Moving a mouse without a mouse

Eindwerk voorgedragen tot het behalen van de graad van bachelor in de informatica

Gemaakt door: Goorts Patrik

Promotor: Prof. Dr. Philippe Bekaert
Co-promotor: Karin Coninx
Begeleiders: Cedric Vanaken en Lode Vanacken

Academiejaar 2006-2007

Abstract

Dit eindwerk beschrijft een methode om de gefilmde beelden van een hand om te zetten in een cursorbeweging. De opstelling is een enkele camera boven een werkoppervlak. Beide handen bevinden zich onder de camera met uitgestoken wijsvinger. De opgenomen beelden worden omgezet naar de HSB-kleurruimte, waar gemakkelijk de kans uit kan berekend worden welke kleur huid is. Dit wordt gedaan door standaard statistiekmethoden. Eenmaal de kans berekend wordt een variabele threshold berekend. Deze threshold wordt gebruikt om een binaire afbeelding te maken van wat huid is en wat niet. Deze afbeelding wordt bewerkt op ruis en gaten en er wordt een stel coördinaten uit berekend. De coördinaten worden aangepast zodat deze geschikt zijn om de cursor te bedienen. De rechterhand bestuurt de plaats van de cursor en de linkerhand bestuurt het klikken door de wijsvinger naar links en naar rechts te bewegen. De bewerking zorgt ervoor dat de cursor vlot beweegt, zonder dat hij rondspringt of extreme waarden aanneemt. Voor het gebruik moet er een leerfase worden doorlopen die het bewegingsgebied van de rechterhand vastlegt.

Er zijn een aantal usertests uitgevoerd die een aantal verbeteringen voorstellen. Zo kan er bijvoorbeeld meer feedback worden gegeven als er wordt geklikt. Over het algemeen wordt deze interactiemanier als handig ervaren.

Voorwoord

Ik heb het onderwerp muisinteractie door een gefilmde hand gekozen omdat het mij op het eerste zicht wel een interessante uitdaging leek om te verwezenlijken. Het is een relatief nieuwe manier om met de computer te communiceren. Omdat de interactie vlot moet gebeuren, is het een uitdaging om een voldoende robuuste en snelle aanpak te gebruiken.

Dit eindwerk is uiteraard niet zonder hulp gemaakt. Mijn dank gaat uit naar mijn promotor, Prof. Dr. Philippe Bekaert, en naar mijn begeleiders Cedric Vanaken en Lode Vanacken voor de hulp en opmerkingen die mij vooruit hebben geholpen. Verder wil ik Tom Haber bedanken voor het beschikbaar stellen van zijn library en de hulp daarmee. Als laatste wil ik iedereen bedanken die mij hebben gesteund en de moed hebben gegeven door te gaan.

Inhoudsopgave

1 Introductie

Computers worden met de jaren sterker, meer en meer mogelijkheden ontplooien zich, en de nood aan nieuwe interactiemethoden stijgt eveneens. Mensen willen op een gemakkelijkere manier communiceren met hun computer, met meer interactiemogelijkheden en dus ook meer vrijheidsgraden. De huidige mogelijkheden zijn beperkt in functie. Een muis kan enkel een (x,y) coördinaat meegeven, samen met een klik. De mogelijkheden zouden uitbreiden als er bijvoorbeeld 2 coördinaten zouden worden gebruikt.

Dit eindwerk gaat over het filmen van een hand om een muiscursor te bewegen. Het doel is een interessant alternatief aanbieden voor interactie met een computer. De mogelijkheid ontstaat om met 2 handen te werken. Met een hand heeft men meer bewegingsvrijheid dan met de huidige invoerapparaten, zoals een klassieke muis of een toetsenbord. Daarmee plaatst deze toepassing zich in het rijtje van invoerapparaten voor normaal computergebruik. De apparatuur is goedkoop, een eenvoudige webcam volstaat.

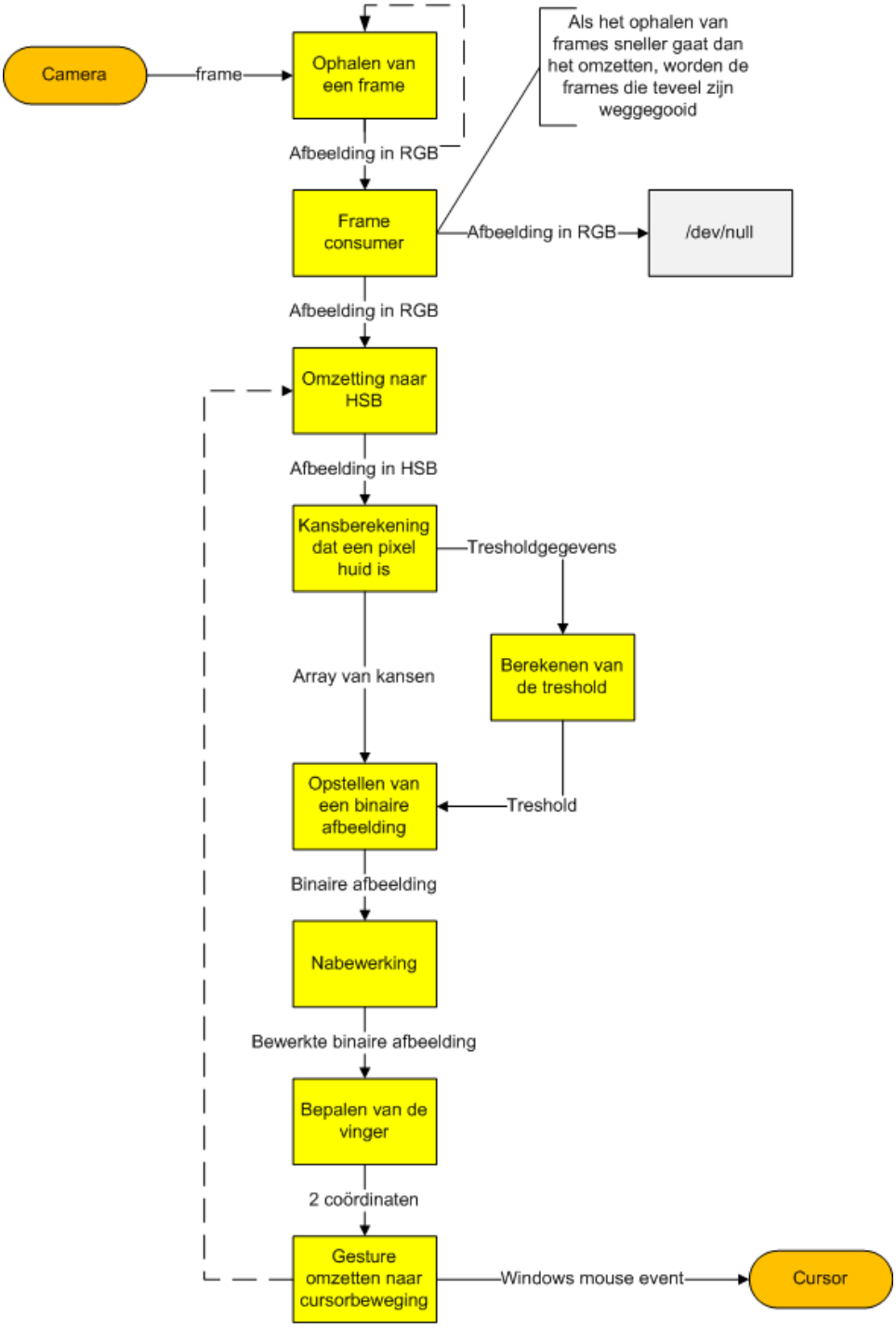
De voorgestelde aanpak werkt met een enkele camera boven een werkoppervlakte geplaatst en filmt de 2 handen tegelijkertijd. De wijsvinger van beide handen is uitgestoken. Beweging van de cursor gebeurt met de rechterhand en klikken met de linkerhand. Het extraheren van de hand van de achtergrond gebeurt door een algoritme gebaseerd op de kleur van huid in de HSV-kleurruimte (tech faq, 2007). De positie van de uitgestoken vinger wordt bepaald door een geoptimaliseerd scanlinealgoritme. Voor de cursorbewegingen wordt de Windows SDK gebruikt.

Een overzicht van de aanpak is weergegeven in figuur 1.

2 Opstelling

De basisopstelling is een enkele camera boven een werkoppervlakte geplaast, bij voorkeur in een kleur waar het huidherkenningsalgoritme niet op reageert. Het algoritme werkt in de HSV-kleurruimte en bijgevolg zijn de kleuren beige, bruin, rood, roze en oranje als achtergrond afgeraden. De opstelling is te zien in figuur 2.

Onder de camera bevinden zich beide handen. In deze aanpak dient de linkerhand voor het klikken en de rechterhand voor de positie van de cursor. Beide handen moeten zich constant onder de camera bevinden. De wijsvinger is uitgestoken naar voor. Tussen de 2 handen moet een verticale ruimte zijn voor de scheiding duidelijk te maken.



Figuur 1: Overzicht van de aanpak .

Figuur 2: Foto's van de opstelling. De camera is omkaderd.

2.1 Andere mogelijke opstellingen

Er zijn een aantal andere opstellingen mogelijk. Zo kan er bijvoorbeeld gebruik gemaakt worden van twee camera's, zodat ook de hoogte van de handen kan worden bepaald. Dan heeft de hand 3 vrijheidsgraden en zijn er meer mogelijkheden met gestures.

Twee camera's kunnen ook gebruikt worden om de 2 handen apart te filmen. Dit heeft als voordeel dat er geen leerfase moet doorlopen worden voor de rechterhand (zie 3.8). Momenteel moet het gebied afgebakend worden waar de rechterhand kan bewegen. Met 2 camera's heeft de rechterhand het hele beeld tot zijn beschikking.

Een andere mogelijke aanpak is filmen langs voor of langs onder. Het voordeel langs vanvoor filmen is dat ook webcams ingebouwd in laptops kunnen worden gebruikt. Dit is wel vermoeiender voor de handen, omdat deze omhoog moeten worden gehouden en de achtergrond (een persoon) is niet vast en kan ook huid bevatten.

Een camera langs onder geeft minder last bij het werken, maar heeft een glasplaat of dergelijke nodig en is duurder en moeilijker in opstelling.

Ook de hoofdbewegingen kunnen worden omgezet in een beweging van de cursor. Voor een commercieel product met dit uitganspunt, zie Skil (2007).

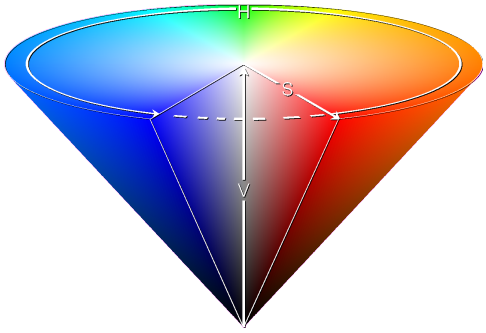
Er kan ook langs boven of onder op een drukgevoelige plaat of touchscreen worden gewerkt. Het klikken wordt dan door andere hardware afgehandeld en niet waargenomen door de camera.

Op de hand kan eveneens een tracker worden geplaats. Dit is een kleine markering op de hand die kan worden herkend door een camera. Op die manier kan de positie van een of meer trackers worden bepaald en daarmee ook de positie van de hand en vingers. Dan kunnen handen eveneens over elkaar worden gebruikt.

3 Algoritmes voor Handherkenning

3.1 Overzicht

De bedoeling van het algoritme is bepalen waar de vingertoppen van de wijsvinger zich bevinden. Daarom moet eerst de hand van de achtergrond worden gescheiden. Dit wordt gedaan door de kans te berekenen dat een bepaalde kleur huid is. Dit gebeurt het gemakkelijkste in de HSB-kleurruimte (zie 3.2.1). Dit levert een binaire afbeelding op die bepaalt wat hand is en



Figuur 3: Voorstelling van de HSB/HSV-kleurruimte .

wat niet. Uit deze afbeelding kan dan de positie van de vingertoppen worden bepaald en die vingertoppen bepalen de beweging van de cursor. Een overzicht is gegeven in figuur 1.

3.2 Omzetting naar een andere kleurruimtes

3.2.1 De kleurruimtes

De gangbare manier om kleuren voor te stellen in een computersysteem is door deze op te splitsen in rood-, groen- en blauwcomponenten. Er zijn ook verschillende andere systemen in gebruik die allemaal hun nut hebben. Kleuren kunnen bijvoorbeeld worden omgezet in cyaan, magenta en geel (CMY), zodat deze bruikbaar zijn voor een printer.

Interessante kleurruimtes voor computervisie zijn HSI (Hue, saturation, intensity) en HSB (ook wel HSV genoemd, met B voor brightness en V voor value). Deze voorstellingen sluiten meer aan bij hoe de mens kleuren waarneemt. De *hue* bepaalt pure de kleur en de *saturation* de verzadiging van deze kleur. Hue wordt uitgedrukt in graden en saturation in percentages. Helrood zal rood hebben als hue en een hoge saturation. De hue kan ook rood zijn en een heel lage saturation hebben. Dan zal de kleur grijsachtig zijn. Als laatste parameter is de helderheid van de kleur (de intensity, de brightness of de value). Deze waarde maakt het verschil tussen zwart en wit (zwart en wit hebben een saturation van nul en verschillen alleen in helderheid). De helderheid wordt uitgedrukt in een percentage. Een voorstelling van de HSB-kleurruimte en de invloed van de parameters is weergegeven in figuur 3

Deze kleurruimte is interessant voor computervisie, omdat de hue en saturation nauwelijks verandert als de belichting verandert. Deze kan natuurlijk

veranderen als er bijna geen licht is of als er sprake is van overbelichting.

Voor meer informatie over kleurruimte, zie Gonzalez and Woods (2002) hoofdstuk 6, tech faq (2007) en Wikipedia (2007) voor de omzetting.

3.2.2 Algoritme

In deze implementatie wordt de afbeelding omgezet naar de HSB-kleurruimte, zoals voorgesteld in Manresa et al. (2005). Dit gebeurt met de volgende formules (max en min zijn het maximum en het minimum van de (R,G,B) waarden):

$$H = \begin{cases} niet\ gedefinieerd & MAX = MIN \\ 60.\frac{G-B}{MAX-MIN} + 0 & MAX = R\ en\ G \geq B \\ 60.\frac{G-B}{MAX-MIN} + 360 & MAX = R\ en\ G < B \\ 60.\frac{B-R}{MAX-MIN} + 120 & MAX = G \\ 60.\frac{R-G}{MAX-MIN} + 240 & MAX = B \end{cases}$$

$$S = \begin{cases} 0 & MAX = 0 \\ 1 - \frac{MIN}{MAX} + 0 & anders \end{cases}$$

$$B = \frac{MAX-MIN}{2}$$

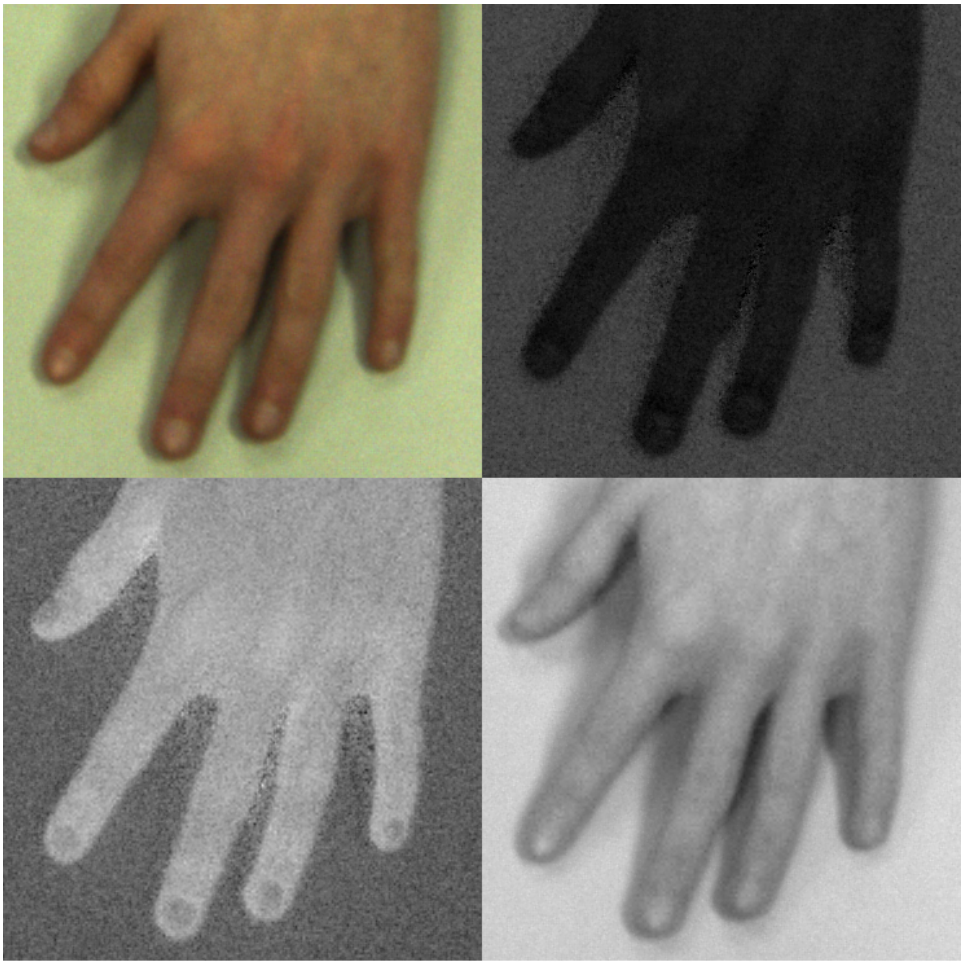
Een omzetting is weergegeven in figuur 4.

3.2.3 Andere aanpakken

Huidsegmentatie op basis van kleur kan ook in de RGB-kleurruimte worden uitgevoerd zoals beschreven in Chang and Robles-Mellin (2000). Deze aanpak is uitgetest, maar gaf resultaten die onvoldoende geschikt waren.

3.2.4 Uitgeteste methoden en verbeteringen

De eerste uitgeteste manier om de hand te scheiden van de achtergrond was door huidsegmentatie in de RGB-kleurruimte (Chang and Robles-Mellin, 2000). Na testen met verschillende afbeeldingen bleek dat deze methode niet toereikend was. Meer bepaald werd er veel valse huid herkend. De waarden werden geschaald zodat er pure kleuren ontstonden, bijvoorbeeld $rood = R/(R+G+B)$. Enkel de blauw en rood component werd gebruikt.



Figuur 4: Een afbeelding omgezet naar de HSB-kleurruimte. Van links naar rechts en van boven naar onder: Originele afbeelding, Hue, Saturation, Brightness .

Figuur 5: Een huidsample gebruikt voor het berekenen van de parameters .

Mogelijk werden er veel valse positieven gegeven door het feit dat verschillende kleuren pure RGB waarde hebben die op elkaar lijken. Bijvoorbeeld donkerrood ((239,59,57), pure kleuren (r = 0.67, b = 0.16)) wordt roze ((195, 73, 88), pure kleuren (r = 0.55, 0.25)).

Na experimenten met het HSB-algoritme op een zwarte achtergrond bleken de hue en de saturation zeer onbetrouwbaar te zijn rond een lage helderheid (het zwart). Daarom werden de hue en de saturation op nul gezet als de brightness kleiner is dan 15% of groter dan 85%. Daarmee verdween het probleem.

3.3 Extraction van de hand

3.3.1 Aanpak

Nadat de afbeelding is omgezet naar de HSB-kleurruimte kan de kans worden berekend dat een bepaalde pixel in die afbeelding huid is. Dit wordt gedaan zoals beschreven is in Manresa et al. (2005). Op voorhand worden een aantal parameters uit een sample(figuur 5) berekend die bepalen wat huid is. Die parameters zijn de gemiddeldenmatrix van Hue en Saturation $\bar{x}$ en de covariatiematrix E. De covariatiematrix is en veralgemening van de statistische variatie en beschrijft de variatie voor verschillende factoren, in dit geval de hue en de saturation (Weisstein, 2007). Deze waarden worden dan gebruikt in de formule:

$$P(x) = \frac{1}{\sqrt{(2\Pi)^2 \left|E\right|}} e^{-1/2(x-\bar{x})E^{-1}(x-\bar{x})^T}$$

Deze formule heeft als parameter de vector x met de hue en saturation en geeft de kans terug dat die pixel huid is gebaseerd op de vooraf berekende parameters.

3.3.2 Andere aanpakken

Een andere mogelijke aanpak voor de handherkenning is door de achtergrond vooraf te filmen en het verschil te bepalen met de te verwerken beelden. Zie bijvoorbeeld Noriega and Bernier (2007) en Kim et al. (2006). Deze algoritmen zijn meestal nogal complex en niet geschikt voor real-time toepassingen.

3.4 Thresholdberekening

Er is nu een array gemaakt waarbij de elementen de kansen voorstellen dat de overeenkomende pixel in de afbeelding huid is. Om deze array om te zetten in een binaire afbeelding, moet voor elke kans bepaald worden of dit huid is of niet. Daarvoor wordt een threshold berekend die bepaalt welke kans huid is en welke niet.

3.4.1 Aanpak

De threshold wordt berekend volgens de methode zoals beschreven op Chang and Robles-Mellin (2000). Hier wordt de volgende bewering gedaan over de ideale threshold:

> The adaptive thresholding is based on the observation that stepping the threshold value down may intuitively increase the segmented region. However, the increase in segmented region will gradually decrease (as percentage of skin regions detected approaches 100%), but will increase sharply when the threshold value is considerably too small that other non-skin regions get included. The threshold value at which the minimum increase in region size is observed while stepping down the threshold value will be the optimal threshold.

Meer concreet wordt de threshold per kans van 0.55 naar 0.05 verlaagd in stappen van 0.1 en wordt er bijgehouden hoe vaak de kans voor deze kans boven de geteste waarde ligt. Nadat alle pixels zijn behandeld, wordt het kleinste interval berekend. Concreet betekent dit dat het verschil tussen het aantal voor 0.55 en 0.45 wordt berekend, tussen 0.45 en 0.35, ... Het kleinste verschil bepaalt dan de ideale threshold. Er wordt voor elke kans berekend of deze boven of onder de threshold ligt. Indien deze eronder ligt, wordt deze gemarkeerd als geen huid, erboven is wel huid. Zo ontstaat een binaire afbeelding.

3.4.2 Andere aanpakken

De threshold kan ook een vaste waarde aannnemen, maar dit is niet zo een geschikte manier, omdat de opnames niet helemaal kunnen overeenkomen

met de vooraf bepaalde samples en dan kan de kans laag zijn voor huid.

3.5 Nabewerking

3.5.1 Aanpak

Omdat de binaire afbeelding gaten kan bevatten, wordt er een dilatie toegepast, zoals beschreven in Gonzalez and Woods (2002). Het gebruikte masker is:

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & x & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Het principe is dat over elke pixel in de binaire afbeelding dit masker wordt gelegd met de x op die pixel. Als de pixel 1 is, wordt in de binaire afbeelding elke pixel op 1 gezet waar in het masker een 1 staat. Zo wordt de afbeelding breder gemaakt en worden kleine gaten opgevuld, omdat de omliggende pixels worden uitgebreid. Een voorbeeld is hieronder gegeven:

$$\begin{pmatrix} . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . \\ . & . & x & x & . & . & . & x & x & . & . \\ . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . \end{pmatrix}$$

wordt

$$\begin{pmatrix} . & . & x & x & . & . & . & x & x & . & . \\ . & x & x & x & x & . & x & x & x & x & . \\ x & x & x & x & x & x & x & x & x & x & x \\ . & x & x & x & x & . & x & x & x & x & . \\ . & . & x & x & . & . & . & x & x & . & . \end{pmatrix}$$

Het gat tussen de 2 pixelgroepen is opgevuld en ze zijn nu verbonden.

Tijdens dit proces worden eveneens legen rijen en kolommen gedetecteerd en gemarkeerd. Deze informatie wordt in de volgende stap gebruikt.

3.5.2 Andere aanpakken

Een andere manier om grote groepen pixels uit de afbeelding te halen of te bewerken, is door *edge detection*, *edge linking* en *boundary detection* algoritmen (Gonzalez and Woods, 2002). Deze creëren een verzameling van lijnen waar dan de hand kan worden uitgehaald. De vingertoppen kunnen ook herkend worden door naar cirkels te zoeken, bijvoorbeeld door een Hough

transformatie. Zie voor de transformatie Rhody (2005) en voor een demo Chan (2004).

3.6 Gesture recognition

3.6.1 Aanpak

In de vorige stap zijn de lege rijen en kolommen bepaald. Eerst worden de 2 breedste stukken afbeelding bepaald waar data in elke kolom staat. Deze 2 stukken bevatten de 2 handen. Dan wordt de positie van het hoogste punt per afbeelding bepaald. Die bepalen de positie van de uitgestoken vinger.

Voor de linkerhand wordt ook een punt lager dan het hoogste punt berekend, zodat de hoek kan berekend worden. Er wordt het midden van de vinger gebruikt, zodat de afrondingen van de vinger geneutraliseerd worden.

3.6.2 Controle op correctheid

Voor de positie van de vinger wordt bepaald, wordt er eerst gecontroleerd hoe breed de stukken zijn waar de handen zich (zouden moeten) bevinden. Als een deel kleiner is dan een vijfde van de breedte van de opname, dan wordt de cursor niet bewogen. Waarschijnlijk bevindt een van de handen zich niet langer in het beeld en is de analyse niet correct. Deze controle vermijdt nutteloos rondspringen van de cursor.

3.6.3 Andere gestures

In deze implementatie worden enkel vingertoppen herkend. Een voorbeeld van een andere aanpak is de detectie van een gesloten hand met duim en wijsvinger tegen elkaar, zoals beschreven in Wilson (2006). Er kan ook gebruik worden gemaakt van een open hand of een vuist. In Manresa et al. (2005) wordt een alfabet voorgesteld gebaseerd op gespreide vingers, een gesloten hand en een vuist.

Er kan ook gebruik gemaakt worden van andere hardware voor het klikken te simuleren. Men kan bijvoorbeeld een touchscreen gebruiken of een eyetracker om knipperen te detecteren.

3.7 Cursorbewegingen

3.7.1 Aanpak

Na het bepalen van de vereiste coördinaten, worden deze bewerkt zodat deze bruikbaar zijn om de cursor te bewegen. Allereerst worden de eerste 30 opnames niet in beschouwing genomen, zodat de gebruiker de kans heeft zijn handen te plaatsen voor de leerfase (zie 3.8). In de leerfase worden

de coördinaten van de rechterhand gebruikt voor het bepalen van het bewegingsgebied. Na de leerfase worden de coördinaten bewerkt, zodat een zachte muisbeweging wordt bereikt. Indien dit niet wordt gedaan, maakt de cursor kleine schokkerige bewegingen die kunnen storen. Deze worden nu verminderd. De formule voor het verzachten is:

$$\begin{bmatrix} x_{nieuw} \\ y_{nieuw} \end{bmatrix} = (1-\alpha) * \begin{bmatrix} x_{huidig} \\ y_{huidig} \end{bmatrix} + \alpha * \begin{bmatrix} x_{vorig} \\ y_{vorig} \end{bmatrix}$$

Voor de implementatie is α ingesteld op 0.4.

Als er voor de linkerhand wordt gedetecteerd dat een muisknop wordt losgelaten, wordt dit niet onmiddellijk doorgevoerd. Er moeten 3 beelden zijn voordat de muisknop wordt losgelaten. Dit wordt gedaan om te vermijden dat er per ongeluk wordt losgelaten als de beeldherkenning voor 1 beeld faalt of als de gebruiker zijn vinger juist niet schuin genoeg houdt.

Voor het verplaatsen van de cursor zelf wordt gebruik gemaakt van de Windows SDK, meer bepaald de mouse_event functie (Microsoft, 2007). Deze functie kan de cursor verplaatsen en muisklikken simuleren.

3.7.2 Andere aanpakken

De posities van de hand kunnen naar elk willekeurig programma gestuurd worden, bijvoorbeeld een tekenprogramma, een digitale atlas, een interface zonder klikken (Frank, 2007), ...

3.8 Gebruik

Onmiddellijk nadat het programma gestart is, moet er een leerfase doorlopen worden. De rechterhand moet bewogen worden om het bewegingsgebied af te bakenen. Het volstaat naar de randen en zeker naar de hoeken te bewegen. Dit is nodig, omdat de rechterhand niet het hele gebied tot zijn beschikking heeft. De linkerhand bevindt zich in hetzelfde beeld en tussen de 2 handen moet zich een verticale ruimte bevinden. Als de rechterhand klaar is, moet met de linkerhand geklikt worden en wordt er een rechthoekige ruimte bepaald waar de rechterhand kan bewegen.

In een oudere versie van de implementatie is het ook nodig dat de linkerhand wordt vastgelegd. In die versie werd de gesture enkel op de x-coördinaat gebaseerd. Dit hield in dat de hand stil moest blijven liggen, wat geen praktische oplossing is. Nu wordt de gesture bepaald door de hoek van de vinger, zodat de linkerhand wel kan bewegen.

3.8.1 Gestures

Klikken gebeurt met de linkerhand. Een overzicht van het bewegen is weergegeven in figuur 6. Klikken is de wijsvinger van het midden naar links en terug naar het midden bewegen. Links slepen is de wijsvinger naar links brengen en met de rechterhand slepen. Loslaten is de wijsvinger terug naar het midden brengen. De bewegingen voor rechts klikken zijn analoog. Het is van geen belang waar de linkerhand zich bevindt, enkel de hoek van de vinger wordt beschouwd.

Bewegen van de cursor gebeurt met de rechterhand. De hand moet zich bewegen in het gebied dat in de leerfase is bepaald (het bewegingsgebied). Als de hand hierbuiten gaat, wordt de cursor bewogen op de rand van het bewegingsgebied. In tegenstelling tot een gewone muis, moet de hand zich terug in het bewegingsgebied begeven om terug in die richting te bewegen. Als bijvoorbeeld de hand teveel naar links gaat, blijft de cursor tegen de linkerkant van het scherm staan (de hoogte kan wel nog worden bepaald door de hand). Als de hand terug in het bewegingsgebied komt, beweegt de cursor weg van de rand van het scherm.

De rechthoek die bepaald is in de leerfase wordt als het ware over het scherm gelegd. De hoeken van het bewegingsgebied komen overeen met de hoeken van het scherm. Dit houdt in dat als de verhouding van het bewegingsgebied niet overeenkomt met de verhouding van het scherm, de cursor in 1 richting sneller beweegt dan in de andere richting. Als bijvoorbeeld een vierkant gebied wordt vastgelegd en een breedbeeldscherm wordt gebruikt, beweegt de cursor verder over de horizontale as dan over de verticale as voor dezelfde afstand voor de hand.

4 Implementatie

4.1 Algemene details

- *Besturingssysteem:* Windows XP
- *Programmeertaal:* C++
- *Omgeving:* Visual studio 2003

4.2 Gebruikte libraries

- *Streams library van het EDM:* Deze library is geschreven door Tom Haber en wordt gebruikt om de webcam in te lezen. Er wordt eveneens gebruik gemaakt van de threads en shared memory functionaliteit.
- *Windows SDK:* Deze wordt gebruikt om de cursor te bewegen.

Figuur 6: Overzicht van de gestures om te klikken .

4.3 Algemene aanpak

Er lopen 2 threads in het programma. 1 thread is verantwoordelijk voor het opvangen van de camerabeelden. De andere thread berekent de positie van de handen en zorgt voor de beweging van de cursor. De 2 threads geven afbeeldingen door via gedeeld geheugen.

4.4 Verloop van het programma

4.4.1 Imageproducer

Deze klasse leest een stream in en geeft deze door aan Raw_Imageconsumer. Hier wordt eveneens het gedeeld geheugen geïnitialiseerd. Deze klasse loopt in een aparte thread.

4.4.2 Raw_Imageconsumer

Deze klasse vangt alle afbeeldingen op die de imageproducer produceert en plaatst ze in het gedeeld geheugen. Als het gedeeld geheugen niet vrij is, wordt de afbeelding gewist om ophoping te voorkomen.

4.4.3 ImageConverter

In deze klasse gebeurt het omzetten. De klasse haalt de afbeelding uit het gedeeld geheugen en voert het algoritme uit. Hier wordt eveneens de cursor verplaatst. Het aantal functies is beperkt om de overhead te verminderen. Deze klasse loopt in een aparte thread.

Hier zijn eveneens opties voorzien om de afbeelding te verkleinen of te spiegelen over een as. Dit beïnvloedt enkel het aflopen van de afbeelding, er is geen extra code die wordt uitgevoerd.

4.4.4 Parameters voor het algoritme

Vooraf is een sample genomen van de huid. Daar zijn de nodige parameters uitgehaald (zie 3.3.1). Deze parameters zijn berekend in een matlab-programma.

4.5 Bespreking, tekortkomingen en uitbreidingen

4.5.1 Extraction van de hand

De extraction in de voorgestelde aanpak loopt vrij goed, maar kan gaten maken in de regio's. Daarom wordt de afbeelding gedileerd, maar dit heeft ook als effect dat de ruis vergroot wordt. Dit kan opgelost worden door de afbeelding eerst te eroderen (Gonzalez and Woods, 2002), maar dit is niet

echt vereist, kost opnieuw rekenkracht en kan vingers afkappen.

De thresholdberekening geeft goede resultaten, omdat deze adaptief is. De threshold zorgt dat er grote gebieden herkend worden. Een kleine en niet volledige respons zal een relatief grote respons hebben als de threshold een beetje wordt verhoogd.

Het herkennen van de hand met behulp van de kleur heeft enkele nadelen. Het algoritme mag niet reageren op de achtergrond. Kleuren zoals rood of oranje kunnen als huid worden herkend, zodat er fouten worden gecreëerd in het herkennen van de hand. Ook de achtergrondverlichting kan voor problemen zorgen. Over- of onderbelichting kan de kleur van de huid te veel aanpassen, zodat deze te wit of te zwart wordt om als huid te worden herkend. Op dezelfde manier is de huidskleur van belang. Een te donkere of te lichte huid kan problemen opleveren.

Als oplossing kan dit algoritme worden uitgebreid met een background subtraction algoritme dat niet gebaseerd is op kleur, bijvoorbeeld door de achtergrond op voorhand op te nemen. Op die manier wordt het algoritme betrouwbaarder.

Om valse vlekken te vermijden kan men meer nabewerking doen door te controleren hoe groot de gevonden regio's zijn en of deze kunnen overeenkomen met handen. Een andere manier is door gebruik te maken van beelden die vlak daarvoor zijn opgenomen. Huid kan niet uit het niets verdwijnen en verschijnen. Deze technieken vereisen weliswaar meer rekenkracht en mogelijk houdt dit de real-time werking van het programma tegen.

4.5.2 Bepalen van de gestures

Deze aanpak gebruikt een simpele en snelle manier om de positie van de vinger te bepalen. Dit is ideaal voor realtimetoepassingen, maar heeft wat moeilijkheden.

Het algoritme heeft moeite met valse vlekken in de afbeelding. Als een vrij groot gedeelte vals wordt herkend als huid, kan dit als vinger worden herkend en kan de cursor gaan rondspringen.

Een andere moeilijkheid is het herkennen van gestures die niet ondersteund worden. Een open hand werkt ook, maar er wordt mogelijk een andere vinger herkend. Er zal altijd het hoogste en meest linkse punt worden herkend.

Het opdelen van de afbeelding in 2 stukken (linker- en rechterhand) is in deze

aanpak enkel mogelijk als er een verticale ruimte is tussen de 2 handen. Een verbetering zou zijn als de ruimte niet verticaal moet zijn, maar elke vorm kan aannemen. Dan kunnen bijvoorbeeld de 2 handen boven elkaar worden geplaatst. Dit houdt ook in dat het bepalen van de linker- en rechterhand ingewikkelder is. In deze aanpak gebeurt dit door de linkerhand te herkennen als de vinger met de kleinste x-coördinaat.

Eventueel kunnen de handen boven elkaar worden geplaatst, als er gebruik wordt gemaakt van 2 camera's.

4.5.3 De gestures

De voorgestelde gestures zijn simpel in gebruik en er is geen training van de gebruiker nodig. Dit maakt de toepassing onmiddellijk bruikbaar.

Een nadeel is dat de gebruiker voortdurend zijn wijsvinger moet uitgestoken houden. Dit kan vermoeiend worden. Alternatieven zijn een open hand, een vuist of een cirkel met duim en wijsvinger, zoals in Wilson (2006). In Manresa et al. (2005) wordt een alfabet voorgesteld gebaseerd op gespreide vingers, een gesloten hand en een vuist.

Ook het klikken kan vermoeiend en traag zijn, omdat de beweging relatief groot is. Andere manieren kunnen zijn: duim tegen de hand duwen, klikken zoals met een normale muis (dit vereist dat ook de hoogte van de hand kan worden bepaald), knipperen met de ogen (vereist een eye tracker), klikken op een touchscreen, ...

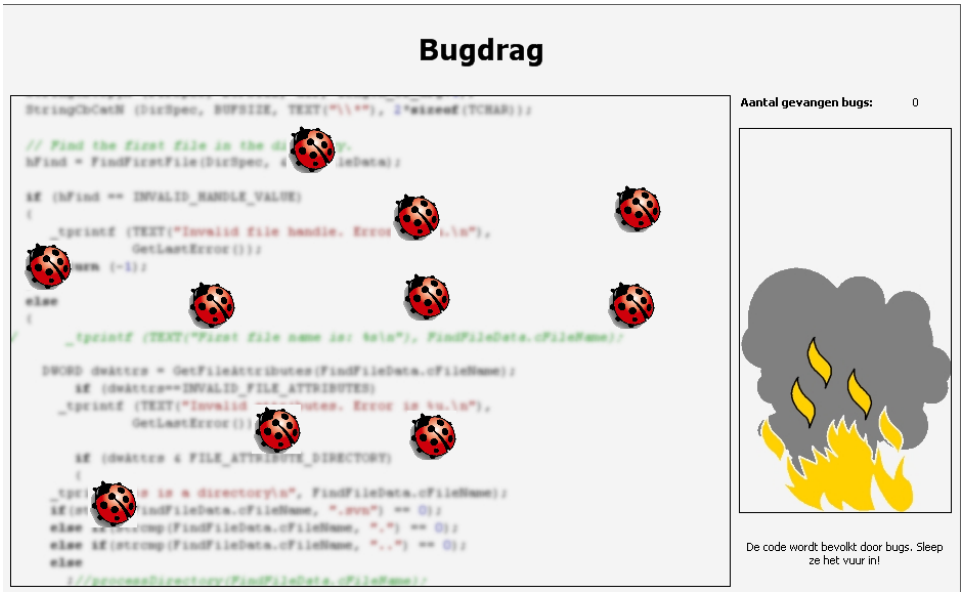
Een andere moeilijkheid is het verplicht gebruik van 2 handen. Indien het klikken gebeurt zoals met een normale muis of met de ogen, kan 1 hand voldoende zijn.

Ook andere interactiemethoden zijn mogelijk, bijvoorbeeld een interface zonder klikken (Frank, 2007) of met een drukgevoelige tafel.

5 Testapplicatie en tests in gebruik

5.1 Een testapplicatie

Om het gebruik te demonstreren, is er een voorbeeldapplicatie geschreven die vereist dat er veel wordt bewogen met de muis, zowel gewoon bewegen als slepen van objecten. De applicatie is een spelletje, genaamd *bugdrag*, waarbij een aantal *bugs* weggesleept moeten worden van 1 gebied naar een andere. De bugs bewegen rond in een rechthoek en versnellen als er minder en minder bugs zijn. Deze aanpak heeft als gevolg dat de gebruiker verplicht



Figuur 7: Screenshot van bugdrag .

is naar het scherm te kijken en moet volgen wat er gebeurt; er kan niet naar de handen worden gekeken. Door de versnelling is er veel beweging van de handen nodig en een voldoende coördinatie van de gebruiker. Dit is een betere test dan een vaak gebruikt tekenprogramma of een simpel interfaceprogramma, omdat hierbij een uitdaging is die de gebruiker afleidt van de interactiemethode zelf.

5.1.1 Implementatiedetails

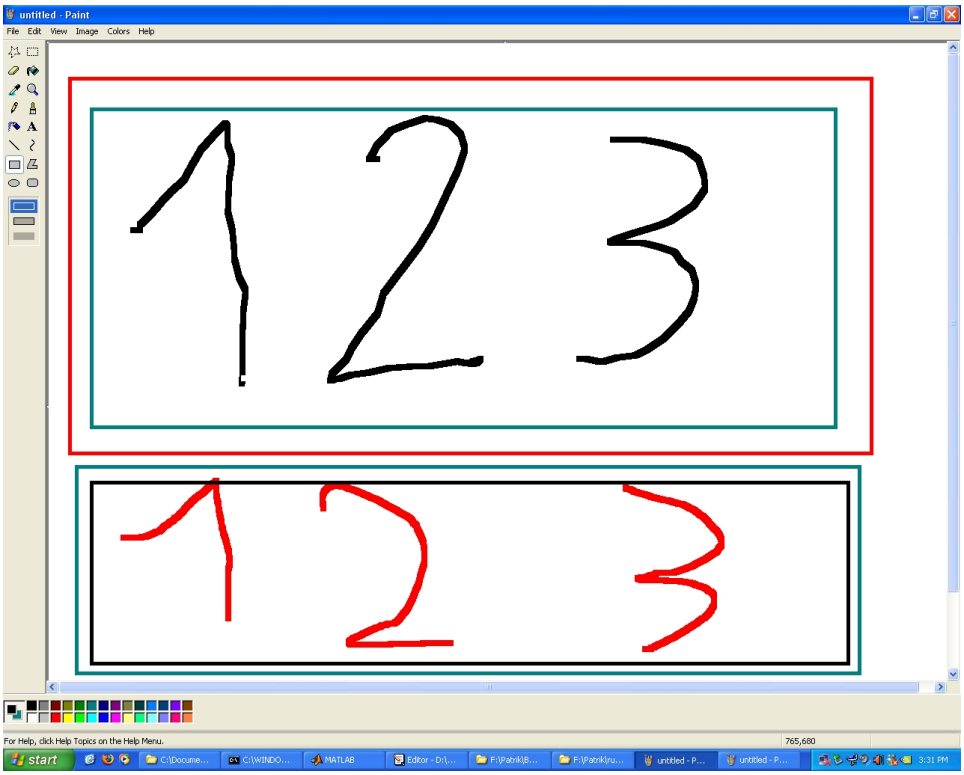
Het programma is geschreven in Visual Basic.Net, omdat dit een simpele manier aanbiedt voor het opbouwen van interfaces. Het programma gebruikt geen speciale technieken om te werken. Een screenshot is te zien in figuur 7.

5.2 User test

5.2.1 Opzet

Er zijn 5 personen gevraagd de voorbeeldapplicatie te gebruiken. Alle gebruikers waren tussen de 20 en 22 en hadden jarenlange ervaring met muisgestuurde interfaces. Ze moesten zelf het programma openen in Windows Explorer en de leerfase voor het kallibreren van de muis doorlopen (zie 3.8). Enkele gebruikers hebben ook een test gedaan in Paint. Zie afbeelding 8

De volgende vragen werden achteraf gesteld:



Figuur 8: Screenshot van een test in Paint. De bovenste is gemaakt met de gefilmde hand, de onderste met de muis.

- Hoe vlot werkt het programma?
- Werkt het gemakkelijker dan een normale muis? Is het handig? Zou je er gemakkelijk mee kunnen werken?
- Zijn er verbeteringen mogelijk?
- Nog extra commentaar?

5.2.2 Resultaten

Elke gebruiker kon de voorbeeldapplicatie vlot opstarten en gebruiken. Sommige gebruikers moesten even wennen aan de bediening. De volgende commentaar is door de gebruikers gegeven:

- Ondanks de soms schokkerige bewegingen werkt de applicatie handig.
- Het zou gemakkelijker zijn als werd aangegeven wanneer er werd geklikt. Dit kan bijvoorbeeld gedaan worden door een cursor die van kleur verandert of door een geluidje.
- De bediening zou ook met de voeten kunnen gebeuren, zodat de handen vrij zijn voor het toetsenbord.
- De techniek kan handig zijn als deze wordt gebruikt op een PDA, zodat er geen touchscreens moeten worden gebruikt. Dit geeft vlekken en een vinger is soms dikker dan het te klikken object. Een mogelijk andere oplossing is een techniek die ze *shift* noemen. Voor kleine objecten wordt er een kopie van de regio onder de vinger elders op het scherm geplaatst en ziet de gebruiker wat hij aanklikt. Zie voor meer info Vogel and Baudisch (2007).
- Er is geen gevoelsfeedback, men heeft geen muis in handen. Dit geeft een raar gevoel.
- De hand kan ook een touchpad simuleren, dus dat de vinger wordt opgetild en ergens anders wordt neergezet zonder dat de cursor beweegt. Dit kan worden bereikt met een andere gesture, bijvoorbeeld de midden- en wijsvinger tegen elkaar en van elkaar af bewegen.
- Het is een intuïtieve manier om te schrijven of om te bewegen in het algemeen. Het is niet zo artificieel als een muis, maar je sleept effectief met een vinger over (of boven) een oppervlak.
- Het is moeilijk om te bepalen wat het oppervlak is waar de rechterhand in kan bewegen. Een visuele steun op de tafel kan handig zijn.

- De meeste gebruikers waren van menig dat de toepassing beter kan werken voor specifieke toepassingen, maar voor normaal gebruik kan een muis misschien handiger zijn. Dit kan ook te maken hebben met het feit dat de gebruiker niet veel ervaring met de toepassing heeft.

De volgende dingen zijn tijdens de test opgemerkt:

- De gebruikers hebben de neiging hun handen en vingers te plooien en op te tillen van de tafel. Dit kan bibberen van de cursor veroorzaken.
- De hoek voor te klikken is mogelijk iets te groot. Dit kan mogelijk ook worden ingesteld in de leerfase.
- Het bewegen zelf met de rechterhand gaf geen problemen.
- De gebruikers kijken niet naar hun handen, zodat ze meer aandacht op het scherm kunnen vestigen.
- Gebruikers hebben de neiging hun hand op te tillen en ergens anders neer te zetten, zoals met een touchpad. Dit werkt niet, omdat de positie wordt doorgegeven als absolute waarden. Om dit op te lossen is een systeem nodig dat *optillen* detecteert.

6 Conclusie

In dit eindwerk is een techniek voorgesteld om de beweging van handen om te zetten in de beweging van een cursor door gebruik te maken van een camera. De voorgestelde aanpak gebruikt een extraction algoritme gebaseerd op de kleur van de huid. De gestures zijn simpel gehouden voor het gemak van de bediening en de snelheid van verwerking. Aan de aanpak kunnen nog dingen verbeterd worden, vooral op het vlak van robuustheid. Denk hierbij aan de achtergrondkleur die valse positieven kan geven of het gebruik van niet-ondersteunde gestures. Ondanks deze tekortkomingen kan het programma zeker een meerwaarde bieden voor een cursorgebaseerde interface.

Referenties

Chan, S. C., 2004: Hand gesture recognition. `http://www.cim.mcgill.ca/~schan19/research/research.html`, laatst bekeken op 20 augustus 2007.

Chang, W. F. H. and U. Robles-Mellin, 2000: Face detection. `http://www-cs-students.stanford.edu/~robles/ee368/main.html`, laatst bekeken op 2 augustus 2007.

Frank, A., 2007: Don't click. `http://www.dontclick.it/`, laatst bekeken op 20 augustus 2007.

Gonzalez, R. C. and R. E. Woods, 2002: *Digital image processing*. Prentice Hall, 793, second edition.

Kim, H., R. Sakamoto, I. Kitahara, T. Toriyama, and K. Kogure, 2006: Robust Foreground Segmentation from Color Video Sequences Using Background Subtraction with Multiple Thresholds (Videos). *Technical report of IEICE. PRMU*, **106**(376), 135–140.

Manresa, C., J. Varona, R. Mas, and F. Perales, 2005: Hand Tracking and Gesture Recognition for Human-Computer Interaction. *Electronic Letters on Computer Vision and Image Analysis*, **5**(3), 96–104.

Microsoft, 2007: mouse_event function. `http://msdn2.microsoft.com/en-us/library/ms646260.aspx`, laatst bekeken op 20 augustus 2007.

Noriega, P. and O. Bernier, 2007: Real time illumination invariant background subtraction using local kernel histograms.

Rhody, H., 2005: Lecture 10: Hough circle transform. `http://www.cis.rit.edu/class/simg782/lectures/lecture_10/lec782_05_10.pdf`, laatst bekeken op 20 augustus 2007.

Skil, 2007: Muisaanpassingen. `http://www.skil-nv.com/nlcombedbody2.htm`, laatst bekeken op 20 augustus 2007.

tech faq, 2007: Wat is hsv? `http://www.tech-faq.com/hsv.shtml`, laatst bekeken op 20 augustus 2007.

Vogel, D. and P. Baudisch, 2007: Shift: a technique for operating pen-based interfaces using touch. *Proceedings of the SIGCHI conference on Human factors in computing systems* 657–666.

Weisstein, E. W., 2007: Covariance matrix. `http://mathworld.wolfram.com/CovarianceMatrix.html`, laatst bekeken op 20 augustus 2007.

Wikipedia, 2007: Transformation between hsv and rgb. `http://en.wikipedia.org/wiki/HSV_color_space#Transformation_between_HSV_and_RGB`, laatst bekeken op 20 augustus 2007.

Wilson, A. D., 2006: Robust computer vision-based detection of pinching for one and two-handed gesture input. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, ACM Press, New York, NY, USA, 255–258.