CHALMERS | GÖTEBORGS UNIVERSITET

Statistisk bildanalys av handgester för människa-dator-interaktion

Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Harald Freij
Viktor Nilsson
David Samuelsson
Simon Sigurdhsson

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet
Göteborg 2011

Statistisk bildanalys av handgester för människa-dator-interaktion

Kandidatarbete i matematik inom civilingenjörsprogrammet
Teknisk fysik vid Chalmers
Harald Freij Viktor Nilsson

Kandidatarbete i matematik inom civilingenjörsprogrammet
Teknisk matematik vid Chalmers
David Samuelsson Simon Sigurdhsson

Handledare: Mats Rudemo
Magnus Röding
Examinator: Carl-Henrik Fant

Institutionen för matematiska vetenskaper
Chalmers tekniska högskola
Göteborgs universitet
Göteborg 2011

Sammanfattning

Den här rapporten beskriver arbetet med och teorin bakom skapandet av ett program som kan identifiera ett antal olika statiska gester med stor noggrannhet. Rapporten beskriver även metoder för att med hjälp av dolda Markovmodeller (HMM) identifiera dynamiska gester.

Klassificeringsmetoderna som används är helt statistiska, och använder varken färdiga handmodeller eller artificiella neurala nätverk. Handidentifieringen kräver att användaren bär långärmad tröja, men i rapporten beskrivs även metoder för att identifiera handleder så att detta krav kan uteslutas.

Analysen utförs på inspelade filmer, men realtidsanalys har implementerats med tillfredställande resultat.

Med hjälp av Gaussian Mixture Models för hudidentifiering och kNN-klassificering för gestigenkänning kan 10 statiska gester identifieras med 91.4 % korrekt klassificering. För analys av dynamiska gester saknas resultat, då samtliga metoder inte är implementerade.

Abstract

This report describes the work with and the theory behind the implementation of a program that can identify a number of different static gestures with high accuracy. The report also describes methods which use hidden Markov models (HMM) to identify dynamic gestures.

The classification methods used are completely statistical, and use neither prefabricated hand models nor artificial neural networks. Hand identification requires the user to wear a shirt with long sleeves, but methods to identify wrists are described in the report, enabling relaxation of this requirement.

The analysis is done on prerecorded video clips, but real-time analysis has been implemented with satisfying results.

With Gaussian Mixture Models for skin identification and kNN classification for gesture recognition ten static gestures can be identified with an accuracy of 91.4 %. For analysis of dynamic gestures results are missing, as all methods are not implemented.

Förord

Vid skrivandet av denna rapport har alla fyra författarna med likvärdiga insatser redigerat och sammanfogat texten till en helhet. Även om varje text ursprungligen har skrivits av en person har alla haft betydande medverkan i samtliga avsnitt. Vi vill ändå nämna vem som ligger bakom några av de större avsnitten. De inledande delarna om färgrymder, avsnittet om morfologiska operationer och genomgången av de olika egenskaperna skrevs först av Simon Sigurdhsson. Sannolikhetsfördelningar för hudfärg, isolering av handen, samt våra gester har främst förklarats av Harald Freij. Avsnitten om dolda Markovmodeller kan tillskrivas David Samuelsson, medan delarna om klassificering och närmsta grannar samt datainsamling främst skrivits av Viktor Nilsson.

Rapporten är skriven på svenska, men eftersom ämnet förutsätter ett flitigt nyttjande av fackterminologi har vi i vissa fall tvingats välja engelskspråkiga termer. Detta har vi endast gjort i de fall då en svensk översättning uppenbart hade varit svår att förstå.

Under arbetets gång har en dagbok med detaljer kring arbetsprocessen förts. Denna ligger delvis till grund för betygssättningen i kandidatarbeteskursen. Den innehåller samtidigt information om sidospår som ej tagits upp i rapporten, och kan tillsammans med versionshanteringsloggar ge en inblick i hur mycket tid olika delar av arbetet har tagit samt vem som har arbetat mest med vad. I stort kan sägas att ingen har haft några specifika ansvarsområden förutom att Simon Sigurdhsson tillsammans med Viktor Nilsson haft huvudansvaret för versionshantering och LaTeX-relaterade frågor. Alla har precis som i rapporten varit lika delaktiga i programmeringen.

Tack till

Vi skulle vilja tacka följande personer och organisationer för deras ovärderliga hjälp i arbetet:

Mats Rudemo och *Magnus Röding* för stöd och tips i arbetet och skrivprocessen, samt *Hans Malmström* för värdefulla synpunkter på den språkliga utformningen.

Våra tålmodiga försökspersoner *Emma Kjellson, Anna Nilsson, Jonatan Rydberg, Dündar Göç, Susanne Schilliger Kildal, Gustav Hansson, Fredrik Johnsson, Josefin Lövmark, Jakob Friman* och *Christoffer Johansson* för era hjälpande händer.

Versionshanteringstjänsterna *Bitbucket* och *Mercurial* utan vilka arbetet skulle befinna sig i fullständigt kaos.

Tack även till stödjande vänner och givmilda vädergudar.

Innehåll

Kapitel 1

Inledning

Moderna och innovativa gränssnitt för interaktion med datorer har länge varit en vision inom filmindustrin. Redan i mitten av 90-talet hänfördes tittare av 3D-gränssnitt i filmen Hackers, och i början av 2000-talet visade scener i Minority Report hur Tom Cruise styr ett operativsystem endast med sina händer med hjälp av två trådlösa handskar.

Denna framtidsvision kan ligga mycket närmare än vad Spielberg förväntade sig. Redan i början av 90-talet gjordes försök att tolka mänskliga rörelser i bildsekvenser (Yamato et al. 1992), och sedan dess har intresset i forskarvärlden bara ökat. Handgester för kommunikation med datorer är alltså inte längre något som hör hemma i filmens värld, utan kan snart vara verklighet.

Det finns två sätt att uppnå denna vision, den för användaren mer krävande metoden med sensorfyllda handskar och den mer lättillgängliga kamerabaserade metoden som använder data från exempelvis vanliga webbkameror. Den senare har fördelen att gemene man kan utnyttja lösningen utan några extra hårdvarukrav.

För tillvägagångssättet där en vanlig kamera används är det mest grundläggande problemet att över huvud taget kunna identifiera en människas hud i bilder, och det problemet har många lösningar. Metoderna förfinas hela tiden, men i stort har det inte hänt något revolutionerande sedan början av 2000-talet (Sebe et al. 2004, Kruppa et al. 2002, Albiol et al. 2001, Brand och Mason 2000), med några få undantag (t.ex. Hassanpour et al. 2008, Khan et al. 2010). När man väl löst detta problem måste datorn även kunna tolka handens form och rörelser. Detta är ett svårare problem, men har också behandlats flitigt (Pavlovic et al. 1997, Garg et al. 2009, Sánchez-Nielsen et al. 2004, Zabulis et al. 2009).

Trots denna uppmärksamhet från forskarvärlden, och på senare tid även den kommersiella sektorn (t.ex. Microsoft Kinect till TV-spelskonsolen Xbox), fortsätter gestigenkänning att vara något som kräver extern och ofta dyr hårdvara

eller mycket kontrollerade miljöer för att fungera bra. Detta problem kan endast lösas genom att implementera ett videobaserat system som är tillräckligt robust för att kunna identifiera handgester i mycket varierande miljöer.

Denna rapport behandlar både hudigenkänning och gestigenkänning, och ger en lösning på gestigenkänningsproblemet i dess enklaste form, när datorn endast förväntas känna igen statiska gester. Detta görs med hjälp av Gaussian Mixture Models för hudklassificering och kNN-metoden (k närmsta grannar) för tolkning av handgester utifrån en mängd egenskaper som extraheras ur bilddata. Vidare lägger den en teoretisk grund för att även kunna känna igen rörliga gester med hjälp av så kallade dolda Markovmodeller.

1.1 Avgränsning

Även om grundproblemet i korthet kan formuleras som en mycket generell gestigenkänning har denna rapport begränsats något. Först och främst begränsas datamängden till enkel videobaserad indata, d.v.s. varken sensorhandskar eller stereoskopiska kameror har använts eftersom detta gör implementeringen mer tillgänglig. Metoder baserade på modeller av virtuella händer har helt undvikits i arbetet.

De egenskaper som identifieras i handen (se kapitel 2.3) bestäms helt utifrån en binärbild av handen, och hänsyn har därför inte tagits till handens inre struktur. Egenskaper som bygger på konturer i handens inre samt handens kurvatur har inte heller studerats.

All analys och inlärning har skett på inspelade filmer. Metoderna har i ett mycket sent skede testats även på realtidsvideo, dock utan kvantitativa resultat.

Kapitel 2

Teori

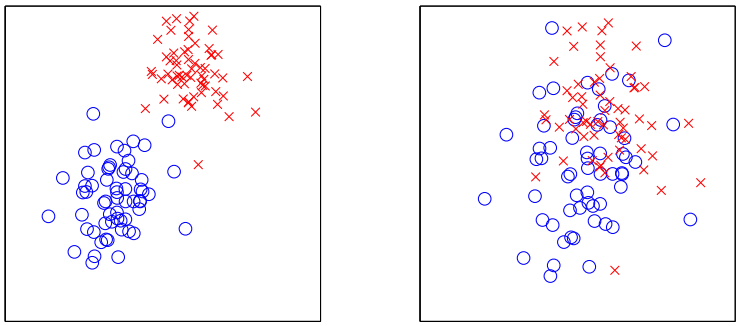
Till grund för metoden som används i rapporten ligger en stor mängd teori, så som statistiska metoder vilka ligger till grund för hudklassificering, klassificeringsmetoder för gestigenkänning, enklare bildbehandlingsmetoder och dolda Markovmodeller för klassificering av dynamiska gester. Dessa teorier presenteras här så att de praktiska aspekterna i kapitel 3 har en grund att vila på.

2.1 Statistisk klassificering

Klassificering är ett centralt statistiskt begrepp inom maskininlärning. Det handlar om att klassificeraren ska göra en kvalificerad gissning för hur observerad data ska grupperas, baserad på någon typ av ”inlärningsprocess”. Man skiljer på modellbaserade och modellfria metoder, där de modellbaserade bygger på modeller av de bakomliggande system som ska analyseras, medan de modellfria kan ses som mer oberoende av sammanhang (Hastie et al. 2009). Man skiljer också på övervakad respektive icke-övervakad inlärning, där man i det övervakade fallet har etiketter på den träningsdata man stoppar in, så att man för varje objekt kan säga om det klassificerats rätt eller fel.

Indata till klassificeraren är någon mängd av objekt med någon uppsättning egenskaper (features) som kan mätas. I egenskapsrummet önskar man då att objekten separeras i olika kluster som stämmer överens med deras verkliga klass. I vissa problem är resultatet binärt, vilket exempelvis gäller i hudisoleringsproblemet. I andra fall väljs bland många klasser, vilket gäller gestklassificeringen. Svårigheten kan förtydligas något med hjälp av figur 2.1 på följande sida.

Tre klassificerare kommer att beröras, den modellfria prototypbaserade kNN samt de två modellbaserade Gaussian Mixture Models och dolda Markovmodeller.



Figur 2.1: *Det är enkelt att se var ett objekt hör hemma i den vänstra figuren, där de två klasserna är väl separerade. I den högra figuren kan man också se skillnad på objekt, men det är nu mycket svårt. Axlarna motsvarar två godtyckliga egenskaper. En annan uppsättning egenskaper kan ge en tydligare separering av klasserna.*

2.2 Klassificering av hudfärg och urskiljning av händer

Det första problemet vid visuell gestigenkänning är att kunna se handen, och för att kunna göra detta måste bilden först behandlas. Bildpunkterna måste klassificeras som hud eller ickehud, och därefter måste bildområdet med hud i sin tur klassificeras som en hand eller som övrig hud.

För att kunna isolera hud i en bild är det lämpligt att börja med en transformation av bilden till en passande färgrymd, varpå statistiska metoder kan användas för att markera bildpunkterna som hud respektive ickehud. Det andra problemet, att identifiera handen bland hudobjekten i bilden kan göras exempelvis genom subtraktion av objekt som matchar ett ansiktes form. En enklare metod är att ta det objekt som ligger längst till vänster i bilden, något som förklaras mer utförligt i 3.2.2.

2.2.1 Färgrymder

En färg kan matematiskt beskrivas som en vektor i ett färgrum, vars basvektorer är grundfärgerna i ett färgsystem. För att kunna representera alla färger behöver man ett tredimensionellt färgrum, men i vissa situationer kan det räcka — kanske till och med vara fördelaktigt — med ett tvådimensionellt färgrum (Kakumanu et al. 2007).

Vilken färgrymd som är lämpligast för hudklassificering har inget entydigt svar. RGB-färgrymden (röd, grön, blå) har använts av bland annat Lockton och Fitzgibbon (2002) och Sebe et al. (2004), och ter sig lämplig eftersom det är denna

färgrymd som används i datorer. Perceptuella färgrymder, såsom HSL, separerar kulör, mättnad och ljusintensitet. De har inte rönt någon större framgång dels då deras definierande transformationer innehåller singulariteter, dels då de är prestandamässigt ofördelaktiga.

Ortogonala färgrymder har däremot nått utbredd användning för hudklassificering (Hsu et al. 2002, Elmezain et al. 2008, Hassanpour et al. 2008) eftersom de både separerar kulör och mättnad från ljusstyrka, och är affina avbildningar av RGB. En ortogonal färgrymd som har visat sig särskilt bra är YC_bC_r (Kakumanu et al. 2007).

2.2.2 Färgrymden $\mathbf{YC_bC_r}$

YC_bC_r-rymden är en ortogonalisering av RGB-rymden genom en affin transformation, vilken resulterar i en basvektor som representerar ljusstyrka (Y) och två som representerar färg (chroma), C_b och C_r. Enligt internationell standard (ITU 2007) ges transformationen av

$$\begin{aligned} Y &= 16 + (65.481R + 128.553G + 24.966B) \\ C_b &= 128 + (-37.797R - 74.203G + 112.0B) \\ C_r &= 128 + (112.0R - 93.786G - 18.214B) \end{aligned}$$

där R, G, och B är värden mellan 0 och 255 som representerar färgens värde i RGB-rymden.

2.2.3 Sannolikhetsfördelning för hudfärg

För att skilja bildpunkter med hud från bakgrunden, kan flera olika statistiska metoder appliceras. Man har till exempel gjort försök med ”random forests” (Khan et al. 2010), bayesianska nätverk (Sebe et al. 2004), generativa modeller (Kruppa et al. 2002) och ”fuzzy arithmetics” (Shang et al. 2010). En större undersökning av metoder för huddetektion gjordes av Kakumanu et al. (2007).

En enklare metod som nått relativt stor framgång är den som baseras på Gaussian Mixture Models (Elmezain et al. 2008, Hassanpour et al. 2008), en modell där bildpunktens färg antas bero stokastiskt på om bildpunkten föreställer hud eller inte.

Färgen antas då vara en multivariat stokastisk variabel, vars fördelning är en blandning av normalfördelningar. Fördelningens parametrar antas bero på om bildpunkten motsvarar hud eller inte. För att kunna hantera olika hudfärger kan en uppsättning av olika fördelningar för respektive hudfärg användas. Den totala sannolikhetsfördelning är sedan en linjärkombination av dessa, där vikterna motsvarar a priori-fördelningen mellan olika hudfärger. Detsamma kan göras med sannolikhetsfördelningen för bildpunkter som inte innehåller hud.

För att bestämma väntevärdesvektorn och kovariansmatrisen i en uppskattad normalfördelning krävs ett antal datapunkter, med vars hjälp parametrarna kan skattas med maximum likelihood.

Sannolikhetstätheten ges i d dimensioner allmänt av

$$p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)/2},$$

där $\mu \in \mathbb{R}^d$ är väntevärdesvektorn och $\Sigma \in \mathbb{R}^{d\times d}$ är kovariansmatrisen, definierad genom sina element

$$\sigma_{i,j} = \mathrm{Cov}[X_i,X_j] = \mathrm{E}[(X_i - \mu_i)(X_j - \mu_j)]].$$

Med komponenterna C_b och en C_r är $d = 2$. Med N datapunkter skattas parametrarna med

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$$

$$\hat{\Sigma} = \frac{1}{N-1}\sum_{i=1}^{N} (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T$$

För att ytterligare förbättra fördelningarna kan de antas vara en linjärkombination av flera normalfördelningar. En iterativ metod för optimering av sådana fördelningar beskrevs av Dempster et al. (1977), och användes till de sannolikhetsfördelningar som ges av Elmezain et al. (2008). Ett exempel på hur fördelningarna kan se ut längs en linje i C_bC_r-planet ses i figur 2.2 på nästa sida.

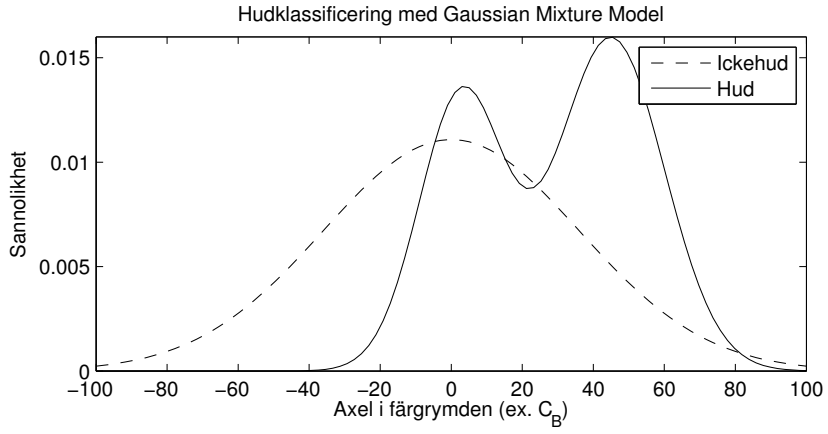
För varje bildpunktsfärg kan med dessa fördelningar sannolikheten bestämmas, för att färgen ska förekomma när bildpunkten föreställer hud respektive ickehud. Det är rimligt att klassa en bildpunkt som hud om sannolikheten för att bildpunkten föreställer hud är större än sannolikheten att den inte gör det givet dess färg, d.v.s. om

$$\frac{\mathrm{P(hud|färg)}}{\mathrm{P(ickehud|färg)}} > 1.$$

Ibland kan det vara önskvärt att kräva att kvoten ska vara ännu större, eftersom det är värre att klassa ickehud som hud (och då lägga handen i det som egentligen är bakgrund) än att klassa hud som ickehud (och få en för liten hand). Man ställer därför istället kravet

$$\frac{\mathrm{P(hud|färg)}}{\mathrm{P(ickehud|färg)}} > c,$$

där konstanten c kan anpassas för att få så bra resultat som möjligt.



Figur 2.2: *Ett exempel på hur sannolikhetsfördelningarna för hudfärg och ickehudfärg kan se ut längs en linje i C_bC_r-planet. Hudfördelningen är en linjärkombination av två normalfördelningar, medan ickehudfördelningen är en enkel normalfördelning. Exemplet är rent illustrativt, och bygger inte på resultat.*

De sannolikheter som kan bestämmas med de skattade parametrarna från den multivariata distributionen är P(färg|hud) och P(färg|ickehud). Med hjälp av Bayes sats kan kvoten ovan omformuleras enligt

$$\frac{\mathrm{P(hud|färg)}}{\mathrm{P(ickehud|färg)}} = \\ = \frac{\mathrm{P(färg|hud)P(hud)}}{\mathrm{P(färg)}} \frac{\mathrm{P(färg}}{\mathrm{P(ickehud)P(ickehud|färg)}} = \\ = \frac{\mathrm{P(färg|hud)P(hud)}}{\mathrm{P(färg|ickehud)P(ickehud)}},$$

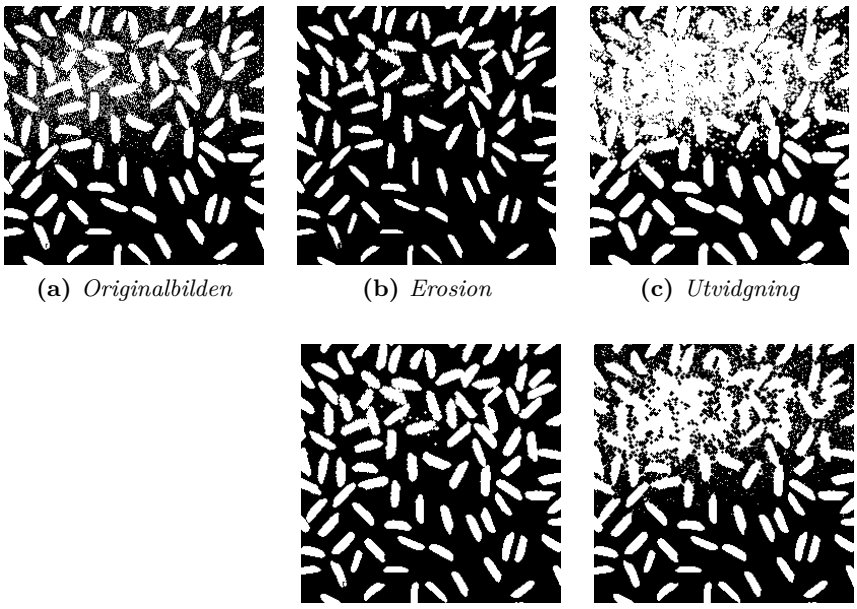
och på så sätt kan man använda tidigare nämnda distributioner för att klassificera hud i bilden.

Kvoten P(hud)/P(ickehud) måste uppskattas och betecknar den förväntade mängden hudbildpunkter i bilden genom den förväntade mängden ickehudbildpunkter. Denna kvot kan bakas in i konstanten c, och det slutliga kravet för att klassa en bildpunkt som hud blir

$$\frac{\mathrm{P(färg|hud)}}{\mathrm{P(färg|ickehud)}} > c,$$

där c anpassas separat. På detta sätt kan en bild från kameran göras binär, där varje bildpunkt är antingen hud eller ickehud (se figur 4.1 på sidan 34 för ett exempel).

Denna metod för att klassa bildpunkter leder dock till att en del bildpunkter klassas fel, vilket resulterar i svarta öar i handen och vita öar i bakgrunden. För att bli av med dessa används morfologiska operationer.



(a) *Originalbilden* **(b)** *Erosion* **(c)** *Utvidgning*

(d) *Öppning* **(e)** *Stängning*

Figur 2.3: *De fyra morfologiska operationerna*

2.2.4 Morfologiska operationer

Morfologiska operationer kan användas för att modifiera områden i binära bilder, till exempel för att rensa bort brus eller för att mjuka upp kanter. Det finns fyra grundläggande morfologiska operationer: *erosion*, *utvidgning*, *öppning* och *stängning* (Rudemo 2009, s. 25), vars resultat visas i figur 2.3.

Dessa definieras som operationer på mängder, specifikt mängden A av alla vita bildpunkter i bilden som behandlas, d.v.s. alla bildpunkter som är ”avstängda”, och mängden S som representerar ett strukturelement som kan liknas vid en pensel med vilken man sveper över bilden. Strukturelementet kan dessutom förflyttas så att dess referensbildpunkt (en punkt kring vilken elementet är definierat, ofta mittpunkten) ligger på plats (i,j) i bilden. Detta förflyttade strukturelement benämns $S_{i,j}$. Innebörden av S framgår av hur det används nedan. Notera dessutom att Rudemo definierar mängden A som alla svarta bildpunkter i bilden. I denna rapport är det mer fördelaktigt att definiera A som mängden vita bildpunkter, då detta är den definition MATLAB använder.

Erosion eroderar bort delar av de vita områdena genom att endast behålla de bildpunkter vars ”omgivning” (definierad av mängden $S_{i,j}$) ligger helt i mängden A. Vad operationen egentligen gör är alltså att skala av det yttersta lagret, randen, av mängden A. Erosion definieras enligt

$$A \ominus S = \{(i,j) \ : \ S_{i,j} \subseteq A\}.$$

Dess motsats, *utvidgning*, agerar som en erosion på komplementet till A och definieras enligt

$$A \oplus S = (A^C \ominus S)^C.$$

Dessa operationer är i sig inte särskilt lämpliga att använda i den tillämpning som beskrivs här, eftersom de förändrar arean av objektet i bilderna. Operationerna kan dock kombineras för att skapa operationer som ”mjukar upp” objekten i bilden utan att förändra deras area markant. Eftersom erosion och utvidgning är motsatser kan dessa appliceras i följd för att behandla bilden och sedan delvis återställa den. Detta ger upphov till två nya operationer: *öppning* och *stängning*.

En öppning är en erosion följd av en utvidgning, enligt

$$\phi_S(A) = (A \ominus S) \oplus S'.$$

Detta resulterar i att vitt brus, d.v.s. vita områden som inte kan täckas av S, försvinner i första steget varefter större vita områdena utökas till sin ursprungliga storlek i det andra steget. Operationen ”öppnar upp” svarta områden.

En stängning består av den omvända proceduren, enligt

$$\Phi_S(A) = (A \oplus S) \ominus S',$$

och ger även motsatt resultat (vita områden ”öppnas upp”).

De båda operationerna använder utöver A och S även S', vilket är S roterad 180° runt sin referensbildpunkt.

2.3 Kvantitativa mått på handens geometri

För att kunna kvantifiera den data en bild av en hand innehåller och tolka den på ett bra sätt, måste denna data transformeras till något meningsfullt. Det finns många sätt att göra detta — man kan exempelvis beräkna bildens egenvärden (Funck 2002), använda filter för att hitta fingertoppar (Nölker och Ritter 1997), eller jämföra bilden med lagrade prototypbilder och sedan studera det s.k. Hausdorff-avståndet till dessa (Sánchez-Nielsen et al. 2004). En enklare metod är att beräkna ett antal olika egenskaper baserade på den binära bilden och använda dessa för klassificering.

Dessa egenskaper kan bestå av mått med enkla definitioner baserade på area, omkrets, mittpunkt eller rotation hos objektet i den binära bilden och kan därför användas av en klassificerare för att skilja mellan olika rörelseoberoende gester.

Eftersom egenskaperna inte innehåller information om rörelse kan denna uppsättning egenskaper inte användas direkt i tillämpningar som involverar rörelsegester, utan måste kompletteras med exempelvis hastighets- eller riktningsegenskaper.

2.3.1 Egenskapsrummet

Egenskapsrummet är ett $\mathbb{R}^n$-rum där varje dimension representerar en egenskap. I detta rum kan man beräkna ”avstånd” mellan olika gester, eller mellan en observerad gest och tidigare träningsdata. Egenskaperna baseras som sagt på olika grundläggande mått hos den binära bilden, och ett antal användbara egenskaper presenteras nedan. Många av egenskaperna som beskrivs kan beräknas effektivt med hjälp av MATLAB-funktionen `regionprops`.

Inledande definitioner. För beräkning av vissa egenskaper behöver man definiera två grundläggande begrepp, den inneslutande lådan och tyngdpunkten. Dessa två kan användas för att hitta ”mitten” av objektet vi undersöker. Den inneslutande lådan kan dessutom användas för vissa jämförelser relaterade till area (d.v.s. densitetsliknande begrepp).

Inneslutande låda. Den inneslutande lådan är den minsta axelparallella rektangeln som innehåller hela objektet (se figur 4.2 på sidan 35), och kan effektivt beräknas genom att lokalisera extrempunkterna i x- och y-led. Lådan har fyra mått: bredd, höjd samt position i x- och y-led. Utifrån dessa kan vi även beräkna mittpunkten i x- och y-led.

Tyngdpunkt. Tyngdpunkten är masscentrum för ett område, och beräknas enligt

$$\text{tyngdpunkt}(A) = \frac{\sum\limits_{a\in A} m_a \mathbf{r}_a}{\sum\limits_{a\in A} m_a} = \sum_{a\in A} \frac{\mathbf{r}_a}{\text{Area}(A)},$$

där man använder att $m_a = 1, \forall a \in A$ eftersom bilden är binär (m_a är helt enkelt intensiteten i punkten a och $\mathbf{r}_a$ är positionen av punkten a). Tyngdpunkten är vektorvärd. Man kan dessutom definiera ett relativt tyngdpunktsläge som avståndet mellan tyngdpunkten och lådans övre/vänstra kant genom lådans bredd/höjd.

Hu-moment. Dessa moment definierades först av Hu (1962) och är ett försök att göra centralmomenten u_{pq} rotations-, skalnings- och translationsinvarianta och därmed mer användbara.

Centralmomenten μ_{pq} definieras av

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x,y)$$

där $\bar{x}$ och $\bar{y}$ är tyngdpunktens x- och y-komponent, och $f(x,y)$ är bildens ljusstyrka. För binära bilder gäller att f antar värdena 0 och 1.

De sju Hu-momenten definieras utifrån centralmomenten (Hu 1962, s. 185):

$$\begin{aligned}
I_1 &= \mu_{20} + \mu_{02} \\
I_2 &= (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2 \\
I_3 &= (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2 \\
I_4 &= (\mu_{30} + \mu 12)^2 + (\mu_{21} + \mu_{03})^2 \\
I_5 &= (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12}) \left((\mu_{30} + \mu 12)^2 - 3(\mu 21 + \mu 03)^2 \right) + \\
&\quad + (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03}) \left(3(\mu_{30} + \mu 12)^2 - (\mu_{21} + \mu_{03})^2 \right) \\
I_6 &= (\mu_{20} - \mu_{02}) \left((\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2 \right) + \\
&\quad + 4\mu_{11} (\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03}) \\
I_7 &= (3\mu_{21} - \mu_{03})(\mu_{30} - \mu_{12}) \left((\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2 \right) - \\
&\quad - (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03}) \left(3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2 \right)
\end{aligned}$$

Excentricitet. Excentriciteten är ett mått på hur utsträckt ett objekt är (precis som fyrkantigheten), men utgår från den ellips vars andramoment μ_{02} och μ_{20} enligt ovan är samma som objektets (se figur 2.4 på nästa sida). Excentriciteten definieras av kvoten mellan avståndet mellan fokalpunkterna i ellipsen och längden av den längsta axeln. Därmed beror måttet, till skillnad från fyrkantigheten, inte på objektets orientering.

Fyrkantighet. Den här egenskapen är ett mått på hur kvadratisk den axelparallella inneslutande lådan är, och på så sätt mäter den hur ”utsträckt” en gest är. Dess definition är mycket enkel:

$$\text{fyrkantighet}(A) = \frac{\text{Box}_w(A)}{\text{Box}_h(A)}.$$

En perfekt kvadratisk låda har alltså fyrkantighet 1, och ju mer fyrkantigheten avviker från detta värde desto mer avlång är gesten. Notera att excentriciteten

(se nedan) i princip mäter samma sak, men utgår från andramomenten och är därmed oberoende av objektets vinkel.

Utsträckning. Utsträckningen mäter hur stor del av den inneslutande lådan som faktiskt täcks av vårt objekt. Egenskapen kan ses som ett alternativ till konvexiteten och definieras enligt

$$\text{utsträckning}(A) = \frac{\text{Area}(A)}{\text{Area}(\text{Box}(A))}.$$

Kompakthet. Kompakthet (Rudemo 2009, s. 26) är ännu ett dimensionslöst mått, som beskriver hur kompakt ett objekt är. Cirkeln är det mest kompakta objektet, och ju fler irregulariteter randen innehåller (jämfört med en cirkel) desto mindre blir kompaktheten. Kompakthet definieras av

$$\text{kompakthet}(A) = \frac{\text{Area}(A)}{|\partial A|^2}.$$

Konvexitet. Konvexitet (Rudemo 2009, s. 26) är ett dimensionslöst mått som beskriver hur konvext ett objekt är — för gestigenkänning är detta ett relevant mått på huruvida fingrarna är utsträckta eller inte, men även på om de sitter tätt ihop (ex. stopptecken) eller löst konfigurerade (ex. segertecken). Det definieras av

$$\text{konvexitet}(A) = \frac{|\partial A_h|}{|\partial A|}$$

där A_h är det konvexa höljet till A (d.v.s. det minsta konvexa område som helt täcker A) och ∂A betecknar randen till A (i diskret mening). Detta innebär i princip att $|\partial A|$ betecknar omkretsen av A.

Figur 2.4: *En ellips vars andramoment μ_{02} och μ_{20} är samma som objektets. Excentriciteten är kvoten mellan avståndet mellan fokalpunkterna (de två prickarna på diagonalen) och den längsta axeln.*

Soliditet. Soliditet är ett mått på hur poröst ett objekt är. Det definieras av kvoten mellan objektets area och det konvexa skalets area enligt

$$\text{soliditet}(A) = \frac{\text{Area}(A)}{\text{Area}(A_h)},$$

och är därför dimensionslöst. Detta mått kan vara mycket hjälpsamt för gester som t.ex. ”OK” (en ring formad av tumme och pekfinger, med övriga fingrar riktade uppåt).

2.3.2 Normering och viktning för klassificering

Eftersom många klassificeringsmetoder använder det euklidiska avståndet mellan punkter i egenskapsrummet är det viktigt att egenskaperna som används varierar över ungefär samma värdemängd. Så är inte är fallet från början och därför måste indata normeras.

Man kan exempelvis skala all indata så att den anpassas till en $\mathcal{N}(0,1)$-distribution (efter att ha sett till att egenskaperna kan tänkas vara normalfördelade, t.ex. genom att logaritmera de kvotbaserade egenskaperna) genom att beräkna uppskattningar $\hat{\mu}$ och $\hat{\sigma}$ utifrån prototyp- eller träningsdata (man får självfallet en uppsättning sådana parametrar per egenskap i egenskapsrummet). Denna metod kommer inte att göra icke normalfördelade egenskaper normalfördelade, men alla egenskapers varians kommer att likställas (Aksoy och Haralick 2001).

Man kan därefter använda dessa parametrar för att normalisera ny indata, så att även den passar in i det nya egenskapsrummet:

$$\hat{x} = \frac{x - \hat{\mu}_x}{\hat{\sigma}_x}$$

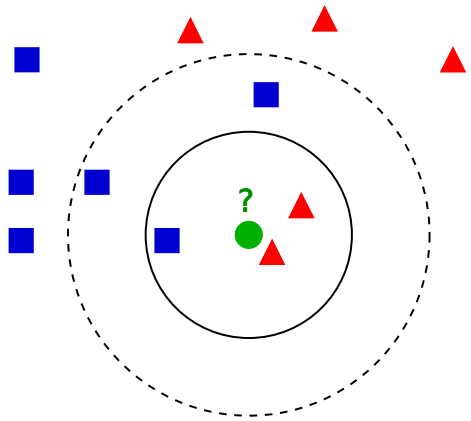
Vill man därefter optimera klassificeraren och se till att den tar större hänsyn till vissa gester kan man även införa en viktningskonstant w_x — en stor sådan gör att egenskapen har större betydelse (eftersom avståndet då ökar snabbare). Denna konstant appliceras lämpligen direkt efter normeringen.

2.4 k nearest neighbours (kNN)

kNN eller k närmsta grannar är en konceptuellt enkel och modellfri klassificeringsmetod. Den baseras på prototypobjekt i egenskapsrummet och en observerad punkt tilldelas en klass baserat på dess närmsta grannar.

Avståndet som används är vanligtvis det euklidiska, d.v.s.

$$d(i) = \left(\sum_{i=1}^{n} (x_i - x_0,i)^2 \right)^{1/2}$$



Figur 2.5: *kNN-klassificering av den runda observationen i mitten i ett rum med två klasser, trianglar och kvadrater. Majoritetsomröstning med $k = 3$ leder till att den runda klassas som triangel, medan $k = 5$ leder till klassificering som kvadrat.*

där n är dimensionen på egenskapsrummet, $\mathbf{x}$ är koordinaterna för en prototyp och $\mathbf{x_0}$ är observationen. Genom en sökning i egenskapsrummet letar vi efter de k närmsta grannarna. Om k är större än ett avgörs klasstillhörigheten genom majoritetsomröstning, d.v.s. den klass med flest nära grannar vinner, se figur 2.5.

Prototypobjekten är redan klassificerade objekt som väljs ur en inlärningsmängd. De kan skrivas som

$$(\mathbf{x}_i, l_i) \qquad l_i \in \{1, \ldots, K\}$$

där l_i är en klassetikett och K är antalet klasser. Prototyperna kan väljas genom mer eller mindre sofistikerade metoder. Exempelvis kan varje redan klassificerad punkt undersökas med kNN mot alla andra punkter, varpå den tas bort ur mängden om den nya klassen avviker från den ursprungliga (Mazurowski et al. 2011).

Klasserna motsvarar i sammanhanget olika statiska gester, eller olika grupperingar i egenskapsrummet. För varje observerad bildruta undersöker man alltså vilken klass den tillhör.

För att metoden ska fungera krävs att alla klasser är representerade med lika många prototypobjekt, annars får vissa klasser större vikt och kommer med högre sannolikhet att väljas. (Klasserna kan fortfarande viktas i efterhand.) Om prototyperna är väl valda avspeglar de den sanna fördelningen hos klassen. Den är därför ofta att föredra framför en Gaussian Mixture Model, där varje klass approximeras av sin anpassade parametriska form (Hastie et al. 2009).

2.4.1 Algoritmer för kNN

Att linjärt söka genom prototypmängden tar lång tid. Istället kan man, beroende på mängdens storlek och antalet egenskaper, använda smartare metoder. Ett tillvägagångssätt är att inte söka de sanna närmsta grannarna, utan att med en slumpmässig metod approximera till önskad noggranhet. Detta tillåter stora datamängder och hög dimension.

En metod som är snabb för dimensioner lägre än ungefär 20 är kd-tree. Den går ut på att prototypmängden delas av plan genom egenskapsrummet, vilket resulterar i ett binärt träd. Färdiga implementeringar finns dessutom i överflöd (Skiena 2008).

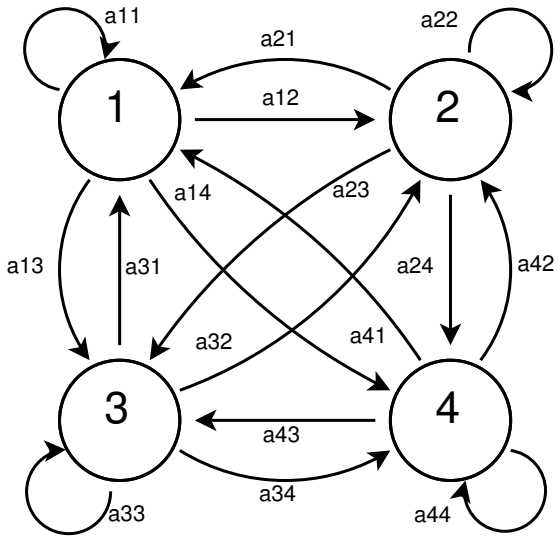
2.5 Dolda Markovmodeller (HMM)

För att kunna klassificera dynamiska gester räcker det inte att analysera bilder var för sig. Man behöver en modell som kan beskriva karakteristiska beteenden för sekvenser av bilder.

En dold Markovmodell eller Hidden Markov Model (HMM) är en utökning av den klassiska Markovmodellen som gjorts för att skapa en signalmodell vilken beskriver ett system utifrån dess utdata. En vanlig Markovmodell består av ett antal tillstånd sammankopplade med övergångssannolikheter (se figur 2.6 på nästa sida). Man kan alltså röra sig i Markovprocessen genom att i varje tidssteg följa en av kanterna ut från tillståndet, och valet av kant sker genom en stokastisk process.

Det som är dolt i en HMM och som särställer den från en vanlig Markovmodell är att varje tillstånd genererar observerbar utdata, symboler, efter ännu en sannolikhetsfördelning. Då man i vanliga Markovmodeller observerar modellens tillstånd direkt, observeras nu istället de symboler som genereras när tillstånden växlar. Dessa symboler motsvarar den verkliga process man önskar modellera. Ett förtydligande exempel är på sin plats:

> En person har ett antal tärningar — vissa är viktade och avviker därmed från den annars uniforma sannolikhetsfördelningen. Han kastar en tärning ur sin samling och berättar för en kompis hur många ögon tärningen visar. Därefter avgör tärningskastaren om han vill byta tärning eller inte, och upprepar proceduren. Kompisen får alltså inte se vilken av tärningarna som kastas för tillfället, utan får bara ta del av vad respektive tärning visar. Denna process skulle kunna beskrivas med hjälp av en HMM där varje enskild tärning representerar ett tillstånd, och antalet ögon som visas är en symbol. Den förstnämnda personen rör sig mellan tärningarna (tillstånden) efter eget behag och tillkännager enbart värdet (symbolerna) som genereras av dem.



Figur 2.6: *En enkel, ergodisk Markovmodell med fyra tillstånd.*

För att tillämpa detta på gestigenkänning måste man konstruera en HMM för varje gest som finns med i systemet. Klassificeringen görs sedan genom att välja den HMM som bäst beskriver den observerade datan, och symbolerna tilldelas genom någon typ av enklare klassificering (t.ex. kNN) av bilddata som resulterar i ett mindre antal "typbilder". Tillstånden motsvarar då olika skeden av gesterna.

2.5.1 Beståndsdelarna av en HMM

En HMM karakteriseras av följande beståndsdelar:

- N, antalet tillstånd i modellen. Vi betecknar de olika tillstånden som $S = \{S_1, S_2, \ldots, S_N\}$, och tillståndet vid tiden t som q_t.
- M, antalet distinkta observationssymboler för ett givet tillstånd. De individuella symbolerna betecknar vi som $V = \{v_1, v_2, \ldots, v_M\}$.
- A, en övergångsmatris där varje element ger sannolikheten att vandra från ett tillstånd till ett annat,

  $$a_{ij} = \mathrm{P}(q_{t+1} = S_j | q_t = S_i), \quad 1 \leq i,j \leq N.$$

  I fallet då man kan nå alla tillstånd från vilket tillstånd som helst med ett enda steg har vi alltså $a_{ij} > 0, \forall i,j$.
- $b_j(k)$, sannolikhetsfördelningar för observationssymboler givet ett specifikt tillstånd j, där

  $$b_j(k) = \mathrm{P}(v_k \text{ vid tiden } t | q_t = S_j) \quad 1 \leq j \leq N, \quad 1 \leq k \leq M.$$

$b_j(k)$ ger med andra ord sannolikheten att symbolen v_k genereras då systemet befinner sig i tillstånd j.

- Sannolikhetsfördelningen för ett begynnelsetillstånd, $\pi = \{\pi_i\}$, där

$$\pi_i = \mathrm{P}(q_1 = S_i).$$

Givet parametrarna M och N, en specificering av symboler, samt de tre sannolikhetsmåtten A, B och π är det möjligt att fullständigt specificera en HMM. I fortsättningen kommer vi att använda oss av den mer kompakta notationen $\lambda = (A,B,\pi)$ för att representera parametrarna för en given HMM.

2.5.2 Grundläggande problem

För att kunna föra över modellen från teori till praktisk tillämpning återstår flera viktiga detaljer. Rabiner (1989) presenterar tre grundläggande frågor för implementeringen av en HMM, varav följande två är av intresse i den här rapporten:

Problem 1: Givet en sekvens av observationer $\mathbf{O} = \{O_1,O_2,\ldots,O_t\}$ och en modell $\lambda = (A,B,\pi)$, hur beräknar man $\mathrm{P}(\mathbf{O}|\lambda)$? D.v.s. vad är sannolikheten för att sekvensen genererades av modellen λ? Hur kan det göras så effektivt som möjligt?

Problem 2: Hur går man tillväga för att optimera $\mathrm{P}(\mathbf{O}|\lambda)$ genom att justera modellparametrarna $\lambda = (A,B,\pi)$?

Kan man lösa det första problemet har man möjlighet att uppskatta hur väl en modell matchar en viss observationsföljd. Man kan se det som ett sätt att jämföra olika modeller och deras relation till en specifik observationsföljd. Denna synvinkel är mycket användbar då man ställs inför utmaningen att välja mellan flera existerande modeller.

Det andra problemet behandlar modellestimering. Hur kan en modell på bästa sätt anpassas till en viss observationsföljd? Lösningen på detta problem utgör en fundamental del i att skapa modellen i praktiken. Med hjälp av en träningssekvens, alltså en följd av observationer, kan man då träna en HMM. Vikten av att optimeringsproceduren ger bra resultat är stor eftersom träningssekvensen utgör den enda kopplingen till det verkliga fenomen man önskar modellera.

2.5.3 Framåt-bakåt-proceduren

En användbar teknik för att lösa det första problemet introducerades av Baum och Egon (1967), kallad *forward-backward*-proceduren. Vi definierar framåtvariabeln $\alpha_t(i)$ enligt

$$\alpha_t(i) = \mathrm{P}(O_1,O_2,\ldots,O_t,q_t = S_i|\lambda),$$

alltså sannolikheten för att ha genererat observationsföljden $\mathbf{O} = \{O_1, O_2, \ldots, O_t\}$ samt befinna sig i tillstånd S_i vid tiden t, givet en modell λ. En av metodens fördelar är att man med hjälp av induktion kan finna $\alpha_t(i)$ genom följande process:

1. Initiering:
$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N$$

2. Induktion:
$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right) b_j(O_{t+1}) \quad 1 \leq t \leq T-1, \quad 1 \leq j \leq N$$

3. Avslutning:
$$\mathrm{P}(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

I första steget, då $t = 1$, initieras framåtvariabeln $\alpha_1(i)$ som sannolikheten att börja i tillstånd S_i och därefter generera observation O_1. För $t = 2$ använder man sedan induktionssteget. Utgår man från första steget kan man övertyga sig om att $\sum_{i=1}^{N} \alpha_1(i) a_{ij}$ måste vara den totala sannolikheten för att hamna i S_j vid $t = 2$, efter att ha genererat O_1 i föregående tillstånd.

Efter multiplikation med $B_j(O_2)$ fås därefter sannolikheten att generera O_2, då systemet befinner sig i S_j med en observerad symbol O_1, eller med andra ord, $\mathrm{P}(O_1, O_2, S_j = q_2|\lambda) = \alpha_2(j)$. Fortsätter man på samma vis har man slutligen $\alpha_T(i) = \mathrm{P}(O_1, O_2, \ldots, O_T, S_i = q_T|\lambda)$.

Sista steget ger slutligen svaret på det första problemet, nämligen sannolikheten $\mathrm{P}(\mathbf{O}|\lambda)$. Detta inses lätt då vi använder oss av definitionen av $\alpha_T(i)$,

$$\mathrm{P}(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} \mathrm{P}(O_1, O_2, \ldots, O_T, q_T = S_i|\lambda).$$

Med andra ord är sannolikheten för att generera observationsföljden $\mathbf{O}$ summan av sannolikheterna för att generera $\mathbf{O}$ med sluttillstånd S_i, $1 \leq i \leq N$.

2.5.4 En iterativ metod för träning av HMM

Med en lösning på första problemet i bagaget kan man tackla nästa, själva träningen av en HMM. Detta är ett betydligt mer invecklat problem än det tidigare, med betydligt fler angreppsvinklar. En analytisk lösning på optimeringsproblemet är inte känd, utan man tvingas istället använda sig av Baum-Welch-metoden eller gradient-tekniker (Dempster et al. 1977, Levinson et al. 1983).

Givet en ändlig observationssekvens går det inte att optimera modellens parametrar globalt. Däremot kan man välja $\lambda = (A,B,\pi)$ på så sätt att $\mathrm{P}(\mathbf{O}|\lambda)$ är lokalt maximerad.

Baum-Welch-metoden utgår från bakåtvariabeln $\beta_t(i)$ som likt framåtvariabeln $\alpha_t(i)$ definieras enligt

$$\beta_t(i) = \mathrm{P}(O_{t+1},O_{t+2},\ldots,O_T|q_t = S_i,\lambda).$$

$\beta_t(i)$ är alltså sannolikheten för att observera en delsekvens av observationer då man startar i tillstånd S_i. Processen för att beräkna $\beta_t(i)$ består av två steg:

1. Initiering:
$$\beta_T(i) = 1, \quad 1 \leq i \leq N.$$

2. Induktion:
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}b_j(O_{t+1})\beta_{t+1}(j), \quad t = T-1,T-2,\ldots,1 \quad 1 \leq i \leq N.$$

Inledningsvis sätts $\beta_T(i)$ godtyckligt till 1, varefter induktionssteget fungerar efter samma princip som för framåtvariabeln α.

Framåt- och bakåtvariablerna kan användas tillsammans för att bilda två nya variabler ξ_t och γ_t. Den första definieras som

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j|\mathbf{O},\lambda),$$

alltså sannolikheten för att befinna sig i tillstånd S_i och därefter i S_j. Uttryckt i α och β kan den skrivas

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\mathrm{P}(\mathbf{O}|\lambda)}$$

där $\mathrm{P}(\mathbf{O}|\lambda)$ normerar ξ_t så att den blir ett korrekt sannolikhetsmått. Den andra variabeln, som beskriver hur många gånger tillståndet S_i besöks vid tiden t, kan uttryckas

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j).$$

Summerar vi sedan $\gamma_t(i)$ över tiden t får vi en storhet som kan tolkas som antalet gånger tillståndet S_i har besökts, eller antalet förflyttningar från tillståndet S_i:

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{antalet förväntade förflyttningar från tillstånd } S_i.$$

På samma sätt kan summation över tiden hos $\xi_t(i,j)$ förklaras. Notera att eftersom $\xi_t(i,j)$ inte är definierad för $t = T$ går summationen endast upp till $T - 1$:

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{antalet förväntade förflyttningar från tillstånd } S_i \text{ till } S_j.$$

Den iterativa metoden som återuppskattar modellens parametrar A,B och π kan nu skrivas som

$$\bar{\pi} = [\text{förväntat antal gånger i tillstånd } S_i \text{ vid } t = 1] = \gamma_i(i),$$

$$\bar{a}_{ij} = \\ = \frac{\text{förväntat antal förflyttningar från tillstånd } S_i \text{ till } S_j}{\text{förväntat antal förflyttningar från tillstånd } S_i} = \\ = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)},$$

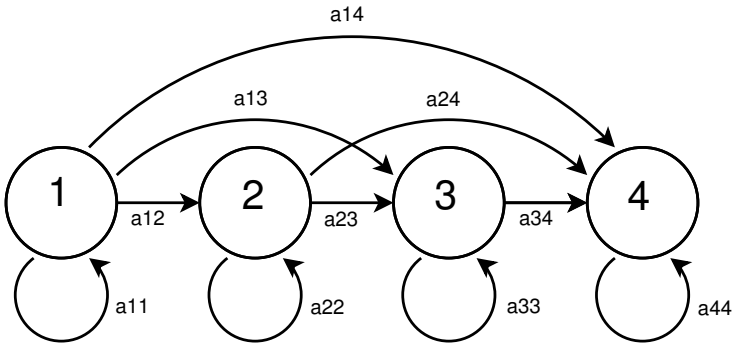
$$\bar{b}_j(k) = \\ = \frac{\text{förväntat antal gånger i tillstånd } S_j \text{ och observerandes symbol } v_k}{\text{förväntat antal gånger i tillstånd } S_j} = \\ = \frac{\sum_{\substack{t=1 \\ s.t\ O_t = v_k}}^{T-1} \gamma_t j}{\sum_{t=1}^{T-1} \gamma_t(j)}.$$

Vänsterleden fås genom att ta den nuvarande modellen $\lambda = (A,B,\pi)$, sedan beräkna högerleden, och därigenom erhålla $\bar{\lambda} = (\bar{A},\bar{B},\bar{\pi})$. Baum och Sell (1968) och Baker (1975) visade att denna iterativa process resulterar antingen i *1*) ett lokalt maximum för $\mathrm{P}(\mathbf{O}|\lambda)$ (alltså att $\lambda = \bar{\lambda}$), eller *2*) i en förbättrad modell $\bar{\lambda}$ på så sätt att $\mathrm{P}(\mathbf{O}|\bar{\lambda}) > \mathrm{P}(\mathbf{O}|\lambda)$.

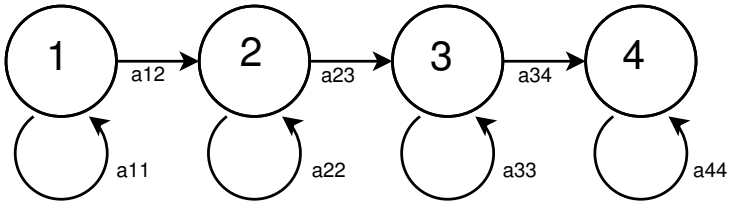
2.5.5 Olika typer av HMM

Då man talar om olika typer av HMM syftar man först och främst på den typ av övergångsmatris som används. Hittills har bara fallet där $a_{ij} > 0$ för alla i,j nämnts — detta kallas en ergodisk modell. Detta innebär att varje tillstånd i modellen kan nås från alla andra tillstånd inom ändlig tid. Om man tänker sig en modell med fyra tillstånd blir övergångsmatrisen följaktligen en allmän 4×4-matris A där alla element kan vara nollskilda.

Det har dock visats att andra typer av modeller i vissa fall presterar bättre än den ergodiska. Elmezain et al. (2008) presenterar resultat som pekar ut



(a) *Markovmodell enligt Bakis-modellen*



(b) *Markovmodell enligt (Yamato et al. 1992)*

Figur 2.7: *Två Markovmodeller med fyra tillstånd enligt olika modeller.*

Bakis-modellen, även kallad vänster-höger-modellen, som den främsta då det gäller modellering av tidsvarierande signaler. Bakis-modellen fungerar på så sätt att den inte tillåter förflyttning till ett ”tidigare” tillstånd. Elementen i övergångsmatrisen A tvingas med andra ord efterfölja restriktionen

$$a_{ij} = 0 \quad \text{om } j < i,$$

och matrisen blir uppåt triangulär. Vidare får sannolikhetsfördelningen för begynnelsetillstånd följande utseende

$$\pi_i = \begin{cases} 1 & \text{om } i = 1 \\ 0 & \text{om } i \neq 1. \end{cases}.$$

Man startar alltså alltid i det första tillståndet för att sedan stanna kvar i samma tillstånd eller röra sig åt höger (se figur 2.7a). Antalet steg man tar är inte begränsat på annat sätt än att det slutgiltiga tillståndet S_N inte får passeras.

Denna modell användes av Yamato et al. (1992) och skiljer sig i vissa avseenden från den som den som används av Elmezain et al. (2008), t.ex. på det sätt att den bara tillåter förflyttning från ett tillstånd till sig självt eller till nästa (man

får alltså inte ”hoppa över” tillstånd, se figur 2.7b på föregående sida), ett val som ger upphov till övergångsmatrisen

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}.$$

Utöver dessa varianter finns det självklart otaliga andra då man kan sammanbinda de olika tillstånden helt efter eget tycke.

Kapitel 3

Metod

Detta kapitel går igenom tillämpningen av den teori som förklarats i kapitel 2, och förklarar hur vi gjort för att skapa ett program som kan identifiera handgester. Då vårt angreppssätt lett till en hel del intermediära resultat är kapitlet hårt knutet till kapitel 4, och ofta har ett delresultat påverkat våra beslut och vår metod.

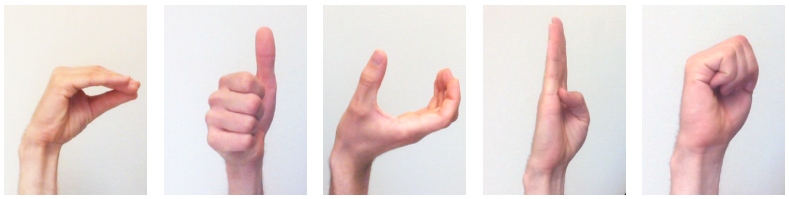
Vi presenterar även praktiska aspekter av vår implementering, och förklarar i detalj några av de algoritmer och programmatiska metoder vi använt oss av. Vi kommer även att förklara en del av den kod vi använt, som även inkluderats i bilaga B.

3.1 Våra gester

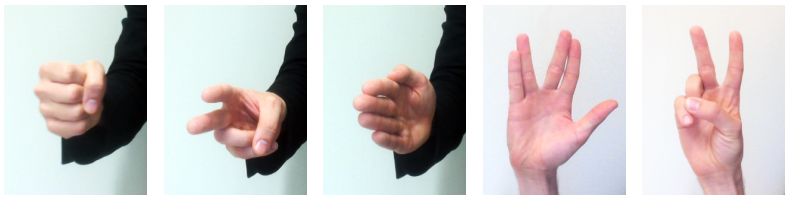
Handgester kan delas in i två kategorier, statiska och dynamiska. De statiska gesterna är sådana där handen inte rör sig, utan en enda bildruta är tillräcklig för att bestämma vad det är för gest. Exempel på sådana är freds- och segergesten med två fingrar i ett V och ”tumme upp”-gesten. Dynamiska gester å andra sidan är sådana som innefattar en rörelse, exempelvis en vinkning. Dessa är betydligt svårare att behandla ur ett igenkänningsperspektiv, då de kräver en följd av bildrutor och bilderna därför inte kan behandlas separat.

Vid testerna av programmet har vi använt tio statiska gester (figur 3.1 på följande sida):

- Svenska handalfabetets tecken för A. 4 fingrar hålls ihop och tummen möter pekfingret. Hålet mellan tumme och pekfinger är ovalt.
- Svenska handalfabetets tecken för B. Sträckt tumme uppåt, i övrigt knutna fingrar, ”tumme upp”.



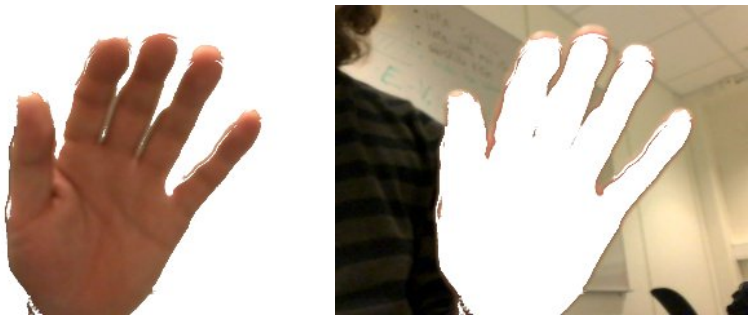
(a) *A* **(b)** *B* **(c)** *C* **(d)** *D* **(e)** *E*



(f) *Sten* **(g)** *Sax* **(h)** *Påse* **(i)** *Spock* **(j)** *Seger*

Figur 3.1: *Vi har tio olika statiska handgester att matcha mot indata.*

- Svenska handalfabetets tecken för C. Kupad hand med tummen upp från handflatan. Handflatan uppåt och lillfingret närmast bröstet.
- Svenska handalfabetets tecken för D. Handen sträckt uppåt med pekfingret mot bröstet. Tummen böjd in i handen.
- Svenska handalfabetets tecken för E. Knuten hand runt tummen. Pekfingret närmast kroppen.
- Stenen i den traditionella leken sten-sax-påse. Knuten hand, tummen utanför pekfingret. Tummen uppåt.
- Saxen i den traditionella leken sten-sax-påse. Pek- och långfinger sträckta framåt, separerade, övriga fingrar knutna.
- Påsen i den traditionella leken sten-sax-påse. Handen sträckt, fingrarna pekande framåt.
- Spock-hälsning. Handflatan mot kameran, fingrarna sträckta med pek- och långfinger ihop, ring- och lillfinger ihop och mellanrum mellan lång- och ringfinger. Även kallad Vulcan Salute, från TV-serien Star Trek, inspirerad av en judisk hälsning.
- Segertecken. Handflatan mot kameran, pek- och långfinger sträckta i ett V, övriga fingrar knutna. Kan även ses som en fredssymbol, flitigt använd av allierade och motståndsrörelser i de ockuperade länderna under andra världskriget.



(a) *Hud* **(b)** *Ickehud*

Figur 3.2: *För att skapa sannolikhetsfördelningar för hudfärg respektive färg för ickehud togs 44 bilder ur vilka huden separerades från övriga bilden.*

3.2 Hud- och handidentifiering

3.2.1 Klassificering av hud

I 44 bilder tagna med webbkameran separerades huden från resten av bilden manuellt med hjälp av fotoredigeringsprogramvara (se figur 3.2. Baserat på dessa bildpunkter bestämdes sedan en sannolikhetsfördelning för både hud och ickehud enligt kapitel 2.2).

Med hjälp av sannolikhetsfördelningarna delades kamerabilden in i hud resp. ickehud, så att resultatet kunde bedömas. För att inte behöva räkna ut sannolikheten för färgen i varje bildpunkt och bildruta som analyseras, vilket hade krävt lång beräkningstid, gjordes beräkningen i förväg för varje tänkbar färg (256^2 stycken i C_bC_r-rymden). Resultaten sparades sedan i två 256×256-matriser, där färgvärdet användes som koordinater. Ett antal binära matriser skapades sedan med ettor på de positioner där klassificeringskravet

$$\frac{\mathrm{P(färg|hud)}}{\mathrm{P(färg|ickehud)}} > c,$$

var uppfyllt för olika värden på konstanten c. Ju större c var desto färre färger klassades som hud.

I dessa matriser, eller ”hudfärgskartor”, kunde man sedan slå upp färgvärden för att direkt få veta om dessa skulle klassas som hud. De olika kartorna testades, och den som gav en så bra bild av handen som möjligt utan att ta med stora bakgrundsområden användes. Metoden med hudfärgskartor har visats vara effektiv av bl.a. Brand och Mason (2000).

De resulterande binära bilderna jämfördes sedan med dem som erhölls med sannolikhetsfördelningarna framtagna av Elmezain et al. (2008). Detta för att bestämma kvaliteten på våra parametrar.

3.2.2 Urskiljning av handen

När väl hudbildpunkterna i figuren urskilts återstår att identifiera handen som ska följas. Det är troligt att man i bilden fått med ett antal mindre områden som felaktigt bedömts vara hud. Dessutom är huvudet ofta med i bilden, vilket kan ställa till med problem.

Vår lösning på problemet är att först använda den inbyggda MATLAB-funktionen `bwareaopen`, som är en morfologisk öppningsfunktion. Metoden som beskrivs i 2.2.4 använder ett strukturelement S, som `bwareaopen` genererar utifrån en parameter som specificerar dess area — vad som är en lämplig area varierar med upplösning och och avstånd till kameran. Vi använder en gräns på 200 bildpunkter, vilket fungerar bra även för relativt stort avstånd till kameran med en upplösning på 320×240 bildpunkter.

För att hitta handen gör vi antagandet att handen är det objekt som befinner sig längst till vänster i bilden. Vi sveper därför över bildmatrisens kolonner från vänster för att se om kolonnen innehåller någon hudbildpunkt — den första hudbildpunkten som nås på detta sätt antas tillhöra handen. Det sammanhängande område som bildpunkten är en del av antas därför vara den sökta handen.

Att på detta sätt välja objektet längst till vänster har tidigare gjorts av bl.a. Sánchez-Nielsen et al. (2004). De använde sig dock av det högra av de två största sammanhängande objekten, vilket har fördelen att man inte behöver sätta en manuell gräns för när ett objekt är ”tillräckligt stort”. Till nackdelarna hör att man tvingas söka igenom hela bilden, att den är något svårare att implementera samt att den endast fungerar då både huvud och hand finns i bilden. Dessutom fungerar den endast för högerhänta användare, men kan relativt enkelt göras tillgänglig även för vänsterhänta. Om endast statiska gester ska behandlas räcker det med att spegelvända bilden, medan man för dynamiska gester (eftersom rörelseriktningen spelar roll) måste söka från höger istället, och även skapa en separat träningsmängd.

Metoden inkluderar all hud som hänger samman med handen, vilket innefattar armen när personen framför kameran bär en kortärmad tröja. För att kringgå detta kan man identifiera handleden och ”kapa” handen där. Detta gjordes av Deimel och Schröter (1999) genom att identifiera den största cirkel som får plats helt i handen, och som antags vara centrerad mitt i handen. Då cirkeln förstoras något är det längsta cirkelsegment som helt får plats inom hudområdet placerat vid handleden, och tangenten till segmentet kommer då skära handleden vinkelrätt — handen utgörs sedan av hudområdet på den sida om handleden där cirkeln befinner sig. Denna metod har vi dock inte testat, varför användaren förväntas bära långärmad tröja.

3.3 Datainsamling och analys

3.3.1 Inspelning av gester

Eftersom syftet med arbetet var att behandla rörliga gester var det naturligt att använda sig av inspelade filmsekvenser av varje gest. För att bearbetningen av dessa filmsekvenser inte skulle ta för lång tid, och för att samma metod senare skulle kunna tillämpas i realtid, nöjde vi oss med en upplösning på 320×240 bildpunkter.

Kameran var en Logitech C250, med automatisk kompensering av vitbalans, vilker medförde att färger i kläder och omgivning kraftigt påverkade bilden, och därmed förvrängde hudfärgerna. Detta gjorde i sin tur att vi var tvungna att ha ett större område på hudfärgskartan som motsvarade hud, varpå fler bildområden felaktigt klassificerades som hud. För att råda bot på detta valde vi att begränsa uppställningen till en miljö fri från hudliknande färger och utan varierande ljusförhållanden.

Vi lät åtta personer spela in varje gest fem gånger. För att inom varje gest erhålla större spridning av handens position och form visade försökspersonen de olika gesterna efter varandra, och upprepade sedan hela sekvensen tills alla filmats fem gånger. Sammanlagt fick vi då 400 filmklipp att analysera. Alla filmer verifierades för hand och uppenbart felaktiga ersattes, varefter de tilldelades ett gestspecifikt index samt ett tal mellan 1 och 40 för spårning till en viss videofil.

3.3.2 Egenskapsanalys av filmerna

Egenskaperna från kapitel 2.3 samlades i en funktion (bilaga B) där MATLAB:s inbyggda funktioner kombinerades med egenskrivna. Dessa egenskaper beräknades för handen i alla bildrutor från varje film, och resultatet sparades i matriser. Detta eftersom analysen av filmerna tog lång tid — mycket längre än filmernas speltid.

Egenskaperna normerades därefter enligt kapitel 2.3.2, och de nya egenskapsvektorerna användes sedan för klassificering.

3.4 Klassificering av gester

För klassificering av gesterna användes kNN. För att metoden skulle ge ett bra resultat var vi tvungna att välja en bra prototypmängd, och dessutom välja vilka egenskaper vi skulle inkludera och vilket värde på k som var optimalt.

3.4.1 Konstruktion av prototypmängd för kNN

Inlärningsmängden skapades genom att plocka ut tre femtedelar av filmerna för respektive gest — de övriga två femtedelarna användes som testmängd, ur vilken vi endast plockade ut de fyra första bildrutorna i varje klipp, sammanlagt 96 bilder per gest. Att ha en oberoende testmängd är viktigt för att kunna avgöra om klassificeraren är effektiv, vi testade därför även med personer som inte redan fanns i vår inlärningsmängd.

Eftersom resultatet vid kNN-klassificering är kraftigt beroende av datamängden, skapade vi en alternativ prototypmängd för realtidstester. Här lät vi testpersonen filma handen ur flera vinklar, för att erhålla en mer varierad representation av varje gest.

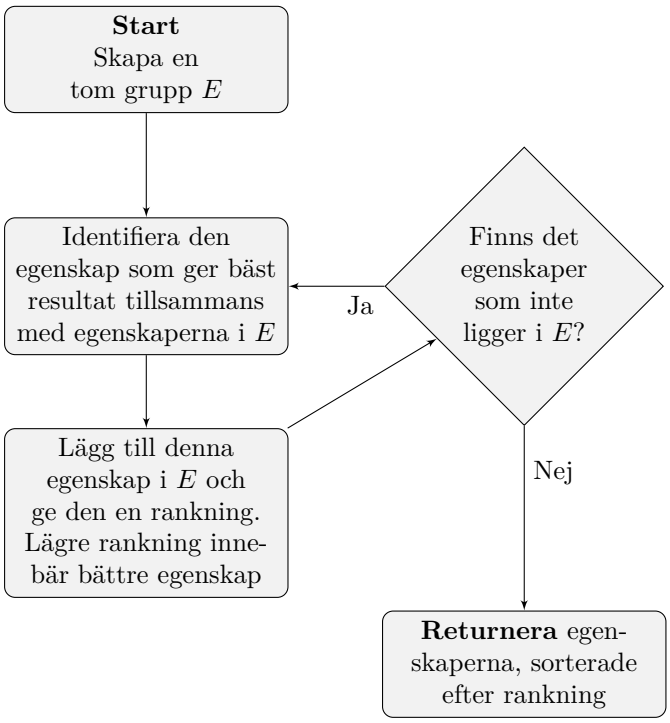
3.4.2 Optimering av antal egenskaper och grannar i kNN

Fram till denna punkt hade vi endast definierat ett antal egenskaper utan att diskutera huruvida de var användbara som klassificerare eller inte. Att nå klarhet i denna fråga, och att därmed ha möjlighet att reducera antalet egenskaper, är fördelaktigt av flera anledningar. Dels kan man märkbart förbättra klassificerarens tidseffektivitet då färre egenskaper behöver beräknas och analyseras, dels kan man eliminera de egenskaper som kanske rentav försämrar klassificeringen. Det kan handla om egenskaper som är överdrivet utspridda i egenskapsrummet eller instabila egenskaper som förorenar datan med slumpmässiga variationer.

Då vi först tog oss an problemet att finna de bäst klassificerande egenskaperna identifierade vi tre olika tillvägagångssätt. En variant är att testa alla möjliga kombinationer av egenskaper, men detta ger upphov till enorma beräkningskrav. Med 15 egenskaper ger detta sammanlagt 32 767 kombinationer, och varje kombination måste testas mot hela testmängden, i vårt fall 640 bilder, vilket hade tagit alltför lång tid.

Den andra varianten utnyttjar möjligheten att vikta de olika egenskaperna för att på så sätt ge de som bättre separerar klasserna en större inverkan. Man optimerar då viktfördelningen för att nå ett så bra resultat som möjligt. Vi experimenterade med denna metod lite kort, men konstaterade snart att även den var för beräkningsintensiv för våra tillämpningar.

Det tredje tillvägagångssättet, det som vi slutligen använde, är s.k. girig optimering. Denna kan utföras med två snarlika algoritmer, framåturval (forward selection) respektive bakåteliminering (backward elimination). Framåturvalet förklaras i figur 3.3 på motstående sida. Bakåteliminering är snarlik, men inkluderar istället samtliga egenskaper från början, och plockar successivt bort den som bidrar minst. Resultatet av dessa algoritmer är en rangordning av egenskaperna. Vi beräknade denna rangordning för olika värden av k, alltså parametern i kNN-metoden.



Figur 3.3: *Girigt framåturval för att välja de bästa egenskaperna*

Vi lät k variera mellan 1 och 13 och fick på så sätt ut andelen korrekta klassificeringar som funktion av k och antal inkluderade egenskaper för de två algoritmerna. Bilaga B visar vår implementering av dessa algoritmer i MATLAB.

Jain och Zongker (1997) utförde en undersökning av algoritmer för urval av egenskaper, och fann att framåturval gav bättre resultat än andra mer beräkningsintensiva metoder såsom branch-and-bound. Här bör nämnas att eftersom algoritmen utför en icke-uttömmande sökning så är det inte garanterat att den hittar ett globalt optimum (Cover och Van Campenhout 1977).

3.5 Implementering av HMM för klassificering av icke-statiska gester

För att klassificera dynamiska gester bygger man vidare på den metod som användes för de statiska. HMM kräver att gester utgörs av följder av symboler, motsvarande olika observationer. Dessa symboler får vi från en klassificering med kNN, med en prototypmängd enligt nedan. Idén är att representera varje gest, statisk som dynamisk, i form av en symbolföljd. De statiska definieras lättast

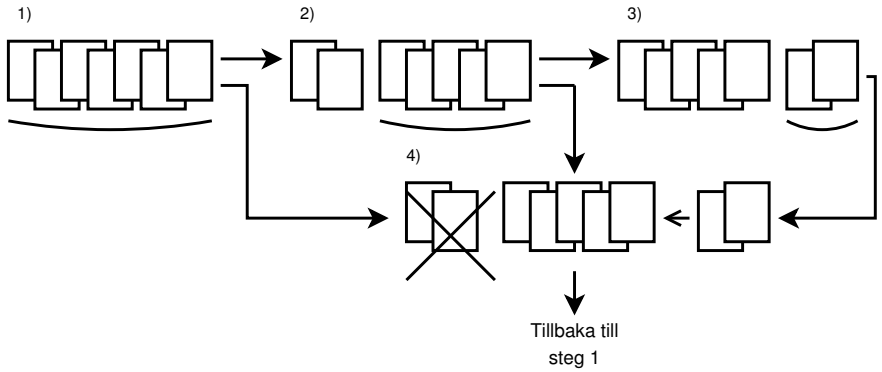
som två eller tre efterföljande symboler av samma typ, medan de dynamiska består av en följd av blandade symboler.

Eftersom de statiska gesterna redan tilldelats symboler är svårigheten att tilldela dynamiska gester symboler på ett effektivt sätt. Vår tanke är att dela upp filmerna med dynamiska gester i tre kategorier: prototypfilmer, träningsfilmer och testfilmer. Tilldelningen av symboler sker genom att varje prototypfilm delas upp i fyra lika långa delar. Den första fjärdedelen tilldelas en symbol, den andra en annan o.s.v. Förhoppningen är att det som ligger i en viss fjärdedel alltid motsvarar samma del av gesten, eftersom de tilldelas samma symbol. De symboler som förknippats med dessa delgester, används med tillhörande egenskapsvektorer för att definiera nya klasser i prototypmängden.

Genom att använda kNN i prototypmängden omvandlar programmet träningsfilmerna till symbolföljder. Längden av symbolföljden för respektive gest beror på programmets prestanda, d.v.s. hur många bilder man önskar analysera per sekund. Det bör uppmärksammas att varianter av en och samma gest, beroende på dess längd, kommer att representeras av olika långa symbolföljder, vilket leder till att vi tränar flera HMM-modeller för varje gest. En möjlighet är att gruppera symbolsekvenserna i tre längder för att motverka alltför strikta krav på gestens utförande.

HMM-modellerna tränas med Baum-Welch-metoden via MATLAB-funktionen `hmmtrain`. Vi använder Bakis-modellen då den enligt kapitel 2.5.5 ger bra resultat i samband med signaler som förändras över tid. Det är en typ av HMM som medför att modellerna måste tränas med flera symbolsekvenser (Rabiner 1989). Då ingen förflyttning till ett tidigare tillstånd är tillåten blir modellen mycket selektiv när det gäller vilka symboler varje tillstånd kan generera — att endast använda en symbolföljd skulle träna modellen till att systematiskt vandra från ett tillstånd till nästa, och i varje tillstånd generera rätt symbol i ordningen med sannolikhet ett.

För att beräkna sannolikheten för att en viss HMM har skapat en symbolföljd används MATLAB-funktionen `hmmviterbi`. Om vi antar att det har tränats HMM-modeller för gester av längd två, fem och sju symboler, kommer programmets klassificering av testfilmerna att ske på det sätt som beskrivs i figur 3.4 på nästa sida.



Figur 3.4: *Klassificering av dynamiska gester med två, fem och sju symboler sker enligt en relativt enkel procedur:*

1. *Programmet beräknar sannolikheten att en HMM tränad med sju symboler skapat symbolföljden. Är någon sannolikhet tillräckligt hög registreras den aktuella gesten och programmet går till steg fyra, annars går det vidare till steg två.*
2. *Programmet beräknar sannolikheten att de fem senaste symbolerna skapats av en HMM tränad med fem symboler. Vid tillräckligt hög sannolikhet registreras en gest och programmet går till steg fyra, annars går det vidare till steg tre.*
3. *Programmet beräknar sannolikheten att de två senaste symbolerna skapats av en HMM tränad med två symboler. Om sannolikheten är tillräckligt hög registreras en (statisk) gest.*
4. *De två äldsta symbolerna kasseras och två nya sätts in i slutet av följden. Programmet börjar om från steg ett.*

Märk att denna algoritm prioriterar långa symbolföljder, d.v.s. störst går först.

Kapitel 4

Resultat

En stor del av resultatet består av den metod som använts och som beskrivs i kapitel 3, samt det ramverk av kod som producerats och som återfinns i bilaga B. Detta kapitel kommer därför endast att presentera några delresultat inom hudidentifiering och val av egenskaper samt beräknade värden för träffsäkerhet hos gestigenkänningen.

4.1 Klassificering av hudfärg och urskiljning av händer

Både vår sannolikhetsfördelning för icke-hudfärg och motsvarande fördelning från Elmezain et al. (2008) har väldigt liten varians, d.v.s. diagonalelementen i kovariansmatrisen är små. För att ge en bra hudidentifiering bör denna varians vara stor, annars klassas allt ”bortom” hudfärg (rödaktiga färger) som hud. Detta kan inses då figur 2.2 på sidan 7 studeras. Alltför stora delar av högra axelhalvan skulle där klassas som hudfärg. Vi valde därför att istället sätta variansen oändlig, så att sannolikhetsfördelningen för bildpunkter som inte föreställde hud var likformig.

Den sannolikhetsfördelning vi erhåller genom egen bildanalys skiljer ut händer relativt bra, men den lämnar ett antal ”hål” i handen. Även den fördelning som ges av Elmezain et al. (2008) fungerar bra, och ger färre felaktiga ickehudklassificeringar. Denna fördelning används därför i resten av arbetet.

Metoden för identifiering av händer visar sig fungera mycket väl i kontrollerad miljö. De förutsättningar som gäller, är — förutom långärmad tröja — att belysningen är god och helst kommer från lysrör samt att bilden inte får innehålla för stora rödfärgade partier, detta eftersom vårt hudfärgsområde är något för stort åt det röda hållet.

En visualisering av hudurskiljningen kan ses i figur 4.1 på nästa sida, där alla

Figur 4.1: *Figuren visar de områden i bilden som klassats som hud. Observera det vita bruset i bildens kanter, vilket filtreras bort med morfologiska operationer.*

områden som klassas som hud är vita och övriga är svarta. Handen är markerad med en inneslutande låda i figur 4.2 på motstående sida.

4.2 Klassificering av gester med hjälp av kNN

Den största andelen korrekt klassificerade gester vi lyckas uppnå med samtliga gester tillåtna är 91.9 %, både med framåturval och bakåteliminering. Antalet aktiverade egenskaper blir 10 då algoritmen för framåturval användes och 7 med bakåteliminering, och antalet närmsta grannar i kNN-metoden är 7. Egenskaperna som används är listade i tabell A.1 på sidan A-1.

Figur 4.4 på sidan 36 visar mer exakt hur andelen korrekta klassficeringar beror på antalet aktiverade egenskaper och på värdet av k. Det framgår tydligt att antalet aktiverade egenskaper har stor inverkan på andelen felklassificerade gester om man använder sig av färre än 5 egenskaper. Därefter planar andelen ut för att återigen stiga då fler än 10 egenskaper används. Andelen felklassificeringar minskar också i takt med att k ökar från 1 till 7, för att sedan öka då värdet ökar från 7 till 13. Med valet av egenskaper och k tagna från optimeringen med framåturval presterade metoden enligt tabell 4.4 på sidan 38 och tabell 4.1 på sidan 37.

Figur 4.2: *Handen har ringats in av programmet, som det ”tillräckligt” stora objekt som befinner sig längst till vänster i bilden.*

4.2.1 Val av egenskaper

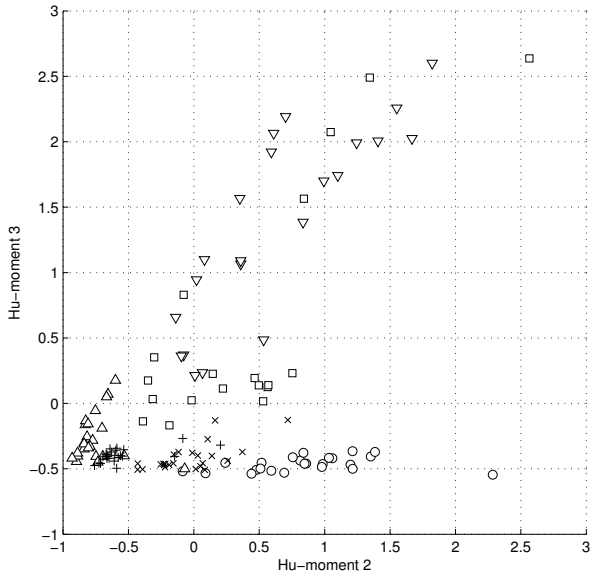
Egenskaperna som använts har visat sig vara mycket bra — upp till en viss gräns. I tabell A.1 på sidan A-1 syns de tio egenskaper som inkluderades med framåturval. Av dessa kan man ta bort tre — excentricitet, Hu-moment 2 och Hu-moment 7 — med oförändrad klassificeringsförmåga, enligt resultat från bakåteliminering. De sju bästa egenskaperna som resulterar från bakåteliminering listas i tabell 4.3 på sidan 37, i ordning efter hur länge de är kvar i körningen.

Rankningen av samtliga egenskaper från respektive metod finns i appendix A.

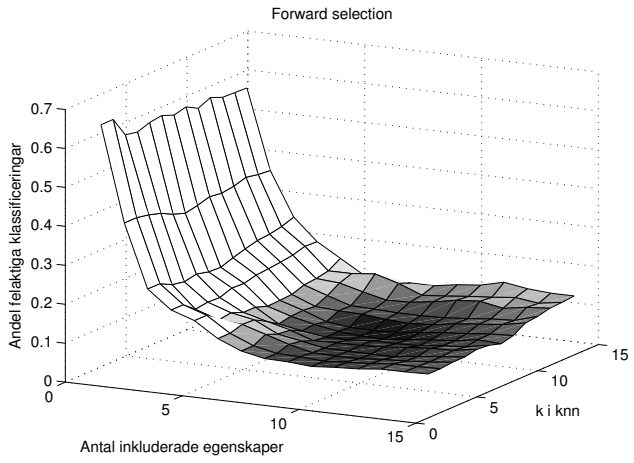
Ett urval av sex gester visar i figur 4.3 på nästa sida hur de två bästa egenskaperna tydligt samlar gesterna i relativt små områden, vilket gör att avstånden i kNN-metoden blir mindre. Det ska dock noteras att egenskaperna överlappar varandra för vissa gester vilket sannolikt minskar träffsäkerheten.

4.2.2 Klassificering i realtid

Med lämpliga val av egenskaper och en stor, representativ prototypmängd, kan klassificeraren i realtid klassificera testpersoner, med upprepat tillfredsställande resultat.



Figur 4.3: *De olika symbolerna motsvarar olika gester, plottade mot de två starkaste egenskaperna enligt framåturvalet. Notera hur de flesta är tydligt samlade, men att vissa gester överlappar varandra. För två av gesterna (nedåtvända trianglar och fyrkanter) är de två egenskaperna tydligt korrelerade, medan de inte är det för andra.*



Figur 4.4: *Optimering av kNN-metoden med avseende på inkluderade egenskaper och värden på k med algoritmen för framåturval. Ett minimum i andel felaktiga klassificeringar, 91.4 %, fås då $k = 7$ och 10 egenskaper inkluderades. Med bakåteliminering fås samma minsta andel felaktiga klassificeringar, fast med 7 inkluderade egenskaper och $k = 7$.*

Tabell 4.1: *Andel felklassificeringar av statiska gester i procent. Apriorifel betecknar sådana fel då gesten varit känd men identifierats som något annat; aposteriorifel betecknar sådana fel då identifikationen är känd men felaktig.*

Försöksgest	Apriorifel	Aposteriorifel
A	14	15
B	11	12
C	0	0
D	0	0
E	30	41
Sten	6	5
Sax	16	18
Påse	9	9
Spock	0	0
Seger	0	0

Tabell 4.2: *De tio bästa egenskaperna med framåturval*

Rankning	Egenskap
1	Hu-moment 3
2	Hu-moment 2
3	Tyngdpunktsläge, y-led
4	Fyrkantighet
5	Soliditet
6	Excentricitet
7	Konvexitet
8	Hu-moment 1
9	Utsträckning
10	Hu-moment 7

Tabell 4.3: *De sju bästa egenskaperna med bakåteliminering*

Rankning	Egenskap
1	Fyrkantighet
2	Tyngdpunktsläge, y-led
3	Hu-moment 1
4	Soliditet
5	Konvexitet
6	Utsträckning
7	Hu-moment 3

Tabell 4.4: *En s.k. confusion matrix för klassificering av tio statiska gester. Tecknet för E var det klart svåraste att klassificera, med endast 70 % korrekta klassificeringar. Varje gest testades med 64 bilder.*

Verklig \ Tolkad	A	B	C	D	E	Sten	Sax	Påse	Spock	Seger
A	**55**		**3**		**1**			**4**		**1**
B		**57**								**7**
C			**64**							
D				**64**						
E	**6**		**2**		**45**	**9**	**1**	**1**		
Sten						**60**			**4**	
Sax						**3**	**54**	**2**	**5**	
Påse	**1**					**4**	**1**	**58**		
Spock									**64**	
Seger										**64**

Kapitel 5

Diskussion

De metoder som använts ger en träffsäkerhet på 91.4 % för de tio statiska gesterna. Denna träffsäkerhet kan sannolikt förbättras på flera olika sätt, dessutom kan metoden utökas och generaliseras för att kunna identifiera dynamiska gester. I detta kapitel presenteras förslag på möjliga framtida förbättringar och utökningar av metoden samt exempel på tillämpningar.

5.1 Hudigenkänning

De sannolikhetsfördelningar som används för att skilja hud från annat fungerar tillräckligt bra för att kunna skilja ut handen i en relativt kontrollerad miljö, men bättre resultat skulle kunna uppnås med bättre sannolikhetsfördelningar. En sak man skulle kunna göra är att använda Gaussian Mixture Models, d.v.s. linjärkombinationer av flera sannolikhetsfördelningar, inte bara för hud utan även för ickehud. Detta bidrar till att problematiska färger, som ofta klassas fel, kan filtreras bort. Bättre fördelningar skulle antagligen även fås med en större datamängd, alltså fler exempelbilder.

Man kan också använda någon annan klassificerare, såsom kNN, vid skapandet av hudfärgskartan. Ett problem med den metoden är att den kräver att man kasserar data för att få lika stora prototypmängder, eller att man viktar datapunkterna. Efter att det statistiska urvalet gjorts skulle man kunna redigera direkt i hudfärgskartan för att få ett bättre resultat.

En robustare metod är eventuellt den adaptiva metod som föreslås av Hassanpour et al. (2008), där klassificerade sammanhängade hudområden återkopplas till beräkningen av fördelningarna. På så vis anpassas hudfärgsfördelningarna kontinuerligt till miljön. Slutligen kan en mer omfattande filtrering och bildbehandling användas för att fylla i de ”hål” som bildas i handen (se figur 4.1 på sidan 34).

5.2 Handidentifiering

Handidentifieringen kan förbättras på ett antal punkter; den metod vi använder, att söka ett tillräckligt stort objekt från höger i bilden, är relativt ”dum”. Förutsätter man att de två objekt som kan finnas i bilden är ett ansikte och en hand (vilket ofta är fallet) så kan man istället välja handen genom att först identifiera ansiktet, exempelvis genom den metod Hsu et al. (2002) använder, för att sedan ”välja bort” detta objekt (d.v.s. ta bort det från bilden). Vad som då finns kvar i bilden bör vara handen (om brus först eliminerats eller om de två största objekten i bilden redan isolerats enligt Sánchez-Nielsen et al. (2004)).

Man kan dessutom klippa bort handleden på det sätt Deimel och Schröter (1999) föreslår, och därmed öka träffsäkerheten — att handleden ibland syns och ibland skuggas är något som gör vissa av egenskaperna alltför osäkra. Dessutom skulle man då kunna släppa på kravet att användaren bär en långärmad tröja.

Då handen identifierats kan man till sist utnyttja vetskapen om dess läge, och klippa bort resten av bilden i nästa bildruta. På så vis minskar risken för att man tappar bort handen och klassar ett annat objekt som handen.

5.3 Förbättrade egenskaper

De egenskaper som använts för klassificering av gesterna har visat sig vara relativt effektiva, men det finns modifieringar och tillägg som skulle kunna öka träffsäkerheten.

En sådan modifiering är att istället för fyrkantigheten $\Delta x/\Delta y$ använda dess logaritm, $\ln(\Delta x/\Delta y)$, eftersom kvotens värdemängd är $(0,\infty)$ och tar värdet 1 när höjden och bredden är desamma, vilket innebär att egenskapen får större viktning när handens utsträckning är horisontell än när den är vertikal. Detta problem försvinner när kvoten logaritmeras. Även övriga egenskaper som definieras genom kvoter kan logaritmeras på samma sätt.

Det finns också hela klasser av egenskaper som inte använts. Genom studier av handens kurvatur kan fingertoppar identifieras — antal, avstånd och relativt läge för dessa är mycket intressanta egenskaper, och ger antagligen ett bra resultat. En annan intressant egenskapsklass är den som utnyttjar mönster i det inre av handen; tidigare analyserades enbart egenskaper för den binära bilden. Detta skulle kunna göras genom att man, efter att ha identifierat handen, studerar den i originalbilden, t.ex. genom ett kantdetektionsfilter. I den resulterande ”kantbilden” kan sedan riktningar och antal kanter beräknas med hjälp av en så kallad Houghtransform (Duda och Hart 1972).

5.4 Prototypval och kNN

För att förbättra kNN-metodens träffsäkerhet kan först och främst prototypmängden utökas till fler punkter per gest. En annan möjlighet är att inspektera de bilder som faktiskt ligger bakom punkten i egenskapsrummet. Om en bild är uppenbart felaktig, och kanske hade varit svår att tolka rätt även för en människa, kan den med fördel tas bort. Slarvigt utförda gester kommer då mindre sannolikt att klassificeras rätt.

Man kan även införa ett tröskelvärde på avståndet till de närmaste prototypobjekten, över vilket gesten inte klassificeras som någon gest. På detta sätt kan man löpande söka efter gester i en filmsekvens, ett viktigt steg mot realtidsidentifiering.

Att klassificeringsresultaten från framåturval och bakåteliminering är identiska, trots skillnaden i antal egenskaper, gör att andra optimeringsrutiner kanske är värda att undersöka. Ett mer sofistikerat prototypurval och ”cross validation” är exempel på möjliga förbättringar.

Det största problemet med den metod som beskrivs i rapporten är dock inte implementeringen utan testandet av den. Eftersom vi tagit fyra efterföljande bilder från varje film i testmängden är datan inte oberoende. Tvärtom är det mycket troligt att samtliga fyra bilder ska klassas likadant, varför den effektiva testmängden inte är mycket mer än 16 bilder per gest. Denna mängd är egentligen för liten för att kunna dra några säkra slutsatser om klassificerarens prestanda.

5.5 Realtid

Målet för en gestigenkänningsapplikation är givetvis att kunna tolka gester i realtid, vilket kan vara svårt eftersom bildbehandlingen och klassificeringen ofta är beräkningsintensiv. Vår implementering är skriven i MATLAB, och skulle bli mycket snabbare om den istället skrevs i ett lågnivåspråk som C, men kan trots det utföra beräkningarna snabbt nog för att nästan fungera i realtid. Ju fler egenskaper man lägger till och ju mer förfinad analys, desto större blir naturligtvis beräkningskraven. Detta är framförallt viktigt vid tillämpningar i robotar och inbyggda system, där beräkningskraften är begränsad.

För att kunna känna igen statiska gester behövs betydligt färre bilder per sekund än de 24 webbkameran ger. Därmed kan man nöja sig med att behandla fyra eller sex bilder per sekund, vilket gör att gestigenkänningen ser ut att vara i realtid. Detta medför givetvis att statiska gester som visas under mindre tid än intervallet mellan bilderna inte skulle kännas igen.

5.6 Tillämpningar

De gester som visas i 3.1 på sidan 24 leder genast tankarna till att utöka programmet till att bli en värdig motståndare i sten-sax-påse. En tillämpning av mindre lekfull karaktär är att ersätta ”petskärmar” och datormöss med gestbaserade ”pekskärmar” i sjukvården, något som skulle kunna hindra smittspridning. Tillämpningar av en mer utvecklad rörelsebaserad metod är t.ex. fullständig tolkning av teckenspråk, liksom avläsning en människas fulla gestspråk — något som tillsammans med röst- och ansiktsigenkänning klart skulle platsa i humanoidrobotar för hemtjänst.

5.7 Dynamiska gester

För att på allvar uppnå den potential människa-dator-interaktion erbjuder är det viktigt att även kunna klassificera dynamiska gester. Med den HMM-metod vi utformat befinner vi oss på gränsen till att nå resultat och att kunna avgöra hur väl den fungerar i praktiken.

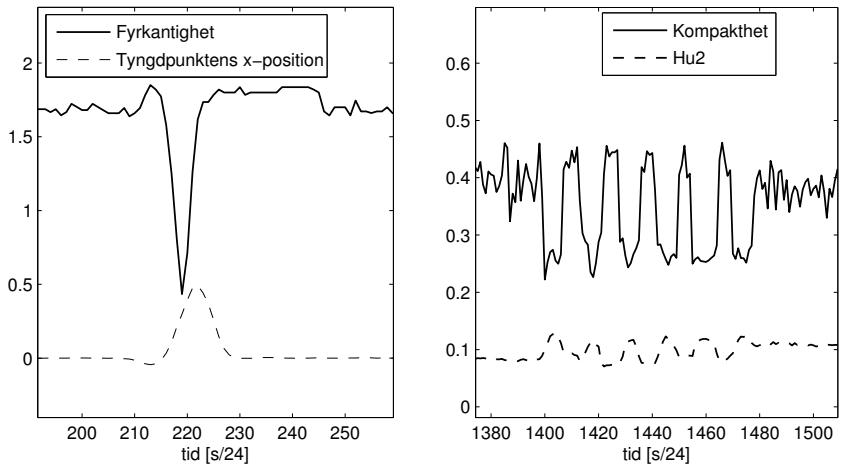
Eftersom metoden behandlar dynamiska gester som flera statiska i följd, återkommer flera av utmaningarna förknippade med dessa. Beräkningsintensiteten tilltar i och med implementeringen av HMM, vilket innebär att MATLAB antagligen blir otillräckligt.

Eftersom dynamiska gester av samma typ kan skilja sig åt i varje bildruta, kommer de totala skillnaderna mellan inspelningarna att vara större än för statiska gester. Detta leder till mer jobb under träningsfasen, då fler filmsekvenser kan behövas för att uppnå en tillräcklig prototypmängd, och för att träna HMM:er till att klara av variationerna. Man måste även introducera nya egenskaper som karakteriserar dynamiska gester, exempelvis position och hastighet.

I ett tidigt skede bearbetade vi även dynamiska gester. I figur 5.1 på motstående sida visas hur egenskaperna då kan se ut. Dessa är exempel på egenskaper som bär på information om gesten, och bör kunna tolkas även av en dator. Det mönster som uppträder är för vissa av egenskaperna ungefär likadant för alla inspelningar av gesten. Dessa tidsberoende egenskaper kan analyseras exempelvis med hjälp av waveletanalys (Hastie et al. 2009).

5.8 Slutsats

Med det ramverk för analys som presenterats, och med koden i bilaga B som utgångspunkt, kan program som klassificerar statiska och dynamiska gester relativt enkelt implementeras och testas. Våra resultat visar att kamerabaserad gestigenkänning med statistiska metoder kan ge tillräckligt goda resultat för många tillämpningar, även om ett antal förbättringar krävs för att nå önskvärd robusthet. Flera vidareutvecklingar för att både vidga funktionaliteten och förbättra prestandan ligger nära till hands.



Figur 5.1: *Till vänster syns en gest som motsvarar en örfil. När handen är riktad mot kameran blir den inneslutande lådan mycket smal, vilket tydligt syns i grafen. Även den (relativa) tyngdpunktsförflyttning som sker är utmärkande för gesten. Till höger syns en gest där handen förs åt sidan och samtidigt upprepat öppnas och stängs.*

Litteraturförteckning

Aksoy, S. och Haralick, R. M. (2001). Feature normalization and likelihood-based similarity measures for image retrieval, *Pattern Recognition Letters* vol. **22**, nr. 5: ss. 563 – 582.

Albiol, A., Torres, L. och Delp, E. J. (2001). Optimum color spaces for skin detection, *IEEE International Conference on Image Processing*, vol. **1**, ss. 122–124.

Baker, J. K. (1975). The Dragon system - an overview, *IEEE Transactions on. Acoustic Speech Signal Processing* vol. **23**, nr. 1: ss. 24–29.

Baum, L. och Egon, J. (1967). An inequality with applications to statistical estimation for probabilistic functions of a markov process and to a model for ecology, *Bull. Amer. Meteorol. Soc.* vol. **73**: ss. 360–363.

Baum, L. och Sell, G. (1968). Growth functions for transformations on manifolds, *Pac. J. Math.* vol. **27**, nr. 2: ss. 211 – 227.

Brand, J. och Mason, J. S. (2000). A comparative assessment of three approaches to pixel-level human skin-detection, *Proceedings — International Conference on Pattern Recognition* vol. **15**, nr. 1: ss. 1056–1059.

Cover, T. M. och Van Campenhout, J. M. (1977). On the possible orderings in the measurement selection problem, *IEEE Transactions on Systems, Man and Cybernetics* vol. **7**, nr. 9: ss. 657–661.

Deimel, B. och Schröter, S. (1999). Improving hand-gesture recognition via video based methods for the separation of the forearm from the human hand, *3rd Gesture Workshop*, Gif-sur-Yvette, France.

Dempster, A. P., Laird, N. M. och Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm, *Journal of the Royal Statistical Society, Series B* vol. **39**, nr. 1: ss. 1–38.

Duda, R. O. och Hart, P. (1972). Use of the hough transformation to detect lines and curves in pictures, *Communications of the ACM* vol. **15**, nr. 1: ss. 11–15.

Elmezain, M., Al-Hamadi, A., Appenrodt, J. och Michaelis, B. (2008). A hidden markov model-based isolated and meaningful hand gesture recognition, *World Academy of Science, Engineering and Technology* vol. **41**: ss. 393–400.

Funck, S. (2002). Video-based handsign recognition for intuitive human-computer-interaction, *i* L. Van Gool (red.), *Pattern Recognition*, vol. **2449** av *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, ss. 26–33.

Garg, P., Aggarwal, N. och Sofat, S. (2009). Vision based hand gesture recognition, *Proceedings of World Academy of Science, Engineering and Technology* vol. **37**: ss. 1024–1029.

Hassanpour, R., Shahbahrami, A. och Wong, S. (2008). Adaptive gaussian mixture model for skin color segmentation, *World Academy of Science, Engineering and Technology* vol. **41**: ss. 1–6.

Hastie, T., Tibshirani, R. och Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*, 2 uppl., Springer, New York.

Hsu, R., Abdel-Mottaleb, M. och Jain, A. K. (2002). Face detection in color images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. **24**, nr. 5: ss. 696–706.

Hu, M.-K. (1962). Visual pattern recognition by moment invariants, *IRE Transactions on Information Theory* vol. **8**, nr. 2: ss. 179 –187.

ITU (2007). Recommendation ITU-R BT.601-6. http://www.itu.int/rec/R-REC-BT.601/

Jain, A. och Zongker, D. (1997). Feature selection: evaluation, application, and small sample performance, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* vol. **19**, nr. 2: ss. 153 –158.

Kakumanu, P., Makrogiannis, S. och Bourbakis, N. (2007). A survey of skin-color modeling and detection methods, *Pattern Recognition* vol. **40**, nr. 3: ss. 1106–1122.

Khan, R., Hanbury, A. och Stoettinger, J. (2010). Skin detection: A random forest approach, *Proceedings — International Conference on Image Processing, ICIP*, ss. 4613–4616.

Kruppa, H., Bauer, M. och Schiele, B. (2002). Skin patch detection in real-world images, *i* L. Van Gool (red.), *Pattern Recognition*, vol. **2449** av *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, ss. 109–116.

Levinson, S., Rabiner, L. och Sondhi, M. (1983). An introduction to the application of theory of probabilistic functions of a markov process to automatic speech recognition., *Bell System Technical Journal* vol. **4**: ss. 1035–1074.

Lockton, R. och Fitzgibbon, A. W. (2002). Real-time gesture recognition using deterministic boosting, *Proceedings, British Machine Vision Conference*.

Mazurowski, M. A., Malof, J. M. och Tourassi, G. D. (2011). Comparative analysis of instance selection algorithms for instance-based classifiers in the context of medical decision support, *Physics in Medicine and Biology* vol. **56**, nr. 2: ss. 473–489.

Nölker, C. och Ritter, H. (1997). Detection of fingertips in human hand movement sequences, *In Wachsmuth and Fröhlich 21*, Springer Verlag, ss. 209–218.

Pavlovic, V. I., Sharma, R. och Huang, T. S. (1997). Visual interpretation of hand gestures for human-computer interaction: A review, *IEEE Transactions on Pattern Analysis and Machine Intelligence* vol. **19**, nr. 7: ss. 677–695.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition, *Proceedings of the IEEE*, ss. 257–286.

Rudemo, M. (2009). Image analysis and spatial statistics. http://www.math.chalmers.se/~rudemo/notes.html

Sánchez-Nielsen, E., Antón-Canalís, L. och Hernández-Tejera, M. (2004). Hand gesture recognition for human-machine interaction, *Journal of WSCG* vol. **12**, nr. 1–3.

Sebe, N., Cohen, I., Huang, T. S. och Gevers, T. (2004). Skin detection: A bayesian network approach, *Proceedings — International Conference on Pattern Recognition*, vol. **2**, ss. 903–906.

Shang, K., Zhou, P. och Li, G. (2010). Study on skin color image segmentation used by fuzzy-c-means arithmetic, *Proceedings — 2010 7th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2010*, vol. **2**, ss. 612–615.

Skiena, S. S. (2008). *The algorithm design manual*, 2 uppl., Springer, New York.

Yamato, J., Ohya, J. och Ishii, K. (1992). Recognizing human action in time-sequential images using hidden markov model, *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, ss. 379–385.

Zabulis, X., Baltzakis, H. och Argyros, A. A. (2009). *The universal access handbook*, CRC Press, Boca Raton, kap. 34 "Vision-based Hand Gesture Recognition for Human Computer Interaction".

Bilaga A

Resultat för egenskaper

Tabell A.1: *Rankning för samtliga egenskaper med framåturval*

Rankning	Egenskap
1	Hu-moment 3
2	Hu-moment 2
3	Tyngdpunktsläge, y-led
4	Fyrkantighet
5	Soliditet
6	Excentricitet
7	Konvexitet
8	Hu-moment 1
9	Utsträckning
10	Hu-moment 7
11	Hu-moment 5
12	Hu-moment 6
13	Hu-moment 4
14	Tyngdpunktsläge, x-led
15	Kompakthet

Tabell A.2: *Rankning för samtliga egenskaper med bakåteliminering*

Rankning	Egenskap
1	Fyrkantighet
2	Tyngdpunktsläge, y-led
3	Hu-moment 1
4	Soliditet
5	Konvexitet
6	Utsträckning
7	Hu-moment 3
8	Hu-moment 7
9	Hu-moment 2
10	Excentricitet
11	Tyngdpunktsläge, x-led
12	Kompakthet
13	Hu-moment 4
14	Hu-moment 6
15	Hu-moment 5

Bilaga B

MATLAB-kod

B.1 Kod för import av videodata

B.1.1 Realtidsvideo

```
function realtid()
%REALTID klassificerar gester i realtid
%
% SYNOPSIS: realtid()
%
% Denna funktion läser in video i realtid från en kamera,
% identifierar handen och beräknar egenskaper för var
% n:te bildruta. Därefter försöker funktionen m.h.a.
% kNN klassificera gesten i bildrutan.
%
% SEE ALSO: videofil

% "öppna" kameran och ställ in den på YCbCr-läge
% Observera att första raden beror på plattform, denna
% kod fungerar med den inbyggda kameran på en MacBook Pro.
vid = videoinput('macvideo', 1, 'YCbCr422_1280x720');
set(vid,'ReturnedColorSpace','YCbCr');
set(vid,'FramesPerTrigger',1);
set(vid,'TriggerRepeat',Inf);
triggerconfig(vid, 'Manual');

% Starta videoinmatningen
start(vid);

% Importera externt lagrad data
SPM = importdata('skinprobmatrix2.mat'); % SPM (se 3.2.2)
normal = importdata('normal.mat'); % Normalisering (se 2.3.2)
```

```

% Aktivera endast en del av egenskaperna
active = [2 3 4 5 6 8 9 10 11 15];
% Välj lämpligt k till kNN
k = 7;

% Iterera över alla bildrutor i indata
for i=1:1000
    % Trigga inmatning av en ny bildruta
    trigger(vid);
    % Läs in och skala om en bildruta
    ycbcr = imresize(getdata(vid,1,'uint8'), 0.25);
    % är det en bildruta vi "hinner" behandla?
    if mod(i,4) == 0
      % Hitta hud och hand i bilden
      hand = gethand(getskin(ycbcr, SPM));
      % Beräkna egenskaper
      feats = cell2mat(struct2cell(features(hand)))';
      % Normalisera egenskaper
      feats = (feats - [normal(:).mu])./[normal(:).stddev];
      % Plocka ut aktiva egenskaper
      feats = feats(active);
      % Använd kNN för att klassificera gesten
      [symb, dist] = getSymbol(feats, k, active);
      % Visa den utplockade handen för användaren
      rgb = im2double(ycbcr2rgb(ycbcr));
      rgb(:,:,1) = rgb(:,:,1).*hand;
      rgb(:,:,2) = rgb(:,:,2).*hand;
      rgb(:,:,3) = rgb(:,:,3).*hand;
      imshow(rgb);
      % Skriv ut vilken gest skriptet tror det var
      gester{symb}
    end
end

% Stäng anslutningen till kameran
stop(vid); delete(vid);
```

B.1.2 Sparade videofiler

```
function geststream = videofil(in_file)
%VIDEOFIL klassificerar gester i en videofil
%
% Denna funktion läser in en videofil, identifierar handen och
% beräknar egenskaper för varje bildruta. Därefter försöker den
% klassificera gester i vajre bildruta m.h.a. kNN.
%
% SYNOPSIS: gester = videofil(filnamn)
%
```

```
10 % INPUT filnamn: Den videofil som ska behandlas
11 %
12 % OUTPUT gester: En vektor med en gestgissning per
13 % bildruta i filmen.
14 %
15 % SEE ALSO: realtid
16
17 % Läs in video från filen
18 vid = VideoReader(in_file);
19 nFrames = vid.NumberOfFrames();
20
21 % Variabel för utdata
22 geststream = [];
23
24 % Importera externt lagrad data
25 SPM = importdata('skinprobmatrix2.mat'); % SPM (se 3.2.2)
26 normal = importdata('normal.mat'); % Normalisering (se 2.3.2)
27
28 % Aktivera endast en del av egenskaperna
29 active = [2 3 4 5 6 8 9 10 11 15];
30 % Välj lämpligt k till kNN
31 k = 7;
32
33 for i=1:nFrames
34     % Läs in en bildruta och konvertera till YCbCr
35     ycbcr = rgb2ycbcr(vid.read(1));
36     % Identifiera hud och hand
37     hand = gethand(getskin(rgb2ycbcr(rgb), SPM));
38     % Beräkna egenskaper
39     feats = cell2mat(struct2cell(features(hand)))';
40     % Normalisera egenskaper
41     feats = (feats - [normal(:).mu])./[normal(:).stddev];
42     % Plocka ut aktiva egenskaper
43     feats = feats(active);
44     % Använd kNN för att klassificera gesten
45     [symb, dist] = getSymbol(feats, k, active);
46     % Lägg till gestgissningen i utdatan
47     geststream = [geststream; symb];
48 end
```

B.2 Kod för hud- och handseparering

B.2.1 getskin.m

```
1 function skin = getskin(img, spm)
2 %GETSKIN klassificerar hud i en bild
3 %
4 % Funktionen använder en fördefinierad SPM enligt
```

```
% kapitel 3.2.1 för att identifiera hud i en YCbCr-bild.
%
% SYNOPSIS: skin = getskin(image, spm)
%
% INPUT image: YCbCr-bild hud skall hittas i
% spm: Skin Probability Map
%
% OUTPUT skin: Binär bild där 1 representerar hud och 0
% representerar ickehud.

% Förallokera utdata
skin = zeros([size(img, 1) size(img, 2) 1]);
% Ineffektiv sökning i SPM för varje bildpunkt i indatan
for j=1:size(img, 1)
    for k=1:size(img, 2)
        skin(j,k) = spm(img(j,k,3), img(j,k,2));
    end
end
```

B.2.2 gethand.m

```
function hand = gethand(img, clip, reverse)
%GETHAND plockar ut objekted längst till vänster (handen)
%
% Plockar ut objektet längst till vänster i bilden (dvs. handen,
% se kapitel 3.2.2), eller längst till höger om så begärs.
% Indatan kan dessutom "klippas" för att spara beräkningskraft.
%
% SYNOPSIS: hand = gethand(image, clip, reverse)
%
% INPUT image: Den binära bild i vilken handen ska hittas
% clip: Vektor av formatet [x y w h] som specificerar
% ett område i vilket funktionen ska söka
% reverse: Funktionen letar från höger om denna variabel är
% nollskild.
%
% OUTPUT hand: En binär bild som endast innehåller ett område,
% som kan anses vara handen i bilden "image"

% Finns argumentet "clip"? Om inte, definiera!
if(nargin < 2 || clip == []) clip = ...
  [1 1 size(img, 2) size(img, 1)]; end
x = clip(1); y = clip(2); w = clip(3); h = clip(4);
% är "reverse" definierat?
if(nargin < 3) reverse = 0; end
% Ta bort små objekt i bilden (se 3.2.2)
img = bwareaopen(img, 200);

% Iterationsvariabler
```

```
29 from = x; to = x+w-1; step = 1;
30 if(reverse ~= 0) from = x+w-1; to = x; step = -1; end
31
32 % Börja med att utgå ifrån hela bilden
33 hand = img;
34 % Stega igenom varje kolumn och se om den innehåller en vit
35 % bildpunkt. Gör den det, så välj det område den tillhör.
36 for j=from:step:to
37     if max(img(y:(y+h-1),j) == 1)
38         hand = bwselect(img, j, ...
39                find(img(y:(y+h-1),j) == 1, 1));
40         break
41     end
42 end
43 % Om vi inte hittar en hand, leta i hela området (ignorera "clip")
44 if(max(max(hand)) == 0) hand = gethand(img); end
```

B.3 Kod för beräkning av egenskaper

B.3.1 features.m

```
1  function f = features(frame)
2  %FEATURES beräknar egenskaper hos en binär bild
3  %
4  % Den här funktionen beräknar femton egenskaper (av dessa nämns
5  % några i kapitel 2.3) utifrån en binär bild "frame" vilken
6  % antas innehålla ett enda sammanhängande objekt föreställande en
7  % hand.
8  %
9  % SYNOPSIS: f = features(frame)
10 %
11 % INPUT frame: binär bild med ett sammanhängande objekt (en hand)
12 %
13 % OUTPUT f: struct innehållandes ett antal egenskaper
14
15 % Importera konstanter för normalisering av egenskaperna
16 norm = importdata('normal.mat');
17
18 % Beräkna ett antal egenskaper m.h.a. funktionen regionprops
19 props = regionprops(this_frame, 'Area','Orientation',...
20         'Perimeter','Centroid','Eccentricity','Extent',...
21         'Solidity','ConvexImage','BoundingBox');
22 props = props(1);
23
24 % Kompakthet
25 f.Compactness = 4*pi*props.Area/(props.Perimeter)^2;
26 % Soliditet
27 f.Solidity = props.Solidity;
```

```
% Konvexitet
convex_props = regionprops(props.ConvexImage,'Perimeter');
f.Convexity = convex_props.Perimeter/props.Perimeter;
% Excentricitet
f.Eccentricity = props.Eccentricity;
% Utsträckning
f.Extent = props.Extent;
% Fyrkantighet
f.Squareness = props.BoundingBox(3)/props.BoundingBox(4);
% Tyngdpunktsläge
f.CentroidBoxPosX = ( props.Centroid(1) - ...
        props.BoundingBox(1) ) / props.BoundingBox(3);
f.CentroidBoxPosY = ( props.Centroid(2) - ...
        props.BoundingBox(2) ) / props.BoundingBox(4);
% Hu-moment (se bilaga B)
phi = moments(this_frame);
f.Hu1 = phi(1);
f.Hu2 = phi(2);
f.Hu3 = phi(3);
f.Hu4 = phi(4);
f.Hu5 = phi(5);
f.Hu6 = phi(6);
f.Hu7 = phi(7);
```

B.3.2 moments.m

```
function phi = moments(Im)
%MOMENTS beräknar Hu-momenten hos en binär bild
%
% Beräknar Hu-momenten (Hu 1962) så som de definieras i
% kapitel 2.3 utifrån den binära bilden "Im".
%
% SYNOPSIS: phi = moments(Im)
%
% INPUT Im: binär bild för vilken Hu-momenten ska beräknas
%
% OUTPUT phi: de sju första Hu-momenten för bilden "Im"

% Bildens bredd och höjd
width = size(Im,2);
height = size(Im,1);
% 4 × 4-matris för att lagra moment M_00 t.o.m. M_33
M = zeros(4);
% Matriser för beräkning av de rena momenten
[K,L] = meshgrid(1:width, 1:height);
% Beräkning av de rena momenten M_00 t.o.m. M_33
for i = 1:size(M,1)
    for j = 1:size(M,1)
        s = sum(sum(K.^(i-1).*L.^(j-1).*Im));
```

```
24          M(i,j) = s;
25      end
26  end
27  % Tyngdpunktens x- och y-koordinat
28  x_bar = M(2,1)/M(1,1);
29  y_bar = M(1,2)/M(1,1);
30  % Beräkning av centralmomenten μ_ij och de skalningsinvarianta
31  % momenten η_ij
32  mu = zeros(4);
33  nu = zeros(size(mu));
34  for p = 1:size(mu,1)
35      for q = 1:size(mu,1)
36          s = 0;
37          for m = 1:size(M,1)
38              for n = 1:size(M,1)
39                  if(p >= m && q >= n)
40                    s = s + nchoosek(p-1,m-1)*...
41                        nchoosek(q-1,n-1)*(-x_bar)^(p-m)*...
42                        (-y_bar)^(q-n)*M(m,n);
43                  end
44              end
45          end
46          mu(p,q) = s;
47          gamma = (p+q)/2;
48          nu(p,q) = mu(p,q)/(mu(1,1)^gamma);
49      end
50  end
51  % Beräkning av de faktiska Hu-momenten
52  phi = zeros(7,1);
53  phi(1) = nu(3,1) + nu(1,3);
54  phi(2) = (nu(3,1) + nu(1,3))^2 + 4*nu(2,2)^2;
55  phi(3) = (nu(4,1) - 3*nu(2,3))^2 + (3*nu(3,1) - nu(1,4))^2;
56  phi(4) = (nu(4,1) + nu(2,3))^2 + (nu(3,2) + nu(1,4))^2;
57  phi(5) = (nu(4,1) - 3*nu(2,3))*(nu(4,1) + nu(2,3))*...
58           (3*(nu(4,1) + nu(2,3))^2 - 3*(nu(3,2) + ...
59           nu(1,4))^2) + (3*nu(3,2) - nu(1,4))*(nu(3,2)...
60           + nu(1,4))*(3*(nu(4,1) + nu(2,3))^2 - ...
61           3*(nu(3,2) + nu(1,4))^2);
62  phi(6) = (nu(3,1) - nu(1,3))*((nu(4,1) + nu(2,3))^2 -...
63           (nu(3,2) + nu(1,4))^2) + 4*nu(2,2)*(nu(4,1) +...
64           nu(2,3))*(nu(3,2) + nu(3,2));
65  phi(7) = (3*nu(3,2) - nu(1,4))*(nu(4,1) + nu(2,3))*...
66           ((nu(4,1) + nu(2,3))^2 - 3*(nu(3,2) + ...
67           nu(1,4))^2)... - (nu(4,1) - 3*nu(2,3))*...
68           (nu(3,2) + nu(1,4))*(3*(nu(4,1) + nu(2,3))^2...
69           - 3*(nu(3,2) + nu(1,4))^2);
```

B.4 Kod för kNN-klassificering

B.4.1 getSymbol.m

```
function [symbol, dist] = getSymbol(featv,k,active)
%GETSYMBOL klassificerar en gest med kNN
%
% Denna funktion klassificerar en gest utifrån ett
% antal egenskaper med hjälp av kNN-metoden. Den
% prototypmängd som används laddas automatisk in.
% Majority voting appliceras.
%
% SYNOPSIS: [gesture, dist] = getSymbol(feats, k, active)
%
% INPUT featv: Egenskaperna gesten klassificeras utifrån
% k: Antalet närmsta grannar
% active: En vektor som berättar vilka egenskaper som
% tas hänsyn till i prototypmängden
%
% OUTPUT gesture: Den klassificerade gesten
% dist: Genomsnittligt grannavstånd

% Ladda in prototypmängd
persistent book;
if isempty(book) book = importdata('protobook.mat'); end

% Leta upp de k närmsta grannarna
[IDX, D] = knnsearch(book(:,active),featv,...
                     'K',k,'NSMethod','kdtree');
neighbours = book(IDX,end-1);

% Sortera stigande efter avstånd
IDXD = [IDX;D]'; IDXD = sortrows(IDXD,2);

% Hitta den vanligaste grannen (majority voting)
[symbol, f, same_freq] = mode(neighbours);

% Kolla om vi har "oavgjort"
while (size(same_freq{1},1) ~= 1)
    % Ta bort den yttersta grannen och försök igen
    IDXD = IDXD(1:(end-1),:);
    neighbours = book(IDXD(:,1),end-1);
    [symbol, f, same_freq] = mode(neighbours);
end

% Beräkna det genomsnittliga avståndet
dist = mean(IDXD(:,2));
```

B.5 Kod för optimering av kNN

B.5.1 Girigt framåturval

```
% Detta skript optimerar vår kNN enligt kapitel 3.4.2.
% Algoritmen som används är framåturval

% Ladda testdata
load testbook; last = size(testbook,1);

% För resultat
best_active = {};
best_ratio = zeros(13,15);

% För varje k vi vill testa
for k = 1:13
    % Skapa tom mängd E
    active = [];
    % Komplementet till E
    nonactive = 1:15;

    for j = 1:15
        % Vi vill hitta lägsta felration.
        % Initiera med ∞ för alla egenskaper!
        ratio = Inf*ones(1,15);
        % Alla egenskaper i komplementet till E ska behandlas
        for t = nonactive
            syms = [];
            % För alla observationer, klassificera gesten
            for i=1:last
                % Hämta det kända gestvärdet
                real_sym = testbook(i,end-1);
                % Hämta egenskapsvektorn
                featv = testbook(i,[active t]);
                % Klassificera gesten
                [obs_sym, dist] = ...
                     getSymbol(featv,k,[active t]);
                % Spara resultatet
                syms=[syms; real_sym obs_sym dist];
            end
            % Beräkna andel fel
            n_fel = size(syms( syms(:,1) ~= syms(:,2) ),1);
            n_okej = size(syms( syms(:,1) == syms(:,2) ),1);
            ratio(t) = n_fel/(n_fel+n_okej);
        end
        % Hitta "bästa" egenskapen och flytta in i E
        best_idx = find(min(ratio)==ratio,1);
        active = [active best_idx]
        nonactive(find(nonactive==best_idx,1)) = [];
```

```
46          best_ratio(k,j) = ratio(best_idx);
47          best_active{k,j} = active;
48      end
49  end
50  % Spara resultaten
51  save('best_ratio_fwd.mat','best_ratio');
52  save('best_active_fwd.mat','best_active');
```

B.5.2 Girig bakåteliminering

```
1   % Detta skript optimerar vår kNN enligt kapitel 3.4.2.
2   % Algoritmen som används är bakåteliminering
3
4   % Ladda testdata
5   load testbook; last = size(testbook,1);
6
7   % För resultat
8   best_active = {};
9   best_ratio=zeros(13,15);
10
11  % För varje k vi vill testa
12  for k = 1:13
13      % Skapa en full mängd E
14      active = 1:15;
15      % Komplementet till E
16
17      for j = 1:15
18          % Vi vill hitta lägsta felration.
19          % Initiera med ∞ för alla egenskaper!
20          ratio = Inf*ones(1,15);
21          % För alla ännu aktiva egenskaper
22          for t = active
23              syms = [];
24              % Klassificera gesten med alla
25              % egenskaper utom en
26              active_now=active;
27              active_now(find(active==t)) = [];
28              for i=1:last
29                  % Hämta det kända egenskapsvärdet
30                  real_sym = testbook(i,end-1);
31                  % Hämta egenskapsvektorn
32                  featv = testbook(i,active_now);
33                  % Klassificera gesten
34                  [obs_sym, dist] = ...
35                       getSymbol(featv,k,active_now);
36                  % Spara resultatet
37                  syms=[syms; real_sym obs_sym dist];
38              end
39              % Beräkna andel fel
```

```
            n_fel = size(syms( syms(:,1) ~= syms(:,2) ),1);
            n_okej = size(syms( syms(:,1) == syms(:,2) ),1);
            ratio(t) = n_fel/(n_fel+n_okej);
        end
        % Hitta "bästa" egenskapen och flytta ut ur E
        best_idx = find(min(ratio)==ratio,1);
        active(find(active==best_idx,1)) = [];
        best_ratio(k,15-j) = ratio(best_idx);
        best_active{k,15-j} = active;
    end
end
% Spara resultaten
save('best_ratio_bwd.mat','best_ratio');
save('best_active_bwd.mat','best_active');
```

B.6 Kod för HMM-träning

B.6.1 hmm_training.m

```
function [A,B] = hmm_training(states,symbols,O)
%HMM_TRAINING tränar en dold Markovmodell
%
% Denna funktion tränar en dold Markovmodell med
% Bakis-metoden (se kapitel 2.5.4)
%
% SYNOPSIS: [A,B] = hmm_training(states, symbols, O)
%
% INPUT states: Antalet states modellen ska ha
% symbols: Antalet symboler i kodboken
% O: Matris av radvisa observationssekvenser
%
% OUTPUT A: övergångsmatris för modellen
% B: Sannolikheter för symboler

% Skapa en uniform LR-modell
A = zeros(states);
for i = 1:states
    for j = i:states
        A(i,j) = 1/(states + 1 - i);
    end
end

% Skapa en B-matris där alla element är noll förutom
% de index som finns i O.
obs = reshape(O,1,size(O,1)*size(O,2));
unique_obs = [];
% Hitta unika observationer i O
for i = 1:symbols
```

```
30      if(find(obs == i) > 0)
31          obs_pos = find(obs == i);
32          unique_obs = [unique_obs, obs(obs_pos(1))];
33      end
34  end
35
36  % Beräkna sannolikheter för symboler
37  B = zeros(states,symbols);
38  for i = 1:states
39      B(i,unique_obs) = 1/length(unique_obs);
40  end
```