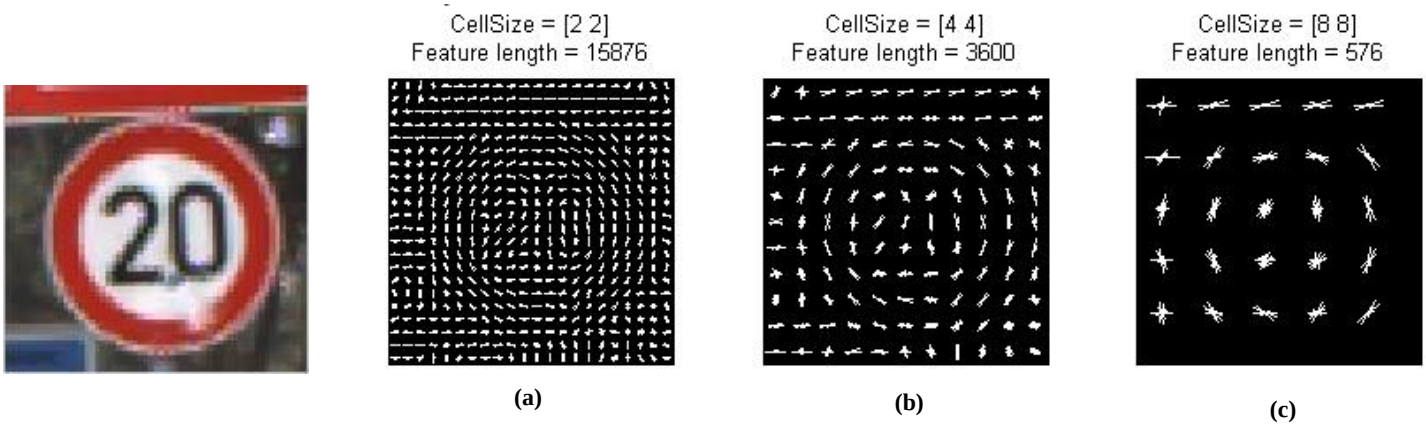
2.1.3 HOG (Histogram of Oriented Gradients)

Metoden etablerades av forskarna Dalal och Triggs under 2005 [7] och beräknar förekomsten av gradientsorientering i lokaliserade delar av en bild och är ett effektivt sätt att fånga in information om form. Den grundläggande idén bakom HOG är att föremål eller former i en bild kan beskrivas genom distributionen av intensitetsgradienter eller kantriktningar. Bilden delas in i små kopplade regioner, så kallade celler, och för varje cell kompileras en HOG-riktning eller kantorientering för varje pixel i cellen. Kantorienteringen är indelad i ett antal ”fack”. I figurerna 2.6**a**, 2.6**b** och 2.6**c** kan ett exempel observeras på hur dessa celler ser ut och hur de kan bilda histogram. Figur 2.6**a** är en cellindelning av en bild och det går att se dess olika gradienter. I figur 2.6**b** kombineras alla gradienter i 2.6**a** till en sammansättning, där gradienter som har samma riktning sätts samman. Slutligen visar figur 2.6**c** ett histogram över sammansättningen i figur 2.6**c**.



Figur 2.6: Histogram skapas från gradienter i 3 olika steg, a, b och c.

Ett val görs av hur många celler och cellgrupper bilden ska delas in i. Desto fler celler, ju mer data kan användas för att rita ut gradientriktningarna. Är bilden dåligt upplöst eller har få pixlar hjälper det att ha en större indelning, vilket kan observeras i figurerna 2.7**a**, 2.7**b** och 2.7**c** som jämför samma bild på en hastighetsbegränsningsskylt fast med olika cellstorlekar. I figur 2.7**a** som har cellstorlek 2x2 får ett större antal kännetecken plats än i exempelvis cellstorleken 8x8 i figur 2.7**c**.



Figur 2.7: HOG med olika cellstorlekar applicerat på en hastighetsskylt

Det första beräkningssteget ligger i att ta fram gradientvärden. Det görs vanligtvis genom att applicera en endimensionell centrerad punkt i både horisontell och vertikal riktning. Metoden medför att en filtrering i gråskala av bilden kan genomföras med hjälp av filterkernels [7], där K_x och K_y är kernels, I är bildens pixelvärden, G är gradientens magnitud och θ är gradientens orientering:

$$K_x = [-1,0,1], K_y = [-1,0,1]^T \quad (2.2)$$

Derivatorna i x och y fås genom att använda en faltningsoperation av bilden I:

$$I_x = I * K_x \, , I_y = I * K_y \quad (2.3)$$

Gradientens magnitud ges av:

$$|G| = \sqrt{I_x^2 + I_y^2} \quad (2.4)$$

Slutligen beräknas gradientens orientering:

$$\theta = \arctan\frac{I_y}{I_x} \quad (2.5)$$

2.1.4 Gaborfilter

Gaborfilter är ytterligare en metod som går att utnyttja för att extrahera kännetecken från bilder och har varit en populär metod i över tre decennier. Metoden uppmärksammades och började implementeras på grund av dess likheter med det mänskliga visuella systemet. Gaborfilter används därför i stor grad inom mönsteranalys såsom igenkänning och detektering av ansikten, iris igenkänning och fingeravtryckmatchning [15].

Frekvenser och deras orienteringar är viktiga strukturegenskaper hos bilder [21]. Genom att ansätta en rad Gaborfilter med olika frekvenser och orienteringar kan kännetecken extraheras från bilder. Impulsresponsen från filtren skapas genom att multiplicera en Gaussfunktion med en harmonisk funktion. Orienteringen görs möjlig genom att funktionerna som skapar filtren utvidgas till 2D [16]. 2D-filtren representeras av en grupp av ”krusningar”, vilka optimalt fångar in både lokal orienterings- och frekvensinformation från en bild [18].

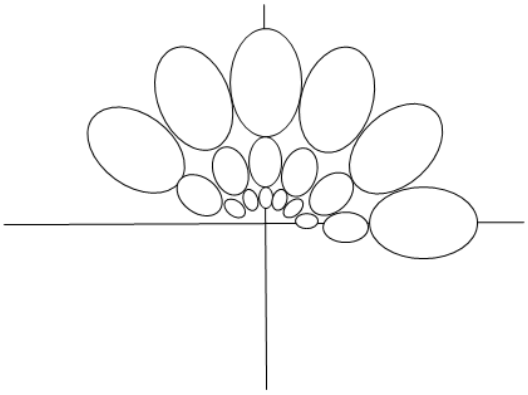
Följande ekvationer beskriver en generell 2D Gabor funktion [16]:

$$g_{\lambda\theta\sigma\gamma\varphi}(x,y) = \exp\left(\frac{-x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\exp(i\left(\frac{2\pi x'}{\lambda} + \varphi\right)) \quad (2.6)$$

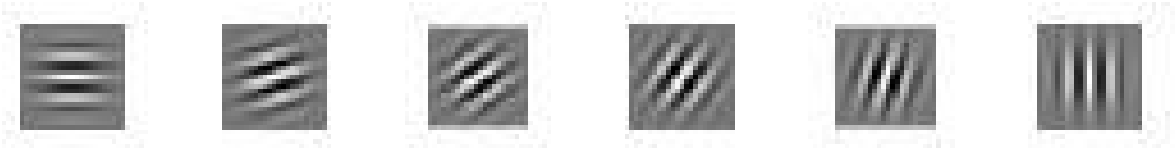
$$x' = xcos\theta + ysin\theta \quad (2.7)$$

$$y' = -xsin\theta + ycos\theta \quad (2.8)$$

Där x, y specificerar pixelvärdet för en bild vid koordinaterna (x, y), λ representerar den sinusformade faktorns våglängd, θ är orienteringen, φ är fasförskjutningen, σ är standardavvikelsen av den Gaussiska faktorn som bestämmer storleken på det receptoriska fältet och γ är den spatiala aspektration som bestämmer ellipticiteten.



Figur 2.8: Figuren visar ett exempel på ett spektrum av ett Gaborfiltret med frekvensen tre (antal lager) och med sex stycken orienteringar (antal ellipser per lager). Med detta satta filter kan till exempel en bild brytas ned i 3x6, 18 stycken delbilder. Varje delbild kan ge information om bildens specifika egenskaper och upplösning och därmed fås en analys av bildens textur.



Figur 2.9: Exempel på Gaborfilter med olika frekvenser och orienteringar (intensitetsplot).

2.2 SVM

SVM är en användbar metod för att klassificera data och är baserad på matematisk statistisk inlärningsteori. Teorin togs fram av Vapnik och hans forskargrupp 1992 [8] som senare vidareutvecklats till SVM. Klassifikationsprocessen innebär att en separering görs av data; en uppsättning för träning och en för testning. Vid varje fall innehåller träningsuppsättningen ett ”målvärde” och flera kännetecken. Syftet med SVM är att producera en modell som förutsäger målvärdet från testdata givet testdatas kännetecken.

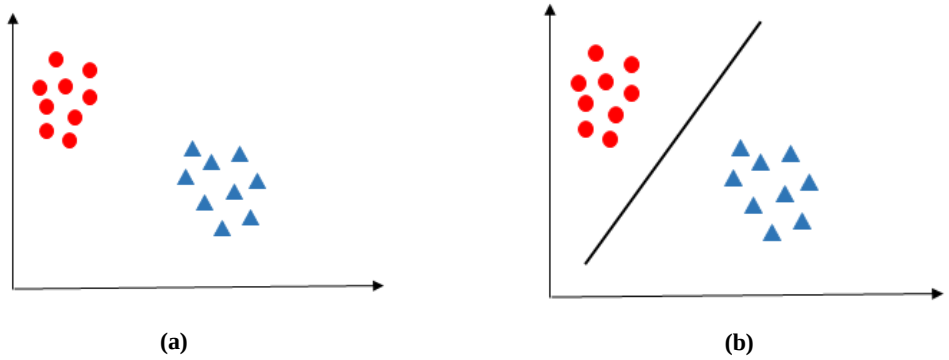
Generellt kan sägas att SVM konstruerar ett hyperplan eller en uppsättning hyperplan i ett oändligt eller högdimensionellt rum. En god separation blir uppnådd då ett hyperplan erhållits som har den längsta distansen till de närmaste datapunkterna från de olika klasserna. Generellt resulterar en större marginal till mindre generaliserade fel i klassifikationen.

2.2.1 Linjär klassifikation

Ponera att det finns två givna klasser av skyltar. Syftet är att hitta ett plan som separerar de två klasserna och därefter tilldela rätt klass till rätt bild (skylt). Ett förenklat scenario med två olika klasser kallas binärklassifikation. Ett exempel på detta kan ses i figur 2.10. Idén med SVM blir här alltså att hitta ett optimalt hyperplan som linjärt separerar de två klasserna. Det optimala hyperplanet har maximalt lika långt avstånd från de båda klasserna, och kallas därför för ett maximummarginalhyperplan som kan betraktas i figur 2.10**b**.

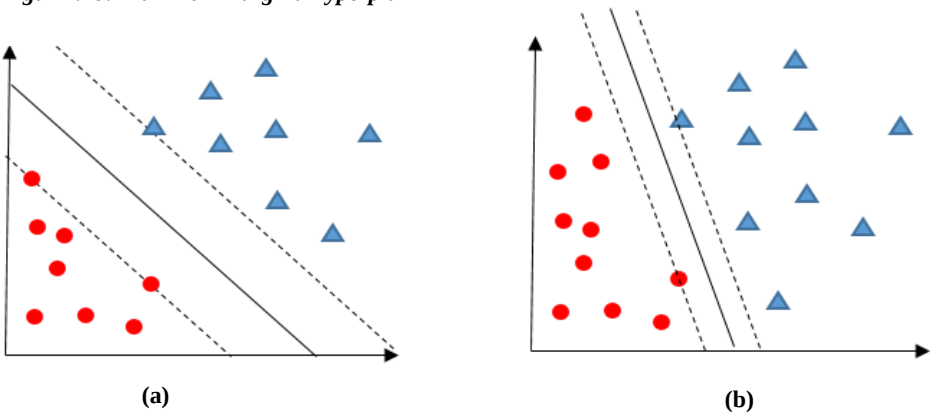
För att få den maximala marginalen genomkorsas klasserna av var sitt träningsmönster, som kan iakttas i figurerna 2.11, där träningsmönstren är de streckade linjerna och inom dessa gränser rensas eventuella klasser bort. Dessa mönster kallas supportvektorer och de bestämmer positionen av hyperplanet.

Det finns två klara begränsningar med linjär inlärning. För det första är träningsdata normalt sett påverkad av störningar och det finns inga garantier att klassificeringen av datan blir korrekt. För det andra kan funktionen som försöker läras in ha en komplex framställning, vilket skulle innebära att den inte kan verifieras på detta sätt.



(a) **(b)**

Figur 2.10: Maximummarginalhyperplan

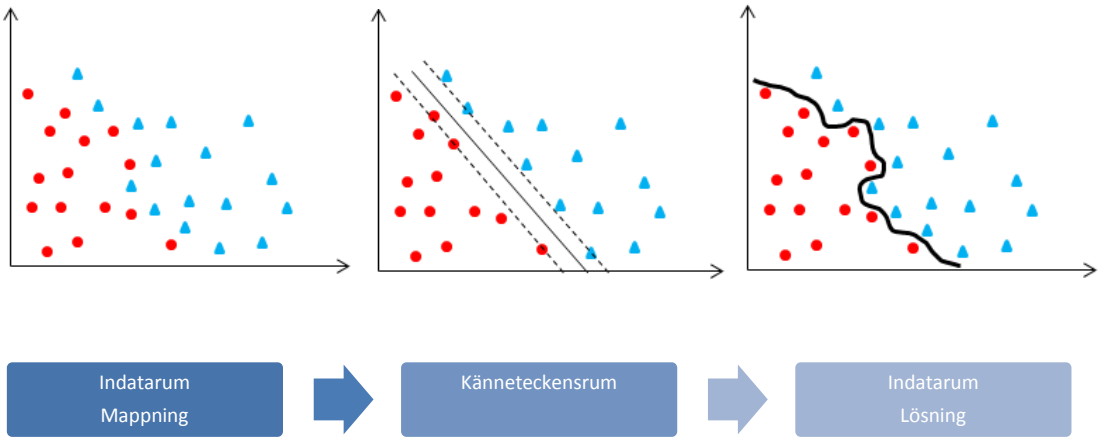


(a) **(b)**

Figur 1.11: a hyperplan med stor marginal, b hyperplan med liten marginal.

2.2.2 Olinjär klassifikation

Vid verkliga problem är ofta de involverade klasserna inte linjärt separerbara. Figur 2.12 påvisar detta problem genom att visa två klasser som överlappar varandra (röda cirklar blandade med blåa trianglar). Även om det är omöjligt att erhålla ett separerande hyperplan i indatarummet, är det möjligt att göra så i ett högdimensionellt kännetecknsrum [8]. Inmatad data blir *mappad* in i det högdimensionella kännetecknsrummet, där det är lättare att utföra en separation och klassifikationen kan ske med hjälp av en linjär SVM. *Mappning* innebär processen där objekten eller skyltarna omdisponeras, i det här fallet med hjälp av matematiska funktioner, så kallade kernels [9]. Slutligen genom att transformera tillbaka till indatarummet, kan en olinjär beslutslinje fås som kan separera de två klasserna.



Figur 2.12: Mappning från och från indata-rum och kännetecknsrum.

2.2.3 Val av kernel

Det finns ett antal kernelfunktioner att välja mellan. Fyra av dessa beskrivs i figur 2.13 [9]. Kernel-funktionen representerar en skalärprodukt av inputdatapunkter som mappas in i det högdimensionella kännetecknsrummet genom transformering.

$$K(X_i, X_j) = \begin{cases} X_i \cdot X_j & \text{Linjär} \\ (\gamma X_i \cdot X_j + C)^d & \text{Polynomisk} \\ \exp(-\gamma |X_i - X_j|^2) & \text{RBF} \\ \tanh(\gamma X_i \cdot X_j + C) & \text{Sigmoid} \end{cases}$$

Figur 2.13: Kernelfunktioner för SVM.

RBF (Radial basis function) är den mest populära av kerneltyperna som används i SVM. Detta beror på att det blir färre numeriska svårigheter och att den kan lokalisera svar över hela den reella x-axeln [2].

C är en regulariserad parameter. Valet av värde på *C* kommer påverka marginalen i SVM. Det innebär att ett litet *C* tillåter att restriktioner lättare blir ignorerade, vilket bidrar till en stor marginal. Väljs ett stort *C* gör det att restriktionerna blir svårare att ignorera, vilket i sin tur skapar en snäv marginal [10]. *Gamma* parametern definierar hur långt influensen från ett träningsexempel når, där höga värden betyder kort avstånd och låga värden betyder långt avstånd. $X_i \in \mathbb{R}^n$ är uppsättning av träningsbilder.

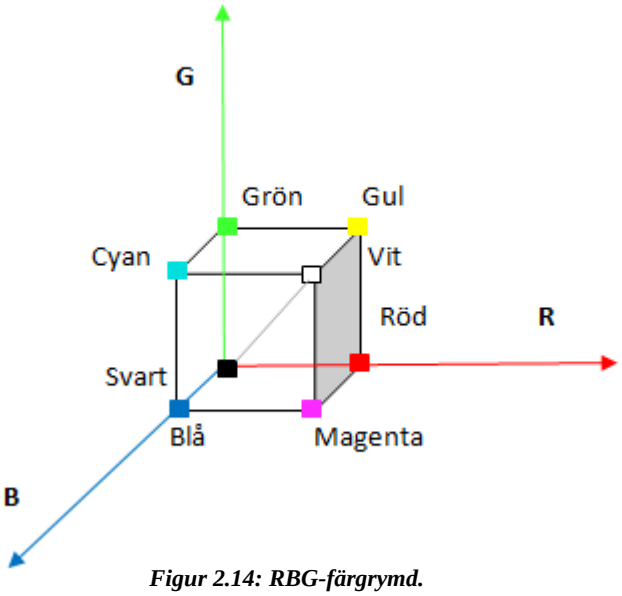
SVM är en binär klassifikationsalgoritm och hanterar endast två klassproblem. Det finns dock möjligheter till att konstruera multi-klass SVM för att klassificera fler klasser, till exempel *all-together*, *one-against-all* och *one-against-one* som metoder.

2.3 Förbearbetning

Färger har en viktig roll i att förse föraren med information om vad trafikskylten har för syfte. Därför skiljer sig dessa färger oftast från färger som finns i naturliga eller urbana miljöer för att skyltarna ska vara urskiljbara. Standard för skyltfärger i Europa är att röd förmedlar att något är förbjudet, blå gäller påbud och gul eller vit är varnande färger[12]. Eftersom skyltar är strikt uppdelade i färger så kan färgerna användas för att förbearbeta bilder. Bilder som tas måste specificeras i en vald färgrymd och därefter kan valda kännetecken i en bild förstärkas innan applicering av känneteckensextrahering. Detta kan underlätta klassificeringen av skyltarna.

2.3.1 RBG-Färgrymd

RBG är en av de mest använda färgrymderna och används ofta inom datorgrafik. Systemet är indelat i tre värden, R=röd, B=blå och G=grön. För att illustrera rymden formas en kub där koordinaterna x, y och z representerar R, B och G, se figur 2.14. I kubens olika hörn kan förutom färgerna röd, blå och grön färgerna magenta, cyan, gul, svart och vit finnas. Dessa färger står för de olika kombinationer som bildas då grundfärgerna blandas på olika sätt. Till exempel fås magenta då röd och blå kombineras i lika mängd, gul fås då grön och röd förenas och färgen cyan när grön och blå blandas. Svart befinner sig i origo medan vit befinner sig i det hörn där alla grundfärger sammanförs.



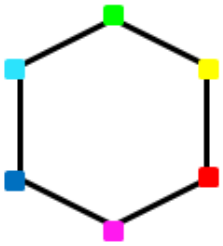
Figur 2.14: RBG-färgrymd.

2.3.2 NMI, Nyans, Mättnad, Intensitet (HSV)

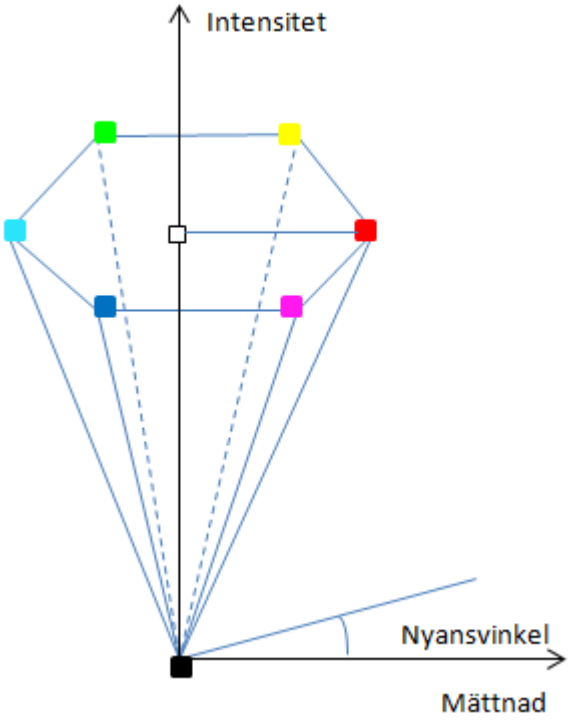
När RGB-bilder analyseras kan problem uppstå på grund av kromatiska variationer av dagsljus i bilden. Detta medför att det blir svårt att separera färginformationen från ljusstyrkan. Det kan därför vara nödvändigt att konvertera bilderna från RGB-färgrymden till en färgrymd kallad NMI. Istället för ett set av grundfärger använder sig NMI av färgspecifikation där användaren väljer en färg från ett spektrum och olika mängder vit och svart. Detta görs för att få olika toner, skuggor och nyanser, se figur 17. 3D-representationen av NMI-modellen har sin grund från den tidigare visade RGB-kuben, se figur 14. Observeras kuben snett ovanifrån bildas en hexagon, där RGB-kubens gränsfärger finns att finna i varje hörn.

Nyansen representeras av hexagonens gränser och är en term som beskriver den dimension av färg som upplevs när en färg observeras. Vinkeln runt den vertikala axeln bestämmer vilken nyans som fås ut och varje ren färg erhålls var sextionde grad. Mättnaden refererar till dominansen av nyans i färgen eller färgfullheten relativt ljusstyrkan. Den representeras av den horisontella axeln, alltså kan hexagonformen ses som ett hjul där nyansernas färgdominans minskar mot mitten av hjulet, där det i det absoluta centrum inte finns någon nyans som dominerar (vit). Intensiteten varierar från 0 vid apex (svart) till 1 som ligger på toppen (vit) som är den maximala intensiteten, se figur 2.16.

$0 \leq Nyans \leq 360°$
$0 \leq Mättnad \leq 1$
$0 \leq Intensitet \leq 1$

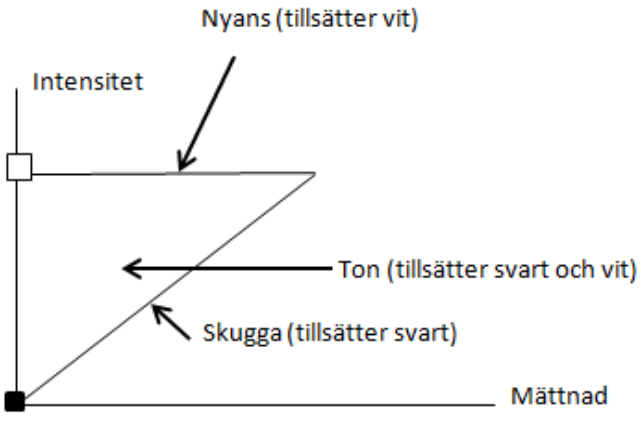


Figur 2.15: Hexagonfigur, NMI-modell sedd ovanifrån.

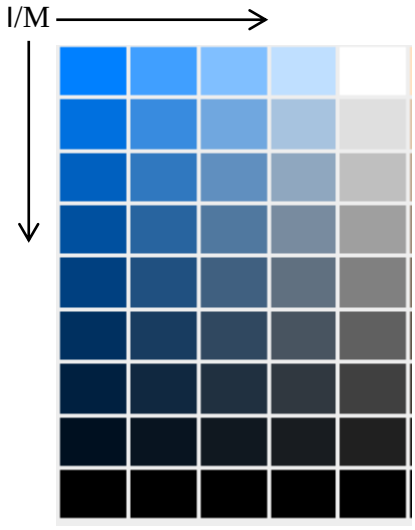


Figur 2.16: NMI-modell.

Tvärsnittet av 3D-modellen uppvisar möjligheter till variering av skugga, toner och nyanser genom att tillsätta vit och svart [11], se figur 2.17. I figur 2.18 kan variationen mellan 0 och 1 för mättnaden och intensiteten iakttas, där maximala värdet finns i det övre vänstra hörnet. Mättnaden sjunker horisontellt medan intensiteten sjunker vertikalt.



Figur 2.17: Nyans, ton och skugga – modell.



Figur 2.18: Exempel på variation av mättnad och intensitet på färgen blå, där pilarna visar hur värdet sjunker från 1 till 0.

2.4 Kännetecknensstandardisering

Inom bildbearbetning är normalisering en process där ändringar på bildens pixellintensitetsvärde görs. Metoden kallas kännetecknsnormalisering eller skalning och behöver appliceras då bredden av värdena från bildernas kännetecken kan variera stort. Vid användning av vissa maskininlärningsalgoritmer (SVM) krävs normalisering för att funktioner ska fungera ordentligt och kan därmed öka systemets träffsäkerhet [13]. Det finns ett antal tillvägagångsätt när normalisering ska utföras och standardisering är den metod som fungerar bra vid användande av SVM som maskininlärningsmetod. Först beräknas medelvärdet och standardavvikelsen för varje kännetecken. Därefter subtraheras medelvärdet från varje kännetecknsvärde. Avslutningsvis divideras detta värde från varje kännetecken med dess standardavvikelse för att få ut det normaliserade värdet [14], se formel.

$$\boldsymbol{f_i} = [\boldsymbol{f}, \boldsymbol{f_2}, \dots, \boldsymbol{f_N}] \tag{2.9}$$

$$Norm_{\boldsymbol{f_i}} = (\boldsymbol{f_i} - \mu)/\sigma \tag{2.10}$$

$\boldsymbol{f_i}$ är en matris med kännetecknsvärden, μ är ett medevärde av matrisen f och σ är standardavvikelsen av $\boldsymbol{f_i}$. Standardiseringen tillämpas på samtliga kännetecknsvektorer.

2.5 K-Faldig Korsvalidering

K-faldig korsvalidering är en standardmetod som används när parametrar i SVM behöver justeras, eftersom dessa till stor del kan påverka klassifikationsresultatet. Metoden ersätter behovet av att göra en manuell sökning efter de lämpligaste värdena på parametrarna C och γ. Korsvalidering hittar istället dessa värden genom en itererande process. Metoden ger därmed en uppskattning på klassifikationsprocessens prestation. Processen för korsvalidering är relativt simpel och proceduren inleds med att träningsbilderna delas in i K stycken grupper, där K är ett positivt heltal, oftast med värdet 5 eller 10. Därefter utförs träning med $(K - 1)$ antal av grupperna medan en av grupperna får vara testgrupp. Detta upprepas K antal gånger, där varje grupp får vara testgrupp en gång. När K stycken antal iterationer har körts tas ett genomsnitt av alla klassifikationsfel, som blir det värde för de parametrar som ansatts och är det värde som kommer jämföras. Därefter byts parametrarna ut och iterationerna görs om igen tills intervallet parametrar som ska testats kört igenom. Avslutningsvis kan det C och γ väljas som ger lägst korsvalideringsfel och kan börja användas för att träna SVM.

2.6 PCA

Vissa kännetecknsmetoder genererar väldigt långa vektorer, vilka kan vara tunga och ta väldigt lång tid att analysera. För att komma runt detta problem och underlätta framförallt korsvalideringens långa datorsimuleringar finns det möjlighet att förkorta samtliga vektorer med hjälp av en metod som kallas PCA, *Principal Component Analysis*. Metodens mål är att minska dimensionen på en vektor samtidigt som minimalt med data förloras.

Det finns ett par matematiska steg som PCA består av [20]:

1. **Medelvärde**: $\mathbf{X}=\frac{\sum_{i=1}^{n} X_i}{n}$ (2.11)
   Där $\boldsymbol{X}$ är den ursprungliga kännetecknsmatrisen och *n* är antalet element.
2. **Kovariansmatris**: $kov = \sum_{i=1}^{n} \frac{(X_i - \boldsymbol{X})(Y_i - \boldsymbol{Y})}{(n-1)}$ (2.12)
   Där $\boldsymbol{Y}$ är lika med $\boldsymbol{X'}$ (transponat). Kovariansen används för att hitta relationen mellan dimensionerna i $\boldsymbol{X}$ då $\boldsymbol{X}$ större än en dimension.
3. **Egenvärden** och **egenvektorer** beräknas av kovariansmatrisen. Dessa extraherar information om data i $\boldsymbol{X}$ från kovariansmatrisen. Egenvektorerna har vissa egenskaper som kommer till användning när antalet dimensioner behövs minskas ned. Även om vektorn skalas en mängd innan du multiplicerar den, får du fortfarande samma multipel, eftersom vektorns direktion inte ändrar sig. Dessutom är alla egenvektorer av en matris vinkelräta mot varandra oberoende av vilka dimensioner matrisen har. Detta gör det möjligt att använda de vinkelräta egenvektorerna för att uttrycka informationen om data.
4. **Kännetecknsvektor** $\boldsymbol{K} = (egenvärde_1,\ egenvärde_2, \dots, egenvärde_n)$ skapas från egenvärdena, där de med lägst signifikans (lägst värde) kan väljas att tas bort. Här förloras en del information men dimensionerna minskar desto mer som kan tas bort.
5. Kännetecknsvektorn $\boldsymbol{K}$ med egenvärdena multipliceras med den ursprungliga matrisen $\boldsymbol{X}$ för att få ut vad som är sökt; en **kännetecknsmatris med mindre dimensioner**:
   $\boldsymbol{X_{ny} = K * X}$ (2.13)

2.7 Störningar mot igenkänning av färg och kännetecken

När en bild ska analyseras och vissa viktiga färger och kännetecken ska extraheras från bilden är det essentiellt att ta hänsyn till eventuella störningar som kan påverka igenkänningen. Eftersom det i det här fallet handlar om bilder på trafikskyltar som kan befinna sig i en rad olika miljöer och är relativt sårbara för påverkan, finns det en rad faktorer som bör tas hänsyn till:

1. Det finns ett antal olika vädersituationer som kan påverka bildernas kvalitet. Dimma, regn och snö kan alla påverka skyltarnas synlighet och minskar det maximala avståndet från en skylt som en bild kan tas ifrån.
2. Ljus är en betydelsefull faktor för bildkvalitén. Synligheten kan bland annat påverkas av lokala ljusvariationer såsom ljusdirektionen, skuggor från andra objekt och ljusstyrkan beroende på tiden på dagen och vilken säsong på året det är. Dessutom kan skyltens färger tona bort med tiden på grund av exponering av solljus under en lång tid.
3. Skyltar kan vara skadade eller åldrade och det kan påverka formen och färgen på skylten.
4. Skyltar kan vara manipulerade på olika sätt, exempelvis med graffiti eller klistermärken.
5. Bilderna kan genomlida hastighetssuddighet och påverkan från bilens vibrationer.
6. Högkomplexa bakgrunder kan försvåra igenkänningen av bilden.
7. Objekt kan vara i vägen för skyltarna, till exempel träd, byggnader, fordon eller andra skyltar.

3. Metod och material

Metodkapitlet handlar om hur systemet byggts upp och hur teorin i föregående kapitel har implementerats för att utveckla systemet. Först sker en beskrivning av databasen som används och hur den har hanterats, därefter gås metodiken kring känneteckensextraheringen, appliceringen av PCA och korsvalidering igenom. Avslutningsvis ges en inblick i något som kallas sannolikhetssummering och hur analysberäkningarna gått till.

3.1 Databas

Inledningsvis sammansattes en databas av bilder på trafikskyltar, som därefter användes för träning och testning av systemet. Som tidigare nämnts hämtade vi databasen från INI Benchmark [1]. Databasen som laddades ned är GTSRB (The German Traffic Sign Recognition Benchmark) som specifikt är till för att användas vid behov av maskininlärning. GSTRB-databasen innehåller mer än 40 klasser av skyltar och över 50 000 bilder. Av dessa 40 klasser valde vi ut 12 stycken skyltklasser med en bred variation av kännetecken, som form och färg, varierande kvalitet och påverkade av olika störning et cetera. Selektionen av dessa skyltklasser och några exempelbilder från databasen kan ses i figur 3.1.

Huvudled:

Stoppskylt:

Varningförflerfärgsignal:

Varning för gående:

Cirkulationsplats:

Hastighetsskylt 70:

Hastighetsskylt 50:

Förbud mot omkörning:

Väjningsplikt:

Förbud mot trafik med fordon:

Förbud mot infart med fordon:

Varning för farlig kurva:

Figur 3.1, Här kan alla skyltklasser och ett smakprov på bildkvalitén observeras.

De valda skyltklasserna tillhör de fyra viktigaste skyltkategorierna, varningsmärken, väjningspliktsmärken, förbudsmärken och påbudsmärken. Det finns flera valda skyltklasser i samma kategori, vilket innebär att de har liknande färger och former och sätter krav på system och klassificeringen att hitta och urskilja liknande kännetecken. Sammanlagt har träningssetet **11 179** bilder, medan testningssetet har **2 629**. Testningssetet behöver inte vara i närheten av träningssetets storlek, eftersom det är träningsbildernas mängd och variation som till viss del kommer bestämma hur många olika kännetecken som kommer finnas i den upptränade maskinen och som går att hitta och matcha under klassificeringen. Testningssetet behöver endast vara tillräckligt stort för att systemet ska kunna kontrolleras och att träffsäkerheten ska vara pålitlig.

3.2 Hantering av data

I detta avsnitt så förklara hur den tillhandahållna data sorteras och används för olika steg av arbetet

3.2.1 Hantering av data under utveckling

Databasen från INI German Benchmark [1] är indelad i mappar för varje specifik klass. Dessa klasser innehåller 30 bilder på varje skylt från varierande avstånd och vinkel.

Vanligtvis när data av den här typen analyseras delas de slumpvis in i en träningsdel och en testningsdel med förbestämda storlekar. Detta kunde vi dock inte genomföra på denna samling av data då det skulle resulterat i att träningsdelen skulle innehålla bilder som är på samma skyltar som förekommer i testningsdelen vilket skulle lett till ett resultat som inte är enhetligt med vad som fåtts ut av kontrolldata.

För att säkerställa att data i testningssetet inte är för liknande det som finns i träningsdelen utformades därför datahanteringen på ett vis som vanligtvis inte är rekommenderat. Detta gjorde vi genom att de första 80 % av bilderna delades in i träningsdelen och de resterande 20 % i

testningsdelen. Detta gjorde att bilderna tagna på samma skylt, fast från olika avstånd och vinkel, inte förekom i både träningsdelen och testingsdelen.

3.2.2 Hantering av data för validering

När utvecklingen av systemet väl är genomförd så måste det nu testas. Vid testningen så kommer det som tidigare var uppdelat i en träningsdel och testningsdel att kombineras till en total träningsdel då det är så som data är tänkt att användas. För att validera resultatet så hämtas nu den riktiga testingdelen från INI German Benchmark som, denna del kommer hänvisas till som valideringsdelen.

Valideringsdelen ligger inte sorterad så som träningsdelen gjorde. Den är slumpvis fördelad och består av samtliga klasser som tillhandahålls. I och med att vi inte använder samtliga klasser så måste dessa först sorteras ut, för att sedan kunna använda valideringsdelen.

3.3 Förbearbetning av bilder

Då kvalitén på bilderna i många fall var förhållandevis låg så förutsatte detta en viss förbearbetning för att underlätta extrahering av kännetecken. Det som framförallt var ett problem var avsaknaden av belysning vilket ledde till att vissa skyltar kunde upplevas som i stort sett helt svarta. För att underlätta känneteckensextraheringen så behövdes differensen mellan olika pixlar ökas.

De bilder som vi använde var från början RGB(se avsnitt 2.3.1). För att bilderna skulle kunna bearbetas så krävdes det att de konverterades till NMI (se avsnitt 2.3.2). När bilderna konverterats till NMI så var det värdet på mättnaden och intensiteten som var av intresse för oss. Modifieringen av dessa två värden resulterade i att kännetecknen i bilderna blev betydligt tydligare. För att hitta den optimala kombination av dessa två förändringar så utfördes flera iterationer där en multipel till varje värde förändrades i varje iteration. Detta resulterade slutligen i att en multipel av två för både mättnaden och intensiteten i bilderna gav den högsta träffsäkerheten.

Figur 3.2: Applicering av NMI på en bild av en ”varning för gående skylt”.

3.4 Extrahering av kännetecken

Extrahering av kännetecken gjordes separat för varje sort av kännetecken och vektorerna placerades i matriser som sedan kunde användas för korsvalidering och senare för klassificering.

3.4.1 Extrahering av Gaborkännetecken

Vid extraheringen av Gabor-kännetecken var vi tvungna att välja och bestämma orienteringen och frekvensen(se avsnitt 2.1.4). I och med att dessa två parametrar varierar beroende på bildkvalité samt vilka objekt som finns i bilderna så var det mycket svårt att veta i förväg vilka värden dessa parametrar skulle ansättas till. För att vi skulle kunna välja det optimala värdet för respektive parameter så utfördes därför iterationer för både frekvensen och orienteringen.

Dessa iterationer började på värdet 1 för båda parametrarna för att sedan stegvis öka till 10. Detta gav oss totalt 100 olika känneteckensvektorer som sedan simulerades i krossvalidering för att på sådant vis resultera i 100 olika värden i träffsäkerhet. Detta ledde alltså till att det var säkerställt att vi valt de mest optimala parametervärdena för att på så vis optimera Gabor-kännetecknens bidrag till den totala träffsäkerheten.

3.4.2 Extrahering av HOG-kännetecken

Extraheringen av HOG-kännetecken krävde även den viss inställning och viktning av parametrarna. Som nämns i teorin(se avsnitt 2.1.3) så finns det olika storlekar på cellerna. Mindre storlekar ledde till ett större antal kännetecken för varje bild, vilket bidrog till en bättre träffsäkerhet när vi använde dessa kännetecken för maskininlärning.

3.5 Applicering av PCA

Efter extraheringen av HOG-kännetecken samt Gabor-kännetecken så hade vi fått fram ett antal matriser fyllda med känneteckensvektorer. Dessa matriser hade väldigt stora dimensioner, vilket gjorde fortsatt analysering och arbete väldigt tidskrävande. För att lösa detta problem använde vi oss av PCA-metoden som berörts i teorin (se avsnitt 2.6) och följde dess metodik steg för steg.

Först beräknade vi ut medelvärdet för alla känneteckenspunkter i samtliga känneteckensvektorer. Därefter subtraherades de ursprungliga känneteckenspunkterna med deras medelvärden som i sin tur skapade vår kovariansmatris. Vi kunde sedan tillämpa funktionen *eig* i MATLAB för att få ut egenvektorerna och egenvärdena av kovariansmatrisen. En summering av egenvärdena utfördes när vi därefter fick fram en ny förkortad slutgiltig egenvektorn genom att ansätta ett tröskelvärde för att iterera fram vilka egenvärden som hade ett tillräckligt bra värde för att vara med i vektorn. Avslutningsvis multiplicerade vi den förkortade egenvärdesvektorn med den ursprungliga matrisen för att projektera fram den nya grovt reducerade matrisen.

3.6 Korsvalidering

Vi valde av praktiska skäl att använda den inbyggda korsvalidering som finns i den SVM vi valt, LIBSVM [2]. För att säkerställa att ett tillräckligt stort intervall täcks in under korsvalideringen ansattes $C = 2^i$ och $\gamma = 2^j$. Spannet för i och j ansattes sedan att gå från -10 till 10 med steglängden 1. Detta leder till att både C och γ gick från 2^{-10} till 2^{10}. Totalt gav detta en matris med storleken $21x21$ där den optimala parameterkombinationen hittades.

I och med att den procentuella ökningen på parametrarna var så pass stor när exponentformen användes så behövdes intervallet minskas ner. Det nya intervallet var därmed [i-1:i+1] respektive [j-1:j+1] där steglängden också halverades sedan tidigare. Detta ledde till att ett än noggrannare parameterval kunde genomföras. Denna intervall- samt steglängdsförändring applicerades än en gång för att få fram det optimala parametervalet.

3.7 Sannolikhetssummering

När samtliga kännetecken har extraherats med hjälp av Gaborfilter och HOG har vi i sin tur tränat alla var för sig i vår SVM. Varje SVM-körning genererar en vektor, en matris och en skalär:

- **Klassifikation**(vektor): vilken klass som den tränade vektormaskinen anser att skylten tillhör. Dessa klassifikationer kommer sedan att användas tillsammans med uppskattningen av svarsäkerheten för att bestämma vilken klass som ska ansättas.
- **Uppskattning av svarssäkerhet**(matris): hur säker vektormaskinen anser att den är över respektive klass i klassifikation. De olika skalärerna i vektorn varierar vanligtvis mellan 0 och 1, vilket ger ett brett spektrum som sedan kan användas vid viktningen genom att säkerheten för varje klass för respektive kännetecken summeras. Denna summering leder då till ett gemensamt beslut för vilken klass som ska ansättas.
- **Träffsäkerhet**: hur stor procent av skyltarna som klassificerades korrekt. Denna träffsäkerhet använder vi enbart under utvecklingen av algoritmen, och är inte av intresse i programmet.

För att fatta ett gemensamt beslut som är baserat på samtliga kännetecken som valts, används uppskattningsmatrisen från vardera SVM. De olika klasserna i de olika matriserna summeras för att ge en gemensam uppskattning för vardera klass. Anledningen till att detta görs är att med så stor sannolikhet som möjligt kunna klassificera bilderna korrekt.

3.8 Analysberäkning

När klassificeringen är genomförd så jämför vi den slutgiltiga klassificeringsvektorn med vektorn bestående av de korrekta klasserna. Här beräknas sedan en träffsäkerhet för att på så vis ge oss ett överskådligt resultat.

Vid analysering och tolkning av resultat så är det främst fyra faktorer som står i fokus; träffsäkerhet, falskalarmsratio (FAR), sensitivitet och noggrannhet. Träffsäkerhet är ett resultat som fås utan

problem medan FAR, sensitivitet och noggrannhet är tre viktningar som är utformade för binära system.
Sensitiviteten beräknas genom att de skyltar som har klassificerats som klass i, och som tillhör klass i, divideras med summan av alla skyltar som tillhör klassen i. I ett okänsligt system så är detta värde så nära 1 som möjligt.

$$\text{Sensitivitet} = \frac{\text{Positiva}_{\text{rätt}}}{\text{Positiva}_{\text{totalt}}}$$

Noggrannheten tas fram genom att samtliga skyltar som har klassificerats som någon annan klass än i, och som tillhöra någon annan klass än i, divideras med det totala antalet skyltar som inte tillhör klassen i. I ett noggrant system så skall detta värde vara så nära 1 som möjligt.

$$\text{Noggrannhet} = \frac{\text{Negativa}_{\text{rätt}}}{\text{Negativa}_{\text{totalt}}}$$

Falsklarmsration beräknas genom att samtliga skyltar som har klassificerats till i, och som inte tillhör klass i, divideras med samtliga skyltar som har klassificerats felaktigt.

$$\text{FAR} = \frac{\text{Positiva}_{\text{fel}}}{\text{Fel}_{\text{total}}}$$

För att eliminera problemet med att systemet inte är binärt så kommer FAR, sensitivitet och noggrannhet räknas ut separat för varje klass. I och med att de räknas ut för varje klass så kan samma formel som för binära system appliceras. Detta ger 12 olika värden för respektive faktor som sedan kan användas för att beräkna ett medelvärde. Detta medelvärde kan sedan användas för att analysera hur stabilt systemet är.

4. Resultat och diskussion

Efter fem månaders arbete kan nu resultatet av många timmars MATLAB-simuleringar presenteras. För att resultatet ska vara begripligt är det viktigt att reda ut begreppen som används. Falskalarmsratio (FAR) beskriver hur stor andel av testningsbilderna för varje klass som blir falskt positiva, alltså de som tilldelas fel klass. Exempelvis är FAR för klass *ett* andelen av testningsbilderna som felaktigt blivit tilldelade klass 1 i klassificeringen. Sensitivitet och noggrannhet är mått per klass på hur väl systemet klassificerar rätt och beskriver hur många som korrekt blir klassade som positiva respektive hur många som korrekt blir klassade som negativa. Sensitiviteten kan också ses som träffsäkerheten för den specifika klassen i fråga.

4.1 HOG

För att visa ett exempel på ett av alla de val som empiriskt gjorts under arbetets gång visar tabell 4.1 träffsäkerheten för HOG med de olika cellstorlekarna som testats. Tabellen visar tydligt att den minsta cellstorleken $2x2$ ger det enskilt bästa resultatet med 85.67 % träffsäkerhet. Detta var väntat, men för att inte lämna något åt slumpen testades också cellstorlekarna $4x4$ och $8x8$.

HOG Cellsstorlek	Träffsäkerhet
2x2	85.67
4x4	83.23
8x8	73.64

Tabell 4.1: Träffsäkerheten för olika HOG-cellstorlekar.

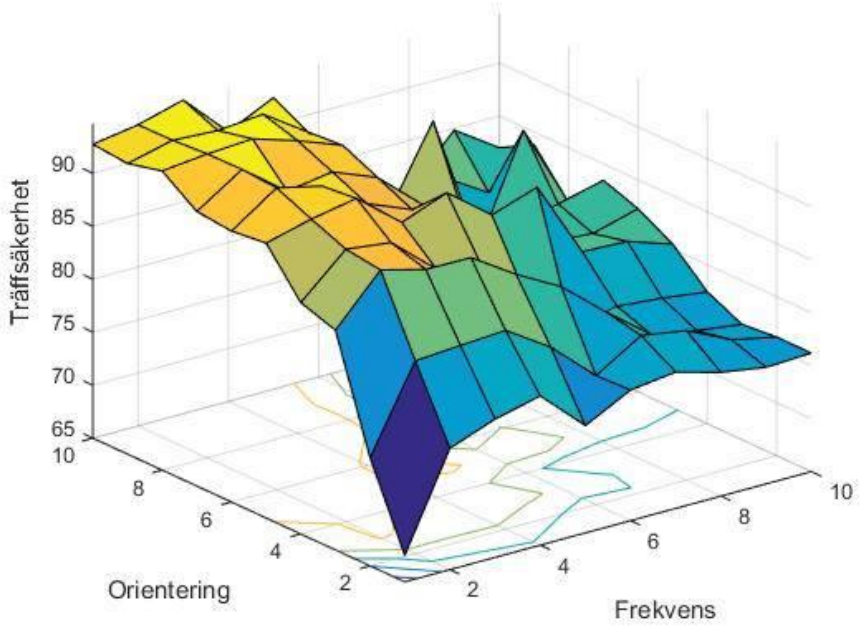
Med cellstorleken vald kan nu resultatet för simuleringar med kännetecknet HOG presenteras i tabell 4.2. Resultatet visar alltså falskalarmsratio, sensitivitet samt noggrannhet per klass.

Klass	1	2	3	4	5	6	7	8	9	10	11	12
FAR	0.01882	0.0004	0.0031	0.0033	0.0039	0.0537	0.0410	0.0120	0.0171	0.0074	0.001174	0.0006
Sens	0.8869	0.8	0.9222	0.8133	0.7111	0.8424	0.8333	0.8291	0.9666	0.6666	0.9222	0.6666
Nog	0.9811	0.9995	0.9968	0.9966	0.9960	0.9462	0.9589	0.9879	0.9828	0.9925	0.9988	0.9993

Tabell 4.2: Resultat för HOG.

4.2 Gabor

Efter att ha gjort långa och resurskrävande simuleringar kunde tillslut en optimal kombination av orientering och frekvens hittas till kännetecknet Gabors systems. Figur 4.3 visar en tredimensionell visualisering av Gabors träffsäkerhet med avseende på testningssetet, vilket inte ska blandas ihop med det valideringsset som slutligen används för att testa HOG och Gabor.



Figur 4.3: 3D-plot mellan träffsäkerhet, orientering och frekvens.

Som figur 4.3 visar avtar träffsäkerheten runt orientering 10 och frekvens 3, och ökningar av orientering samt frekvens visar försumbara förbättringar. Att fortsätta simuleringar med högre värden är extremt tids- och resurskrävande och har därför valts att inte göras. Bakom värdena som ses i figur 4.3 ligger ett hundratal timmars simuleringar, och desto större orientering, desto mer tids- och resurskrävande blir simuleringarna. Med dessa värden kan nu resultatet för Gabor presenteras, se tabell 4.4.

Klass	1	2	3	4	5	6	7	8	9	10	11	12
FAR	0.0050	0	0.0029	0.0015	0.0081	0.0179	0.0142	0.0045	0.0043	0.0052	0.0018	0
Sens	0.9579	0.9888	0.9777	0.9733	0.8222	0.95	0.9226	0.9854	0.9888	0.8476	0.8194	0.8166
Nog	0.9949	1	0.9970	0.9984	0.9918	0.9820	0.9857	0.9954	0.9956	0.9947	0.9981	1

Tabell 4.4: Resultat för Gaborfilter.

4.3 Slutligt resultat

Med hjälp av sannolikhetssummeringen, som förklaras i avsnitt 3.7, kan nu ett totalt resultat fås fram, se tabell 4.5 samt tabell 4.6 för träffsäkerheten för de respektive kännetecknen.

Klass	1	2	3	4	5	6	7	8	9	10	11	12
FAR	0.005343	0.0002	0.0004	0.0015	0.0046	0.0234	0.0162	0.0041	0.0051	0.004308	0.0007	0
Sens	0.9594	0.9703	0.9666	0.9466	0.8111	0.9515	0.9053	0.9625	0.9930	0.838095	0.9277	0.7666
Nog	0.9946	0.9997	0.9995	0.9984	0.9953	0.9765	0.9837	0.9958	0.9948	0.995692	0.9992	1

Tabell 4.5: Resultat för totalen där både Gaborfilter och HOG används.

Metod	Träffsäkerhet
HOG	85.87
Gabor	94.16
Total	94.22

Tabell 4.6: Träffsäkerhet för respektive metod samt den totala träffsäkerheten.

Det kan konstateras att träffsäkerheten är relativt jämn för samtliga klasser förutom klass fem och tolv där endast 81 % samt 77 % av bilderna blir korrekt klassificerade. Dessa klasser drar alltså ner den totala träffsäkerheten, men en titt på tabell 4.7 visar att båda klasserna har relativt få bilder sett till det totala antalet, med 90 respektive 60 bilder i valideringssetet. De försämrar alltså inte resultatet så mycket som man skulle kunna tro vid en första anblick på sensitiviteten.

Klass	1	2	3	4	5	6	7	8	9	10	11	12
Träning	2099	779	599	539	359	1979	2249	1469	2159	629	1109	210
Validering	690	270	180	150	90	660	750	480	720	210	360	60

Tabell 4.7: Antalet tränings- respektive valideringsbilder för varje klass.

Att just dessa två klasser har så låg träffsäkerhet kan bero på en mängd variabler, men troligt är att kvaliteten på en uppsättning bilder i antingen träningssetet eller valideringssetet är mycket dålig. Eftersom det är trettio bilder per skylt, skulle till exempel en skylt kunna vara tagen under mycket dåliga förhållanden, kanske i totalt mörker, vilket leder till att samtliga bilder på skylten är svåra att känna igen och klassificera. Vidare kan ses att de klasser med en stor urvalsmängd är de som har högst sensitivitet och alltså bäst blir klassificerade. Se till exempel klass ett, sex, åtta och nio. Detta kan tolkas som att systemet fungerar bra och kännetecknen valts rätt, och anledningen till att högre total träffsäkerhet inte uppnås helt enkelt är för att urvalsmängden på bilder är för få. Detta kan givetvis inte säkerställas utan att först testas, men resultatet pekar helt klart åt detta håll.

4.4 Tidigare systemstrukturer

I ett tidigt skede av arbetet togs ett klassifikationsträd fram vars syfte var att beskriva systemets arbetsprocess. Trädet var uppbyggt så att vid varje förgrening skulle en del av klasserna försvinna, så systemet tillslut kom fram till en specifik klass. Tanken var att inte alla steg skulle använda sig av SVM för att klassificera bilden, och de som skulle använda SVM skulle göra det binärt. Till exempel skulle alla cirkulära skyltar i första förgreningen skiljas från alla ickecirkulära, det fanns i detta steg alltså bara två klasser.

Figur 4.8: Figuren visar ett exempel på ett klassifikationsträd som vi byggde upp. Steg ett som börjar på toppen av trädet skulle klassificera om det var en cirkelär eller ickecirkulär skylt. Steg två skulle på vänster sida klassificera om det var en röd eller blå cirkulär skylt och på höger sida om trädet om skylten var triangulär formad eller inte.. De sista stegen i trädet skulle klassificeras skylten med hjälp av mer avancerade kännetäckensmetoder, såsom HOG, Gaborfilter och så vidare, i samband med SVM.

En annan klassificeringsmetod som utretts och tänkts användas är färgigenkänning, där systemet till exempel skulle separera röda skyltar från blåa. Avsikten var att öka differensen mellan färgerna i bilderna genom att konvertera bilderna från RGB till NMI. Då bilderna var i NMI-format så kunde värdet på nyansen samt värdet på mättnaden ökas innan bilderna sedan skulle konverteras tillbaka till RGB-format. Ett program skulle gå igenom varje pixelvärde för rött, grönt och blått. Den skulle räkna ut antalet och den färgen med störst majoritet skulle klassificera skyltfärgen. Med en MATLAB-algoritm fick vi slutligen en träffsäkerhet på cirka 90 % när vi testade klassifikationen. Anledningen till att vi inte kunde få ett högre värde berodde främst på bildernas usla kvalitet. Eftersom denna sortens klassifikation inte exakt klassificerar vilken skyltklass skylten tillhör utan

bygger på fortsatt klassificering bidrar det till att en träffsäkerhet på 90 % inte är acceptabel eftersom felet multipliceras och växer till nedåtgående trädgrenar.

Figur 4.9: Exempel på hur applicering av NMI för att hitta röda pixlar. Höger bild är standardbilden medan vänster bild har applicerats med värdeskalning i NMI.

5. Slutsats

Som nämndes i diskussionen så har antalet träningsbilder en mycket stor inverkan på resultatet. Den huvudsakliga förbättringen skulle kunna vara att öka antalet träningsbilder för att säkerställa att det finns tillräckligt stor variation av kännetecken för varje klass.

Utöver antalet bilder för varje klass så bör det även observeras att Gabor och HOG har liknande resultat gällande sensitivitet och noggrannhet för varje klass. Det skulle vara önskvärt att använda ytterligare kännetecken som bygger på metoder som skiljer sig mer från Gabor och HOG. Vi har testat flera olika känneteckensextraheringsmetoder utan att få en träffsäkerhet som har bidragit till en bättre total träffsäkerhet vilket därför har lett till att de har uteslutits.

I detta fall ger användningen av flera kännetecken ett minimalt bättre resultat än användningen av ett kännetecken. Sannolikhetssummeringen ger, som visats tidigare, en högre total träffsäkerhet än Gabor, även om HOG har en träffsäkerhet som ligger under denne. Den totala träffsäkerheten ökar dock endast med 0.06 % vilket gör att sannolikhetsummeringen verkar smått överflödig. I bilaga A går det dock att se en tabell över tidigare värden över träffsäkerheten innan förbearbetningen. Gabor och HOG har här ett mer liknande värde än de givna i tabell 4.6 där skillanden är närmare 9 procentenheter. Vi kan därmed se att när träffsäkerheten för de båda kännetecknen är närmare varandra kan sannolikhetssummeringen vara till stor hjälp för att öka totalen. Metoden kan alltså vara användbar vid kombinering av andra kännetecken.

Avslutningsvis kan slutsatsen dras att Gaborfilter är den mest givande metoden som ger oss en träffsäkerhet på 94.16 % och 94.22 % med hjälp av HOG.

Källor

[1] German traffic sign dataset:
2015-02-02
http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset#Downloads

[2] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin,
"A Practical Guide to Support Vector Classification"
http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[3] Isabelle Guyon and André Elisseeff,
"Introduction to Feature extraction"
http://clopinet.com/fextract-book/IntroFS.pdf

[4] Markus Mathias, Radu Timofte, Rodrigo Benenson, and Luc Van Gool,
"Traffic Sign Recognition – How far are we from the solution?"
http://rodrigob.github.io/documents/2013_ijcnn_traffic_signs.pdf

[5] R. Kimmel Fellow, IEEE, C. Zhang, A. Bronstein Member, IEEE, M.Bronstein Member, IEEE,
"Are MSER features really interesting?"
http://vista.eng.tau.ac.il/publications/KimCuiBroBroPAMI09.pdf

[6] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas,
F. Schaffalitzky, T. Kadir, and L. van Gool, "A comparison of affine
region detectors"
http://www.robots.ox.ac.uk/~vgg/research/affine/det_eval_files/vibes_ijcv2004.pdf

[7] Navneet Dalal and Bill Triggs
"Histograms of Oriented Gradients for Human Detection"
http://vision.stanford.edu/teaching/cs231b_spring1213/papers/CVPR05_DalalTriggs.pdf

[8] Hasan Fleyeh
"Traffic and Road Sign Recognition"
http://www.diva-portal.org/smash/get/diva2:523372/FULLTEXT01.pdf

[9] Support Vector Machines, Statsoft
2015-03-15
http://www.statsoft.com/textbook/support-vector-machines

[10] A.Zisserman
"Lecture 2: The SVM classifier"
http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

[11] G. Scott Owen
“Hue, Saturation, Value Color Model”
2015-04-04
https://www.siggraph.org/education/materials/HyperGraph/color/colorhs.htm

[12] Vägmärken, Transportstyrelsen
2015-04-05
https://www.transportstyrelsen.se/sv/vagtrafik/Vagmarken/

[13] Sebastian Raschka
”About Feature Scaling and Normalization”
2015-04-20
http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

[14] Selim Aksoy and Robert M. Haralick
“Feature Normalization and Likelihood-based Similarity Measures for Image Retrival”
http://www.cs.bilkent.edu.tr/~saksoy/papers/prletters01_likelihood.pdf

[15] Joni-Kristian Kamarainen
”Gabor Feature in Image Analysis”
http://personal.lut.fi/users/joni.kamarainen/downloads/publications/ipta2012.pdf

[16] Daugman, J. G
“Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters." J. Optical Society of America A, vol. 2, no. 7, pp. 1160-1169, July 1985.

[17] R. Timofte, K. Zimmermann, and L. Van Gool,
“Multi-view traffic sign detection, recognition, and 3D localization,”
Machine Vision and Applications, December 2011.

[18] Felix Măriut, Cristian Foşlău, Manuel Avila and Daniel Petrişor
“Detection and Recognition of Traffic Sign Using Gabor Filters”
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6043668

[19] Oncel Tuzel, Fatih Porikli and Peter Meer
“Region Covariance: A Fast Descriptorfor Detection and Classification”
http://coewww.rutgers.edu/riul/research/papers/pdf/covfeat.pdf

[20] Lindsay I Smith

”A tutorial on Principal Components Analysis”
http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

[21] Justin Domke and V.Shiv Naga Prasad
“Gabor Filter Visualization”
http://wwwold.cs.umd.edu/class/spring2005/cmsc838s/assignment-projects/gabor-filter-visualization/report.pdf

Matlabkodsreferens:
M. Haghighat, S. Zonouz, M. Abdel-Mottaleb
"Identification Using Encrypted Biometrics,"
Computer Analysis of Images and Patterns, Springer Berlin Heidelberg,
pp. 440-448, 2013.

Bilaga

Bilaga A

Träffsäkerhet	
HOG	85.67
Gabor	90.43
Total	91.54

Figur A.1: Tabell över tidigare värden utan förbearbetning