Auteursrechterlijke overeenkomst

Opdat de Universiteit Hasselt uw eindverhandeling wereldwijd kan reproduceren, vertalen en distribueren is uw akkoord voor deze overeenkomst noodzakelijk. Gelieve de tijd te nemen om deze overeenkomst door te nemen, de gevraagde informatie in te vullen (en de overeenkomst te ondertekenen en af te geven).

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling met

Titel: CAPTCHAs breken
Richting: master in de informatica - multimedia Jaar: 2009

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Ik ga akkoord,

VAN MIERLOO, Ben

Datum: 14.12.2009

CAPTCHAs breken

Ben Van Mierloo

promotor :
Prof. dr. Philippe BEKAERT

Eindverhandeling voorgedragen tot het bekomen van de graad
master in de informatica multimedia

Voorwoord

Bij het beschouwen van de opgegeven Master thesis onderwerpen werd mijn aandacht vooral getrokken door het breken van CAPTCHA's. Mijn interesse voor digitale beeldverwerking is altijd groot geweest. Ik ben dus ook met veel enthousiasme aan dit eindwerk begonnen. Naarmate ik dieper op de materie inging geraakte ik enorm geboeid door de vele mogelijkheden van dit onderzoeksgebied. Ik ben dan ook fier dit werk te mogen voorstellen. Ik heb een poging gedaan om een zo volledig mogelijk beeld te geven van mijn opzet en het resultaat. Ik heb dit echter niet alleen verwezenlijkt. Graag zou ik van de gelegenheid gebruik maken om de volgende mensen te bedanken:

Prof. dr. Philippe Bekaert, mijn promotor, voor de nodige richtlijnen en informatie om dit eindwerk tot een goed einde te brengen.

Mark Gerrits en *Karel Frederix*, mijn begeleiders, voor het meermaals nalezen van mijn eindwerktekst en het geven van nuttige tips en ideen die hebben bijgedragen tot het eindresultaat.

Mijn ouders die me de kans gaven om Informatica studies te volgen, en mij voortdurend gesteund hebben bij het maken van dit eindwerk.

Tenslotte *mijn medestudenten*, voor hun hulp en ideeën omtrent de mogelijkheden van dit eindwerk.

Zonder hen zou dit eindwerk niet tot stand gekomen zijn.

Abstract

Het doel van deze thesis is het analyseren en breken van CAPTCHA's. In deze thesis zullen twee technieken besproken worden met betrekking tot het herkennen van karakters. Deze technieken zijn het Shape Context Matching algoritme en het Distance Transform Matching algoritme. Beide technieken kunnen vormen herkennen in afbeeldingen. Nadien wordt nog een combinatie gemaakt van beide technieken om zo een beter resultaat te verkrijgen bij het analyseren en breken van CAPTCHA's. Er worden een aantal CAPTCHA's uitgebreid besproken. Zowel de sterke als de zwakke punten van de hedendaagse CAPTCHA's worden uit de doeken gedaan.

Inhoudsopgave

Lijst van figuren

Hoofdstuk 1

Inleiding

Iedereen komt tegenwoordig in aanraking met CAPTCHA's. Het zijn de afbeeldingen die ervoor moeten zorgen dat spambots of andere software geen toegang krijgen tot een bepaalde site. De afbeeldingen beelden typisch een reeks karakters af die door de gebruiker moeten ingegeven worden. Meestal worden de karakters vervormd en wordt er ruis geïntroduceerd in de afbeelding om zo te zorgen dat algoritmes de afbeelding niet kunnen lezen.

Het doel van deze thesis is beter inzicht te krijgen in het analyseren van CAPTCHA's. CAPTCHA's worden tegenwoordig vaak gebruikt op het internet in verband met beveiliging. De vraag is natuurlijk hoe veilig een CAPTCHA echt is. Kunnen deze CAPTCHA's garanderen dat enkel mensen toegang krijgen? Zijn de CAPTCHA's met andere woorden te breken door algoritmes.

Er worden in deze thesis verschillende technieken besproken die helpen in het analyseren van CAPTCHA's. De voor- en nadelen van de technieken zullen uitvoerig besproken worden. Ook zal getracht worden een combinatie te maken van de technieken om zo het analyseren van de CAPTCHA gemakkelijker en efficiënter te maken.

Aan de hand van de resultaten van de besproken algoritmen zal uitgelegd worden waarom sommige CAPTCHA's een goede beveiliging zijn, en anderen niet.

Hoofdstuk 2

Wat is een CAPTCHA

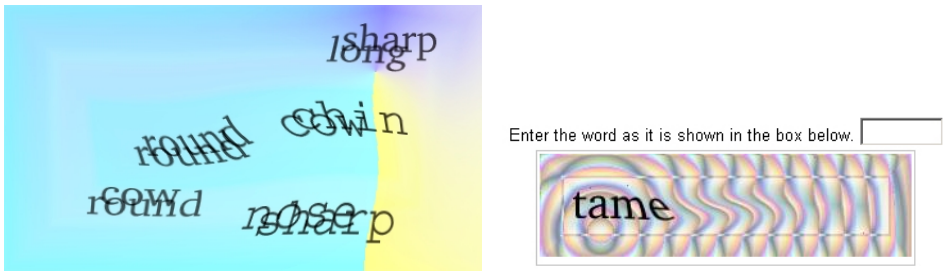
De afkorting CAPTCHA betekent "**C**ompletely **A**utomated **P**ublic **T**uringtest to tell **C**omputers and **H**umans **A**part". In het jaar 2000 werd de term geïntroduceerd door Luis von Ahn, Manuel Blum en Nicholas J. Hopper van de Carnegie-Mellon Universiteit [1]. Een CAPTCHA is een computer gegenereerde puzzel die kan uitmaken of een gebruiker menselijk of artificieel is.

CAPTCHA's komen voor in verschillende vormen. De bekendste is de tekstuele visuele CAPTCHA zoals te zien in figuur 2.1. Dit zijn afbeeldingen met vervormde tekst die moeilijk te ontcijferen zijn door een standaard tekstherkennings algoritme. Een menselijke gebruiker is echter gemakkelijk in staat deze karakters te lezen.

Figuur 2.1: Een voorbeeld van een visuele CAPTCHA

De bedenkers van de CAPTCHA term hebben ook de eerste CAPTCHA gecreeërd. Ze noemden deze CAPTCHA "Gimpy". Later is een vereenvoudigde versie van Gimpy gebruikt door Yahoo (zie figuur 2.2).

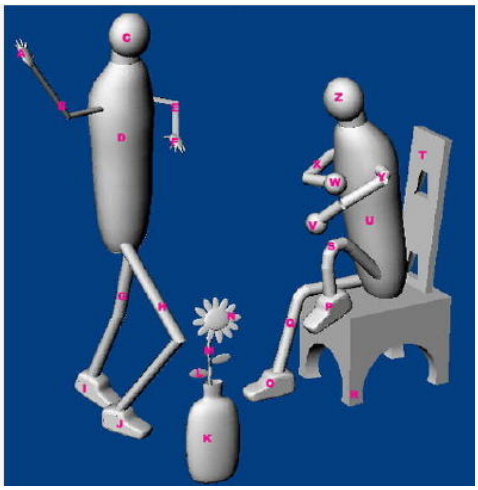
Onder de noemer van visuele CAPTCHA's, kunnen we ook nog de niet tekstuele visuele



(a) Gimpy (b) Oude Yahoo CAPTCHA

Figuur 2.2: Gimpy en Yahoo CAPTCHA

CAPTCHA's onderscheiden. Dit soort CAPTCHA toont willekeurige objecten die moeten geidentificeerd worden door de gebruiker. Deze objecten kunnen in alle mogelijk formaten getoond worden. Het gemakkelijkste voorbeeld is een foto van een kat, waarna de gebruiker het woord "kat" moet intypen. Een meer geavanceerd voorbeeld van een niet tekstuele CAPTCHA is de 3D CAPTCHA (figuur 2.3). De gebruiker kan dan gevraagd worden om de letters te geven van bepaalde objecten in de 3D scene. De vraag "geef de letters van het hoofd van de lopende man en de vaas" zou dan resulteren in C en K. Wanneer er echter gesproken wordt over CAPTCHA's in de strikte betekenis, geldt onderstaande CAPTCHA niet. Een CAPTCHA moet volledig geautomatiseerd kunnen gegenereerd worden. Een 3D scene moet op voorhand manueel gelabeld worden. Er kunnen natuurlijk wel verschillende afbeeldingen gegenereerd worden van dezelfde scene, waardoor het automatische aspect weer terugkomt.



Figuur 2.3: Een voorbeeld van een 3D CAPTCHA

Een ander soort CAPTCHA is de auditieve CAPTCHA. Een voorbeeld hiervan is een

geluidsfragment waarin een reeks karakters wordt voorgelezen die door de gebruiker moet herkend worden. Ook hier is het moeilijk voor een computeralgoritme om te verstaan wat de CAPTCHA zegt.

Deze thesis zal zich uitsluitend beperken tot het onderzoeken van tekstuele visuele CAPTCHA's.

2.1 Toepassingen

Er zijn vele praktische toepassingen van CAPTCHA's. Enkele voorbeelden hiervan zijn:

- Het voorkomen van spam in blogs: vaak worden bots gebruikt om reclame te maken op publieke blogs. Door een captcha te gebruiken kan alleen een menselijke gebruiker een bericht achterlaten op de blog.
- Het beschermen van website registraties: verschillende bedrijven (waaronder Yahoo!, Microsoft, Google, ...) bieden gratis diensten aan waarvoor men zich online kan inschrijven. Deze diensten hadden voornamelijk veel last van bots die duizenden keren registreerden voor een bepaalde service in enkele minuten. De oplossing hiervoor is het gebruik van een CAPTCHA. Tegenwoordig worden bijna alle online diensten beschermd door een CAPTCHA.
- Het beschermen van emailadressen tegen spambots: vaak doorzoeken spambots het internet om zo emailadressen te bemachtigen. Door de zichtbaarheid van emailadressen te beschermen met een CAPTHCA kan dit vermeden worden.
- Online polls (stemmingen): bots kunnen zonder het gebruik van een CAPTCHA duidenden stemmen uitbrengen. Dit maakt het resultaat van een stemming compleet nutteloos.
- Het voorkomen van aanvallen op paswoorden: In plaats van een account te blokkeren na een bepaald aantal foute pogingen om een wachtwoord in te geven, kan men simpelweg een CAPTCHA laten invullen.

De mogelijkheden van een CAPTCHA zijn natuurlijk veel uitgebreider als de voorbeelden die hier gegeven zijn. De hoofdreden om een CAPTCHA te gebruiken is wel duidelijk beveiliging en veiligheid.

2.2 Richtlijnen

Er zijn vele CAPTCHA-implementaties, sommige beter dan anderen. Er zijn twee criteria waaraan een goede CAPTCHA moet voldoen. Vanzelfsprekend moet de CAPTCHA moeilijk te lezen zijn door een algoritme. En ten tweede mag een CAPTCHA niet teveel inspanning vragen van de gebruiker. Als een CAPTCHA te ingewikkeld wordt, kan het zijn dat gebruikers de tekst ook niet meer kunnen lezen. Dit moet ten aller tijden vermeden worden. Om aan deze eisen te voldoen, kunnen enkele richtlijnen gevolgd worden:

- Beschikbaarheid: CAPTCHA's moeten beschikbaar zijn voor alle mensen. Als er een visuele CAPTCHA gecreëerd wordt, moet er ook een alternatief aanwezig zijn voor mensen die niet goed zien. Een voorbeeld hiervan is een auditieve CAPTCHA.
- Afbeelding beveiliging: Sommige CAPTCHA's gebruiken slechts kleine aanpassingen aan tekst zodat een simpel algoritme deze al kan kraken. Letter zouden bijvoorbeeld vervormd moeten worden dat ze niet meer leesbaar zijn voor een algoritme.
- Willekeurig: CAPTCHA's moeten voldoende willekeurig gegenereerd worden om het moeilijk te maken simpele aanvallen door computer algoritmen te overleven. Dit wil zeggen dat niet altijd dezelfde woorden of karakterreeksen gebruikt moeten worden.
- Script beveiliging: Wanneer een veilige CAPTCHA gecreëerd is, moet men er voor zorgen dat er geen simpele manier is rond de CAPTCHA zonder deze op te lossen. Om een CAPTCHA te maken is het noodzakelijk dat er op script niveau geen manier is rond de CAPTCHA. Dit is echter een algemeen internet beveiligingsprobeem. Twee veel voorkomende voorbeelden van deze onveiligheden zijn de volgende:
  - Systemen die het antwoord van de CAPTCHA in ongeëncrypteerde tekst doorsturen als deel van een website formulier.
  - Systemen die een vast woordenboek gebruiken voor hun CAPTCHA. Hierdoor is het mogelijk dat oplossingen meerdere keren gebruikt kunnen worden. De meeste gratis CAPTCHA scripts op het Web zijn kwetsbaar voor dit soort aanvallen.
- Veiligheid zelfs na veelvuldig gebruik: Een heel eenvoudige CAPTCHA kan heel goed werken indien de CAPTCHA maar op weinig plaatsen gebruikt wordt. Stel de vraag "hoeveel is 1 + 1?" als CAPTCHA. Er zou perfect een algoritme geschreven kunnen

worden dat de vraag analyseert en 2 antwoordt. De meeste bots kunnen dit echter niet omdat er verondersteld wordt dat deze CAPTCHA's niet gebruikt worden omdat ze te simpel zijn. Moest een populaire site als bijvoorbeeld Google deze CAPTCHA gebruiken, zouden alle bots dit kunnen kraken. Een goede CAPTCHA moet ook veilig blijven wanneer ze vaak gebruikt wordt.

- Zelf een CAPTCHA maken? In het algemeen is het geen goed idee om zelf een CAPTCHA script te schrijven. Er zijn heel veel veiligheids problemen waar men misschien niet aan denkt. Het is altijd beter om een doorwinterde CAPTCHA te gebruiken die zijn dienst al bewezen heeft.

Het doel van deze thesis is het onderzoeken van de afbeelding beveiliging.

Hoofdstuk 3

Voorgaand Werk

Het breken van een CAPTCHA komt neer op het succesvol uitvoeren van een aantal taken. De belangrijkste taak is het herkennen van karakters in de afbeelding. Aangezien CAPTCHA's vaak gebruik maken van vervormingsalgoritmen, is het noodzakelijk dat de karakterherkenning robuust genoeg is zodat zelfs vervormde karakters onderscheiden kunnen worden. Een tweede taak die uitgevoerd moet worden, is de mogelijke posities bepalen waar eventueel karakters kunnen voorkomen. Sommige herkenningsalgoritmen, bijvoorbeeld Distance Matching (zie hoofdstuk 6), kunnen zelf bepalen waar de karakters zich bevinden. Dit is echter niet altijd het geval. Andere algoritmen hangen af van een goede segmentatie van de CAPTCHA. Dit wil zeggen dat de afbeelding moeten opgedeeld worden in stukken of segmenten. Deze segmenten bevatten typisch slechts één karakter. Hierna kunnen de segmenten geanalyseerd, en de karakters herkend worden.

Er zijn reeds verschillende pogingen gedaan om bestaande CAPTCHA's te breken. Er werd bijvoorbeeld reeds getracht[2] de EZ-Gimpy CAPTCHA te breken. EZ-Gimpy is een variant van de Gimpy CAPTCHA. Bij Gimpy moeten er drie woorden uit de CAPTCHA gehaald worden. EZ-Gimpy werkt met slechts één woord. Deze pogingen om deze CAPTCHA te breken zijn over het algemeen redelijk succesvol. Het slaagpercentage van het breken van de EZ-Gimpy CAPTCHA is 83%. Voor de Gimpy CAPTCHA lag het lager. Aangezien er bij de Gimpy CAPTCHA drie woorden moeten gezocht worden in de afbeelding, is dit normaal. Het slaagpercentage is echter nog steeds 33%. Op het eerste zicht lijkt die weinig. Het wil echter wel zeggen dat een computeralgoritme een op de drie keer de CAPTCHA successvol kan breken. Een spambot kan vaak duizenden malen per minuut

proberen om de CAPTCHA te breken. De Gimpy CAPTCHA zou dus op enkele seconden gebroken zijn.

Het nadeel is echter dat de CAPTCHA's die gebruik worden in voorgaande studies vaak oud en achterhaald zijn. De Gimpy CAPTCHA is, zoals eerder vermeld, de eerste CAPTCHA ooit gemaakt. Een grote zwakte van Gimpy is dat de CAPTCHA gebruik maakt van een vaste database van 561 woorden en altijd hetzelfde lettertype hanteert. Dit is zelfs door brute force gemakkelijk te breken. Wanneer een spambot slechts één woord kent uit de database, kan het de CAPTCHA statistisch gezien breken in 561 kansen. Stel dat er elke seconde een poging wordt gedaan, dan is de CAPTCHA gebroken binnen 10 minuten.

De techniek die gebruikt wordt bij het kraken van EZ-Gimpy is Shape Context Matching (zie hoofdstuk 5). Doordat de CAPTCHA een vaste woorden database heeft, kan de techniek aangevuld worden door het combineren van letters tot woorden. Op deze manier kunnen bijna alle foutieve karakters gecorrigeerd worden. Voor elke karaktercombinatie wordt een score opgesteld. De oplossing met de hoogste score is de oplossing van de CAPTCHA.

In deze thesis wordt de nadruk gelegd op het individueel onderscheiden van karakters in een CAPTCHA. Er wordt dus geen rekening gehouden met de volgorde van de karakters, en of er eventueel woorden gevormd kunnen worden. Dit wordt gedaan omdat de meeste hedendaagse CAPTCHA's combinaties zijn van willekeurige letters en cijfers.

Bij het breken van CAPTCHA's worden verschillende herkennigsalgoritmen gebruikt, ook wel *matching* algoritmen genoemd. Hier is reeds veel onderzoek naar gedaan. Sinds lange tijd is het mogelijk om ingescande documenten te analyseren en om te zetten naar digitale tekst. Dit gebeurt door Optical Character Recognition[3][4] (OCR) algoritmen. De standaard OCR technieken zijn echter niet goed genoeg om een CAPTCHA te analyseren. Wanneer teveel vervorming en ruis voorkomt in het document, geven de meeste OCR technieken een slecht resultaat. OCR algoritmen maken gebruik van *feature extraction* om karakters te onderscheiden. Feature extraction werkt echter alleen goed wanneer er geen ruis of andere hinderingen voorkomen. Er worden in deze thesis twee technieken besproken die een robuustere oplossing geven voor het object herkennings probleem. Er wordt getracht het probleem van ruis te overwinnen om betere feature extraction te bekomen. Deze twee veel gebruikte technieken zijn ten eerste de voordien aangehaalde Shape Context Matching. Als tweede techniek wordt Distance Transform Matching besproken.

Met Shape Context Matching (zie hoofdstuk 5) probeert aan de hand van *shape contexts* te beschrijven hoe een vorm eruit ziet. De shape context beschrijving van een vorm is zeer gedetailleerd, en kan weerstaan aan lichte (en soms zelfs zware) vervormingen. Met behulp van de shape contexts kunnen vormen gemakkelijk vergeleken worden met referentievormen. Zo kan een efficiënt herkenningsalgoritme bekomen worden.

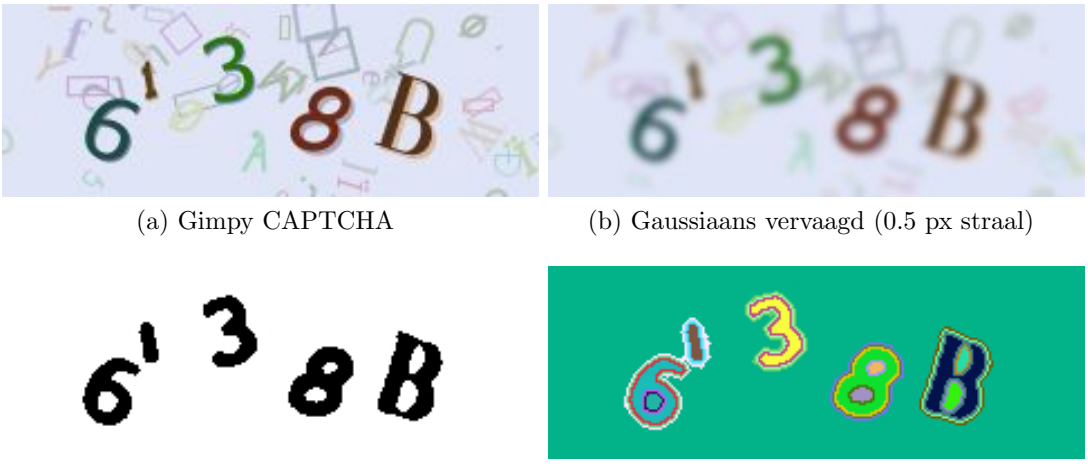
Distance Transform Matching (zie hoofdstuk 6) gaat anders te werk. Ook hierbij wordt gebruik gemaakt van referentievormen. Deze vormen worden door middel van een distance map gezocht in een afbeelding. Hoewel de Distance Transform niet goed bestand is tegen vervormingen, heeft het een ander groot voordeel. Dit algoritme is in staat om in een afbeelding vol ruis nog de referentievormen te vinden. Indien er dus een groot aantal referentievormen aanwezig zijn, is dit een zeer efficiënte aanpak om CAPTCHA's te analyseren.

In deze thesis zullen de twee bovenstaande algoritmen uitgebreid aan bod komen. Nadien wordt er getracht de technieken te combineren zodanig dat de voordelen van beide technieken opgenomen worden in de nieuwe. Distance Transform Matching is zeer geschikt voor het vinden van mogelijke kandidaat lokaties van karakters. Aangezien Shape Context Matching afhangt van een goede segmentatie van de CAPTCHA, kan Distance Transform Matching hier een grote hulp bieden. De kandidaat lokaties kunnen dan op hun beurt door Shape Context Matching opnieuw worden onderzocht. Dit maakt het op voorhand segmenteren van een afbeelding overbodig. De combinatie van de twee technieken is geen voorgaand werk, en wordt als nieuwe methode onderzocht en uitgewerkt.

Hoofdstuk 4

Naïef CAPTCHA's breken

Aangezien CAPTCHA's in honderden verschillende vormen en maten beschikbaar zijn, is het bijna onmogelijk om een algemene methode te ontwikkelen om alle CAPTCHA's te kraken. Er zijn echter wel gelijkenissen tussen CAPTCHA's. Daarom kunnen enkele algemene richtlijnen gevolgd worden om het proces te vergemakkelijken. Meestal zijn simpele beeldverwerkingstaken het effectiefste om een goede initiële segmentatie te bekomen. Er zal dus ook veelvuldig van deze operaties gebruik gemaakt worden in deze thesis.



(a) Gimpy CAPTCHA (b) Gaussiaans vervaagd (0.5 px straal)

(c) drempelwaarde (waarde 195) (d) segmentatie

Figuur 4.1: Het effect van beeldverwerkingsoperaties en segmentatie

Algemene ruis in een afbeelding is bijvoorbeeld heel snel te elimeneren door Gaussiaanse vervaging toe te passen, gevolgd door een drempelwaarde filter. Afbeelding 4.1 laat de

CAPTCHA zien die gebruik wordt door phpBB [1]. Deze CAPTCHA is een vrij simpel, maar wordt toch massaal gebruikt op het internet. Zoals te zien is de achterground ruis volledig verdwenen in 4.1c. Hierna is een segmentatie algoritme[5] toegepast. Het resultaat is zichtbaar in figuur 4.1d. Het algoritme kleurt alle segmenten in een willekeurige kleur. Ook de randen van elk segment worden in een andere kleur ingekleurd. Op deze manier is duidelijk te zien welke segmenten er gecreëerd zijn. Het is duidelijk te zien dat de letters goed gesegmenteerd worden. Na vormherkenning toe te passen, kan hier gemakkelijk het gewenste resultaat bereikt worden. Het is dus in dit geval niet moeilijk om de desbetreffende CAPTCHA te breken.

Eender welk ander segmentatie algoritme kan hier gebruikt worden. Aangezien er een preprocessing stap wordt uitgevoerd om de achtergrond ruis te verwijderen, moet het segmentatie algoritme enkel nog de verschillende letters van elkaar scheiden.

(a) Google CAPTCHA

(b) Gaussiaans vervaagd (0.5 px straal)

(c) drempelwaarde (waarde 205)

(d) segmentatie

Figuur 4.2: Dezelfde operaties van figuur 4.1 op de Google CAPTCHA

Om aan te tonen dat het breken van een CAPTCHA niet altijd even voordehandliggend is, worden dezelfde beeldverwerkingstaken toegepast op de CAPTCHA van Google, zichtbaar in figuur 4.2. Hier is duidelijk te zien dat de Gaussiaanse vervaging een negatieve invloed heeft op het resultaat.

Gaussiaanse verganging is maar een van de vele vervagingstechnieken. Technieken zoals mediaanse, bilaterale, ... vervaging kunnen natuurlijk ook gebruikt worden. Het hangt hier weer af van het sooft CAPTCHA welke techniek het beste resultaat geeft. Een zeer

[1] phpBB is een opensource forum: http://www.phpbb.com/

effectieve manier om selectieve vervanging te doen van de achtergrondruis, is de SGBNR vervagingstechniek[2]. Deze techniek probeert door middel van een low-pass filter te detecteren waar de details zitten in de afbeelding. Hierna wordt de rest vervaagd, waardoor de letters zichtbaar blijven.

(a) Gimpy CAPTCHA (b) SGBNR

Figuur 4.3: Het effect van SGBNR in plaats van gewone Gaussiaanse vervaging

Dit toont dus aan dat deze procedures verschillen van CAPTCHA tot CAPTCHA. Soms is het bijvoorbeeld niet nodig om een vervaging te doen, en kan dit zelfs slechtere resultaten opleveren. Bij het breken van een CAPTCHA is het zeer belangrijk om op voorhand te weten hoe een CAPTCHA er ongeveer gaat uitzien. Er kan dan geanticipeerd worden welke beginoperaties het beste resultaat geven. Indien het niet mogelijk is om de ruis volledig te verwijderen, zullen de herkenningsalgoritmen moeten aangepast worden. Dit kan bijvoorbeeld door andere parameters mee te geven. Het is dus duidelijk dat het bijna onmogelijk is om een universele techniek te ontwikkelen om een willekeurige CAPTCHA te breken.

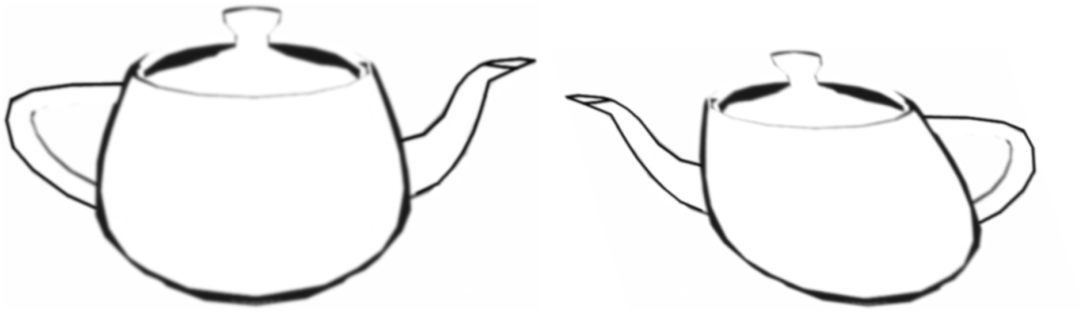
[2]SGBNR: Selective Gaussian Blur Noise Reduction

Hoofdstuk 5

Shape Context Matching

5.1 Historiek

De wiskundige definitie van gelijkheid verschilt van de intuïtieve definitie. Wiskundigen beschrijven vormen typisch als gelijk indien ze slechts van elkaar verschillen door een bepaald aantal transformaties. Deze transformaties kunnen bijvoorbeeld schaleringen, rotaties, spiegelingen, ... zijn. In figuur 5.1 zijn de twee vormen dus wiskundig hetzelfde, oftwel gelijk, omdat ze slechts verschillen door drie transformaties.



(a) Teapot (b) Teapot met schalering, spiegeling en shear

Figuur 5.1: Dezelfde afbeelding op drie transformaties na

Deze definitie is niet voldoende in de context van visuele analyse en computer vision. Dit vertelt ons alleen maar als twee vormen exact gelijk zijn. In het herkennen van objecten zullen vormen nooit perfect overeenkomen. Er zijn altijd kleine verschillen terwijl de vormen toch hetzelfde voorstellen. Dit is voornamelijk het geval bij karakterherkenning. Er

zijn honderden verschillende lettertypes die toch dezelfde vormen bevatten, bijvoorbeeld het alfabet. Er is dus meer nodig dan alleen transformaties om de gelijkenissen van vormen te berekenen.

Uitgebreide onderzoeken[6][7] tonen aan dat er twee grote aanpakken zijn in verband met vorm gelijkenis. De eerste is *feature*-based matching. Hier wordt voornamelijk gelet op de ruimtelijke plaatsing van belangrijke kenmerken van de vorm. Veel onderzoek naar vorm gelijkenis is gedaan met behulp van silhouette afbeeldingen. Silhouettes hebben geen gaten of andere afwijkingen. De omtrek van een silhouette kan worden voorgesteld door een gesloten curve die geparametriseerd kan worden door de booglengte. Silhouettes zijn echter gelimiteerd als beschrijving voor vormen. Interne eigenschappen van de vorm worden genegeerd en silhouette afbeeldingen zijn moeilijk te bekomen uit echte afbeeldingen. Betere methoden beschouwen een vorm als een set van punten in een 2D afbeelding. Deze punten kunnen veel gemakkelijker bekomen worden uit een bestaande afbeelding door bijvoorbeeld een edge-detection algoritme te gebruiken.

De andere aanpak is *brightness*-based (of appearance-based) matching. Deze aanpak maakt gebruik van de kleur en/of helderheid van een bepaalde pixel om vormen te herkennen en vergelijken. In tegenstelling tot de feature-based matching methoden, maakt deze methode rechtstreeks gebruik van de luminantiewaarden van de pixels uit de vorm.

Shape context matching valt in de feature-based matching categorie.

5.2 Introductie

Shape contexts[8] zijn een mogelijke benadering tot het meten van gelijkheid tussen verschillende vormen. Men kan dit uitbreiden naar het herkennen van objecten en vormen. Wanneer vormen boven een bepaalde gelijkheidsgraad zitten, worden ze als dezelfde vorm beschouwd.

Beschouw de twee handgeschreven cijfers in figuur 5.2. Wanneer deze cijfers pixel voor pixel vergeleken worden met elkaar, zijn ze grotendeels verschillend. Toch zijn het allebei afbeeldingen van het cijfer 5. Een menselijk oog kan gemakkelijk de gelijkenis in de vorm van de cijfers onderscheiden. In CAPTHCA's zullen de vormen ook vaak verschillend zijn. Het eenvoudigste verschil is het gebruik van verschillende soorten lettertypes. Er wordt dan

nog niet gesproken over de vervormingen die kunnen toegepast worden op de individuele karakters.

Figuur 5.2: Een voorbeeld van twee handgeschreven vijven

Het doel van shape contexts is deze notie van vorm te kunnen definieren en verschillende vormen te kunnen vergelijken met elkaar op een objectieve manier. In essentie bestaat het shape context matching proces uit de volgende stappen:

1. het opstellen van de shape contexts
2. het vaststellen van overeenkomsten in de verschillende vormen
3. deze overeenkomsten gebruiken om een transformatie te vinden van de ene vorm naar de andere
4. de afstand berekenen tussen twee vormen als de som van de foutmarges tussen de verschillende punten
5. de vormen met een afstand kleiner dan een bepaalde drempelwaarde zijn gelijke vormen

Het is niet mogelijk om de volledige vorm op te nemen in het berekenen van shape contexts. Om overeenkomsten te vinden tussen vormen, is het noodzakelijk dat er een concreet aantal punten gekozen wordt uit deze vormen. Typisch zijn dit een honderdtal pixels die zich aan de rand van de vorm bevinden. Om de rand van de vorm te berekenen wordt een standaard edge-detection algoritme gebruikt (vb: Canny Edge Detection). De gekozen punten (samples) hebben geen speciale eigenschappen met betrekking tot de vorm. Het zijn willekeurige samples. Hoe meer samples er gekozen worden, hoe beter de beschrijving wordt van de onderliggend vorm. Aangezien de samples willekeurig gekozen worden, kan het zijn dat er bij onvoldoende samples een groot deel van de vorm niet beschouwd wordt. Bij een groot aantal samples, gaan de samples wel overal op de rand van de vorm voorkomen. Dit geeft een veel beter beschrijving van de vorm.

Vervolgens wordt aan elk van de samples een *shape context* gekoppeld. Deze shape context bevat informatie over de ligging van andere samples ten opzichte van de huidige. Wanneer twee vormen met elkaar vergeleken moeten worden, probeert men voor elke sample in de ene vorm de beste overeenkomst te zoeken in de andere door de shape contexts van de betreffende samples met elkaar te vergelijken. Bij het vergelijken van de shape contexts met elkaar, komt een foutmarge voor. Door alle foutmarges van alle samples uit de vormem bij elkaar op te tellen, bekomen we een bepaalde afstand. Deze afstand geeft aan in hoeverre de vormen met elkaar verschillen.

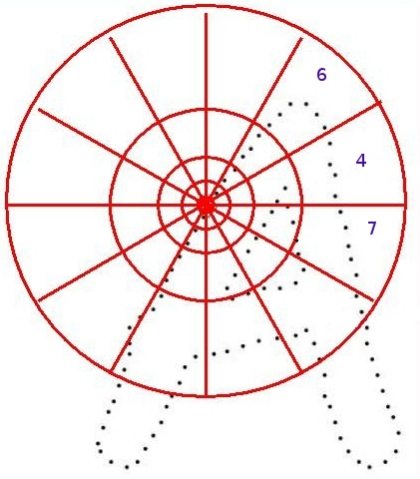
5.3 Opstellen van Shape Contexts

In de shape context methode worden objecten beshouwd als een set van (mogelijk oneindig aantal) punten. Er wordt aangenomen dat de vorm van een object kan voorgesteld worden door een eindig aantal punten uit het object. Meerbepaald, een vorm wordt voorgesteld door een bepaald aantal punten dat gesampled wordt uit de interne en externe contouren (edges) van het object. Praktisch kunnen deze samples bekomen worden door de pixels te gebruiken die een edge-detection algoritme geproduceerd heeft. Deze punten komen meestal niet overeen met belangrijke kenmerken van de vorm. De vorm wordt bij voorkeur gesampled met gelijke afstand tussen de punten. Enkele sample voorbeelden zijn zichtbaar in figuur 5.3.

Neem nu een sample p en n het aantal samples in de vorm. Er bestaan dan $n-1$ vectoren van p naar elke andere sample. Deze set van vectoren beschrijft de onderliggende vorm relatief tot een referentiepunt, in dit geval sample p. Aangezien n typisch zeer groot wordt, is het duidelijk dat deze set een gedetailleerde beschrijving is van de vorm. De volledige set is echter veel te gedetailleerd om te dienen als een vormbeschrijving. Het is niet praktisch om alle vectoren van alle samples van een vorm te vergelijken met deze van een andere vorm. In plaats hiervan wordt een histogram opgesteld voor elke sample waarin informatie zit over de naburige samples. Wanneer dit hier toegepast wordt, bekomt men een robuus-

Figuur 5.3: Verschillende lettertypen met verschillende sample dichtheden

te en compacte methode om vormen te be-
schrijven. Ondanks de compactheid blijft deze beschrijving er in slagen vormen correct en nauwkeurig te identificeren. Elke sample p_i krijgt een histogram h_i toegekend. Dit histogram wordt dan uiteindelijk de *shape context* van sample p_i genoemd. De opbouw van de shape context gaat als volgt, met k het aantal bins:

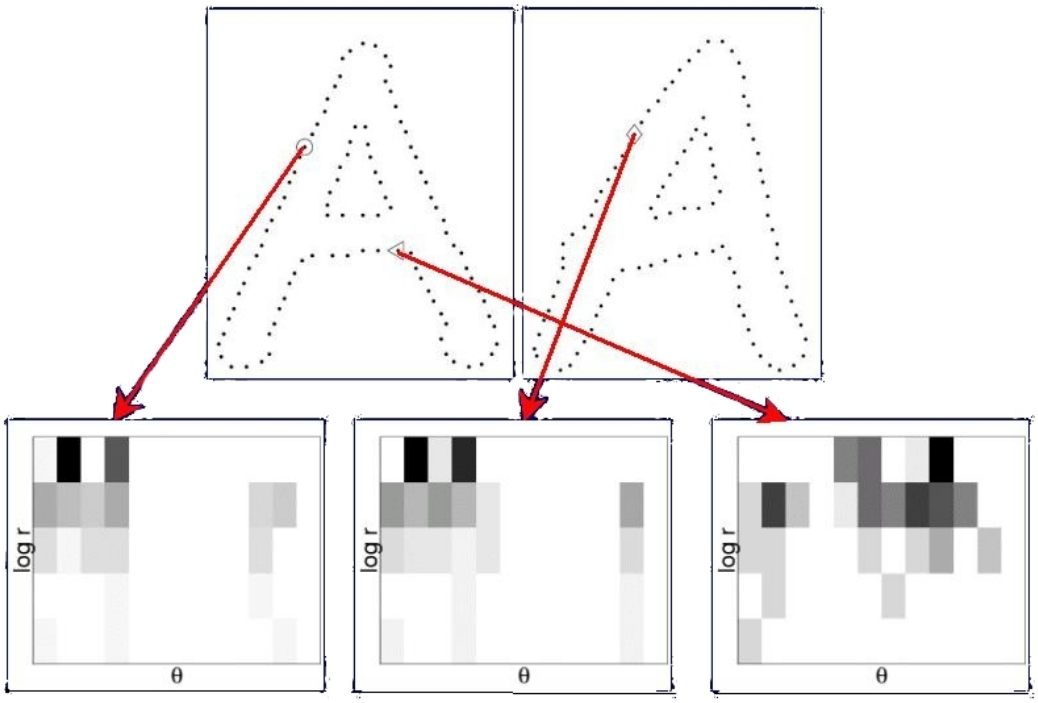


Figuur 5.4: Voorbeeld van bin verdeling van samples

$$h_i(k) = \#\{q \neq p_i : (q - p_i) \in bin(k)\} \tag{5.1}$$

De bins worden uniform verdeeld in de logaritmisch-polaire ruimte. Dit zorgt ervoor dat de samples die dichtbij liggen meer doorwegen als de verder gelegen samples. Figuur 5.4 laat zien hoe de bins worden verdeeld en berekend voor de sample aangeduid door de ruit uit figuur 5.5. Wanneer deze nu weergegeven worden door een histogram, bekomt met het resultaat eronder. Hoe meer samples er in een bepaalde bin zitten, hoe donkerker het histogram. Deze histogrammen zijn dus de shape contexts.

De shape contexts uit figuur 5.5 aangeduid met de cirkel en de ruit zijn bijna identiek. Dit komt omdat de verdeling van de samples rond de cirkel en de ruit ongeveer hetzelfde is. Dit is typerend voor gelijke vormen. Hoewel de vormen wiskundig verschillend zijn, zorgen de shape contexts ervoor dat het toch mogelijk is om de gelijkheid van de vormen aan te tonen.



Figuur 5.5: Voorbeelden van shape contexts.

5.4 Shape Contexts vergelijken

5.4.1 Overeenkomsten zoeken

Zoals eerder vermeld, moeten er eerst overeenkomsten gezocht worden tussen de verschillende samples op de verschillende vomen. Aangezien een shape context een distributie van samples voorstelt, is het aangewezen[8] om een chi-kwadraat test uit voeren om de kost te berekenen tussen twee verschillende punten. De uiteindelijke kost wordt bepaald door de globale som van alle kosten te nemen en deze te minimaliseren.

Beschouw dus eerst punt p_i uit een eerste vorm en punt q_i uit een tweede. $C_{ij} = C(p_i, q_i)$ is de kost berekend voor deze twee punten.

$$C_{ij} = \frac{1}{2} \sum_{k=1}^{K} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)} \tag{5.2}$$

Hierbij zijn $h_i(k)$ en $h_j(k)$ de K-de bin van het histogram van punt p_i en q_i respectievelijk. Het is mogelijk om aan deze kost nog een extra term toe te voegen. Deze term wordt de *appearance similarity* genoemd. De vorm van deze kost kan verschillen van applicatie tot applicatie. Het is bijvoorbeeld mogelijk om de richting van de raaklijn aan de vorm mee te geven. Indien er gewerkt wordt met grijswaarden in plaats van binaire afbeeldingen, is het mogelijk om deze grijswaarden rond punt p_i en q_i te analyseren en de standaardafwijking van de grijswaarden mee te geven als appearance similarity term. Het kiezen van de vorm van deze term hangt af van de noden van de applicatie (vb: robuustheid, invariantie voor speciale aspecten).

Gegeven is nu een set van kosten C_{ij} tussen alle puntenparen p_i en q_i uit de eerste en tweede vorm respectievelijk. De totale kost van het matchen wordt dan aangeduid als volgt:

$$H(\pi) = \sum_i C(p_i, q_{\pi(i)}) \tag{5.3}$$

waarbij π een permutatie is van alle punten p_i en q_i. Om een optimale match te vinden, wordt de Hungarian bipartite graph matching[9] methode toegepast. Deze methode kan opgelost worden in $O(N^3)$. Om uitschieters op een robuuste manier af te handelen worden er dummy-nodes toegevoegd aan elke puntenset met kost ϵ_d. Deze dummy kost wordt toegewezen aan alle punten die geen match vinden met een kost kleiner als ϵ_d. ϵ_d kan dus gezien worden als een soort drempelwaarde die geïntroduceert wordt om uitschieters tegen te gaan. Een andere toepassing van deze dummy waarden is het gelijk maken van het aantal samples uit de verschillende vormen. Wanneer een vorm meer samples heeft als de andere, worden de samples opgevuld met dummy waarden. Deze situatie kan voorkomen wanneer de afbeelding niet groot genoeg is en er niet de mogelijkheid is om een groot aantal samples te nemen uit een vorm.

5.4.2 Invariantie en robuustheid

Wanneer vormen met elkaar vergeleken worden, moeten er aan een twee grote criteria voldaan worden:

- invariantie bij schalering, translatie en rotatie (of zelfs een volledige groep van affiene transformaties)

- robuustheid bij kleine geometrische veranderingen, occlusies of uitschieters

Invariantie bij translatie is vanzelfsprekend aangezien alle metingen gedaan worden relatief tot een sample op de vorm. Ook invariantie bij schaleren is gemakkelijk te verkrijgen. Alle radiale afstanden worden genormaliseerd met behulp van de gemiddelde afstand van alle sample paren tussen de verschillende vormen. Shape contexts bevatten veel extra informatie. Hierdoor zijn ze niet erg gevoelig voor kleine storingen in de vorm. Ook wordt aangenomen dat kleine geometrische veranderingen aanvaard worden.

Het is mogelijk om shape contexts volledig invariant te maken voor rotaties. Dit is echter niet van toepassing aangezien het onderscheiden van sommige symbolen niet meer mogelijk is (vb: 6 of 9). Door de rotaties niet mee te rekenen, doet zich wel een nieuw probleem voor in het herkennen van karakters. In CAPTCHA's worden karakters namelijk wel in kleine mate geroteerd. Dit probleem kan echter opgelost worden door een grote referentiedatabase te gebruiken. In deze database zitten typisch alle mogelijke variaties van een bepaald karakters, met rotaties inbegrepen.

5.4.3 Transformatie modelleren

Er is nu een eindige set van overeenkomsten tussen de punten op de verschillende vormen gevonden. Men kan nu doorgaan met het schatten van een tranformatie $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ die kan gebruikt worden om willekeurige punten te mappen van de ene vorm op de andere.

T moet gekozen worden uit een familie van affiene transformaties. Het standaardmodel van een affiene transformatie is:

$$T(x) = Ax + o \tag{5.4}$$

Hierbij is A een matrix is en o de begin offset. Hiermee kunnen alle mogelijke toegelaten affiene transformaties gemodelleerd worden. Er wordt gebruik gemaakt van de kleinste-kwadratenmethode om T te schatten. De oplossing $\hat{T} = (\hat{A}, \hat{o})$ wordt bekomen door:

$$\hat{o} = \frac{1}{n} \sum_{i=1}^{n} (p_i - q_{\pi(i)}) \tag{5.5}$$

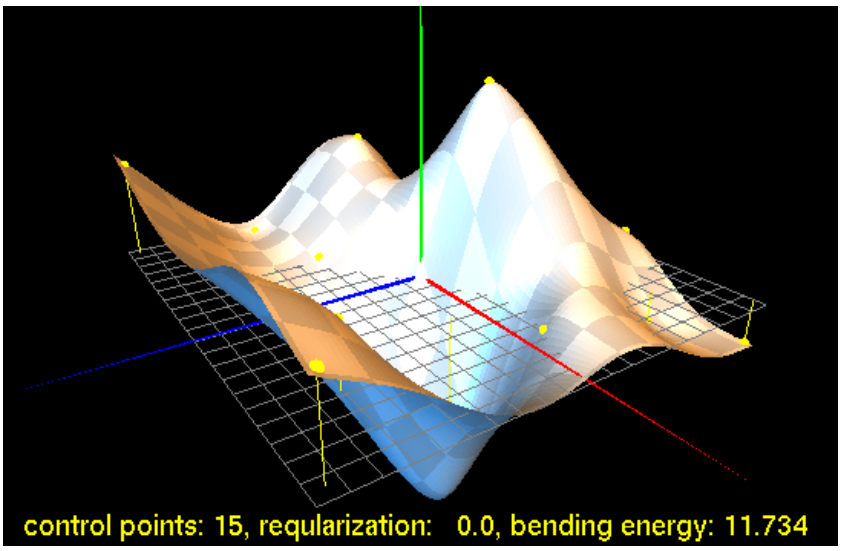
$$\hat{A} = (Q^+ P)^t \tag{5.6}$$

waarbij P en Q de homogene coordinaten bevatten van de verschillende vormen als volgt:

$$P = \begin{pmatrix} 1 & p_{11} & \cdots & p_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & p_{n1} & \cdots & p_{nm} \end{pmatrix} \text{ en } Q = \begin{pmatrix} 1 & q_{11} & \cdots & q_{1r} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & q_{s1} & \cdots & q_{sr} \end{pmatrix} \tag{5.7}$$

Q^+ stelt het pseudo-inverse van Q voor.

Er wordt hier voornamelijk gebruik gemaakt van de *Thin Plate Spline* (TPS) methode[10][11]. Deze methode maakt het mogelijk om op een flexibele manier aan de hand van controlepunten een transformatie te modelleren. De methode kan vergeleken worden met een ijzeren plaat die gebogen wordt zodat deze door alle gegeven controlepunten komt. In figuur 5.4.3 wordt de methode visueel voorgesteld. Om de methode beter te begrijpen moet



Figuur 5.6: Een 3D oppervlak dat gebogen wordt door de controlepunten (geel)

men eerst de achtergrond van het algoritme kennen. Men begint bij het 1 dimensionaal interpolatie probleem. Laat de controlepunten v_i de functiewaarden zijn op de overeenkomstige punten $p_i = (x_i, y_i)$ in een vlak, met $i = 1, 2, \ldots, n$. In het bijzonder, v_i wordt om de beurt gelijkgesteld aan x_i' en y_i' om een continue transformatie te bekomen voor elke coordinaat. Er wordt aangenomen dat alle (x_i, y_i) coordinaten verschillend en niet collineair zijn. De TPS interpolant $f(x, y)$ minimaliseert de "buig energie"

$$T_f = \int\int_{\mathbb{R}^2} (\frac{\partial^2 f}{\partial x^2})^2 + 2(\frac{\partial^2 f}{\partial x \partial y})^2 + (\frac{\partial^2 f}{\partial y^2})^2 dxdy \tag{5.8}$$

en heeft als vorm:

$$f(x,y) = a_1 + a_x x + a_y y + \sum_{i=1}^{n} w_i U(||(x_i, y_i) - (x, y)||) \tag{5.9}$$

waarbij de funtie $U(r)$ gedefiniëerd wordt door $U(r) = r^2 \log r^2$ en $U(0) = 0$. Om ervoor te zorgen dat $f(x, y)$ kwadratisch integreerbare tweede afgeleiden heeft, moet

$$\sum_{i=1}^{n} w_i = 0 \text{ en } \sum_{i=1}^{n} w_i x_i = \sum_{i=1}^{n} w_i y_i = 0 \tag{5.10}$$

Samen met de interpolatie condities $f(x_i, y_i) = v_i$ houdt dit het volgende lineair systeem in voor de TPS coefficiënten:

$$\left(\begin{array}{c|c} K & P \\ \hline P^T & 0 \end{array}\right) \binom{w}{a} = \binom{v}{0} \tag{5.11}$$

waarbij $K_{ij} = U(||(x_i, y_i) - (x_j, y_j)||)$, de $i$de rij van P gelijk is aan $(1, x_i, y_i)$, w en v kolom vectoren zijn gevormd door w_i en v_i respectievelijk, en a een kolom vector is met als elementen a_1, a_x, a_y. Deze $(n + 3) \times (n + 3)$ matrix noteren we als L. Men kan aantonen dat L inverterbaar is. Als men de links bovenste blok van L^{-1} als A noteert, kan men aantonen dat

$$I_f \alpha v^T A v = w^T K w \tag{5.12}$$

Omdat er ruis kan aanwezig zijn in de controlepunten kan het gewenst zijn om de interpolatie te benaderen in plaats van de exacte oplossing te berekenen. Dit kan bereikt worden door de volgende vergelijking te minimaliseren:

$$H[f] = \sum_{i=1}^{n} (v_i - f(x_i, y_i))^2 + \lambda I_f \tag{5.13}$$

De *regularizatie parameter* λ is een positief getal dat de gladheid, of afvlakking (smoothing) van de functie bepaalt. Wanneer $\lambda = 0$ wordt er terug overgegaan op exacte interpolatie.

Door de TPS methode nu toe te passen op de geschatte overeenkomsten tussen verschillende afbeeldingen, kan een transformatie geschat worden. Deze initiële schatting is vaak onjuist aangezien de geschatte overeenkomsten vaak fouten bevatten. Door het proces te itereren, wordt dit verholpen. De geschatte transformatie wordt toegepast, de punten wor-

den opnieuw gesampled en er worden opnieuw overeenkomsten gezocht. Typisch wordt een vast aantal iteraties gebruikt.

5.4.4 Shape Distance

Met dit alles in het achterhoofd, kan men beginnen aan object herkenning. Object herkenning met behulp van shape context valt onder de categorie van *prototype*-based herkenning. Dit wil zeggen dat vormen vergeleken worden met gekende referentie vormen. In de context van deze thesis wil dit zeggen dat er een dataset van karakters beschikbaar is waarmee de vormen moeten vergeleken worden.

Er moet dus een manier gevonden worden om te meten welke prototype, of referentieafbeelding het beste overeenkomt met de te zoeken vorm. Hiervoor wordt de *shape distance* geïntroduceert. De shape distance is een afstandswaarde tussen de verschillende vormen. Door de kleinste shape distance te kiezen, kan de beste overeenkomst gekozen worden. De shape distance wordt als volgt berekend:

$$D_{sc}(P,Q) = \frac{1}{n}\sum_{p\in P} \arg\min_{q\in Q} C(p,T(q)) + \frac{1}{m}\sum_{q\in Q} \arg\min_{p\in P} C(p,T(q)) \tag{5.14}$$

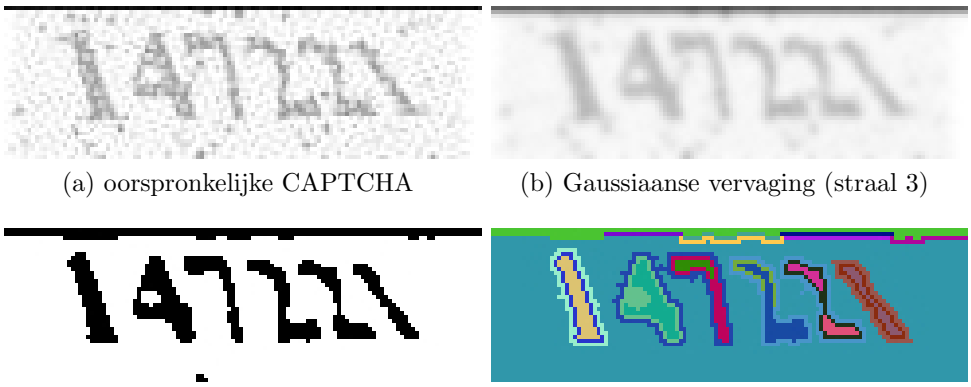
In vergelijking 5.14 zijn P en Q twee verschillende vormen. n is het aantal iteraties dat gebruikt werd in de TPS methode. De functie C is de kost functie uit vergelijking 5.3 en T de geschatte transformatie.

In deze thesis wordt gebruik gemaakt van de "MNIST database for handwritten digits". Deze database bevat 60000 afbeeldingen van cijfers waarmee vergeleken kan worden. Door elke vorm te vergelijken met de afbeeldingen uit de MNIST database, kan een accurate schatting gemaakt worden van het onderliggende cijfer. De MNIST database bevat ook test afbeeldingen die gebruikt kunnen worden om het algoritme te testen. Er werd reeds aangetoond[8] dat shape contexts een zeer goede performantie halen op deze dataset met 99,4% juistheid. Aangezien er alleen maar cijfers in de database zitten, zal er in deze thesis geen gebruik gemaakt worden van CAPTCHA's met letters. De procedure is echter hetzelfde. Indien er een database beschikbaar is met letters, kan deze gebruikt worden om alle andere CAPTCHA's te analyseren.

Het effectieve matching van CAPTCHA's met behulp van shape contexts vereist wel dat

de letters al van elkaar gescheiden zijn. Dit is een groot nadeel aangezien vele CAPTCHA's de grootste moeite doen om deze procedure zo moeilijk mogelijk te maken. In het volgende hoofdstuk zal er echter een methode besproken worden om deze segmentatie te omzeilen.

5.5 Voorbeeld



(a) oorspronkelijke CAPTCHA (b) Gaussiaanse vervaging (straal 3)

(c) drempelwaarde (d) segmentatie

Figuur 5.7: Shape Context matching CAPTCHA voorbeeld 1

Figuur 5.7a laat een relatief eenvoudige CAPTCHA zien. De reden voor het kiezen van deze CAPTCHA is het feit dat deze automatisch gesegmenteerd kan worden. Het is dus mogelijk om deze CAPTCHA te breken zonder tussenkomst van de gebruiker. De meeste CAPTCHA's zijn echter veel complexer, en dus bestendig tegen dit soort aanvallen. Het voorbeeld is wel geschikt om de werking van Shape Context matching te illustreren. Na

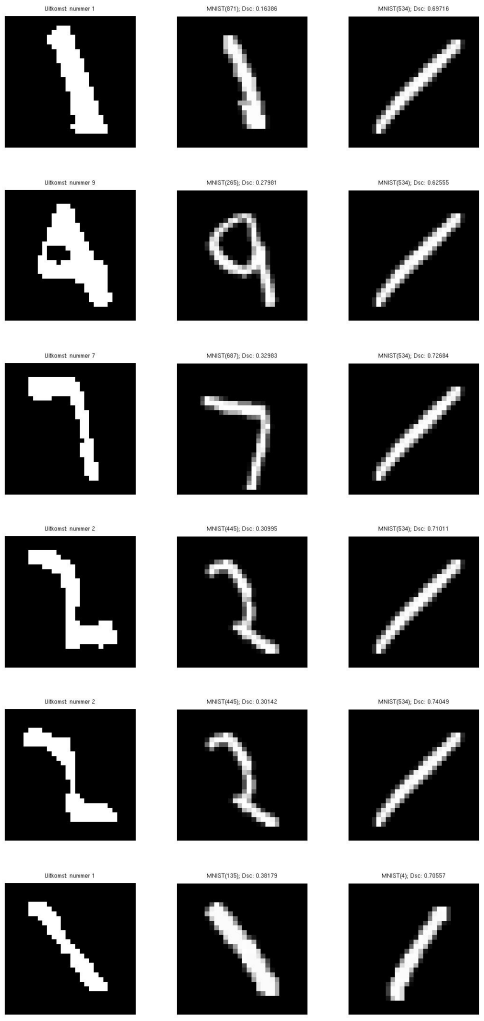
Figuur 5.8: Individuele karakters uit figuur 5.7

segmentatie (figuur 5.7d) kunnen de karakters uit de CAPTCHA gehaald worden. Deze karakters worden daarna individueel geanalyseerd. Er wordt geen rekening gehouden met de volgorde. De te analyseren karakters worden afzonderlijk getoond in figuur 5.8. De karakters worden opgevuld tot een vierkante afbeelding. De afbeeldingen zijn binair.

Bij het matchen werden enkel de eerste 1000 afbeeldingen uit de MNIST database gebruikt. Uit onderzoek blijkt dat bij het gebruiken van alle 60000 afbeeldingen het resultaat nog

weinig verandert. In de eerste 1000 afbeeldingen zitten genoeg verschillende afbeeldingen om een juiste match te vinden. Er worden per afbeelding 45 samples gekozen. Het aantal bins dat gekozen wordt voor de shape context op te stellen is 120 (5 bins voor $log(r)$ en 24 voor θ, zie figuur 5.5). Het aantal iteraties dat gebruikt is bij het opstellen van de transformatie tussen de vormen, is 1. Er zal verder in dit hoofdstuk aangetoond worden dat een hoger aantal iteraties niet werkt.

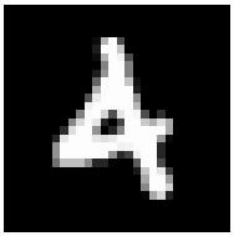
Het resultaat van het matchen van de afbeeldingen met de MNIST database wordt getoond in figuur 5.9. De linkse kolom laat de origine afbeelding zien met als titel het gevonden resultaat. De middelste kolom laat het resultaat zien. De titel geeft de index van de afbeelding weer uit de MNIST database en de shape distance (D_{sc}). De rechtse afbeelding is hetzelfde, maar dan met de slechtste match. Merk op dat de afbeeldingen in de linkse kolom allemaal op elkaar lijken. Dit is geen toeval aangezien de oorspronkelijke karakters schuin gedrukt zijn. Ze hellen allemaal naar links. De slechtste overeenkomst met een naar links hellende afbeelding is een naar rechts hellende afbeelding.



Figuur 5.9: Shape Context matching resultaten: links - origineel, midden - beste match, rechts - slechtste match

In dit geval zijn alle karakters juist gematcht behalve de 4. Door echter alle 60000 afbeeldingen van de MNIST database te gebruiken, bekomt men wel een goed resultaat (zie figuur 5.10 De 40577ste afbeelding had een $D_{sc} = 0.2409$ terwijl het gevonden resultaat in figuur 5.9 $D_{sc} = 0.27981$.

Er zal nu een uitgebreide analyses volgen. De volledige werking van de Shape Context matching zal uitgelegd worden aan

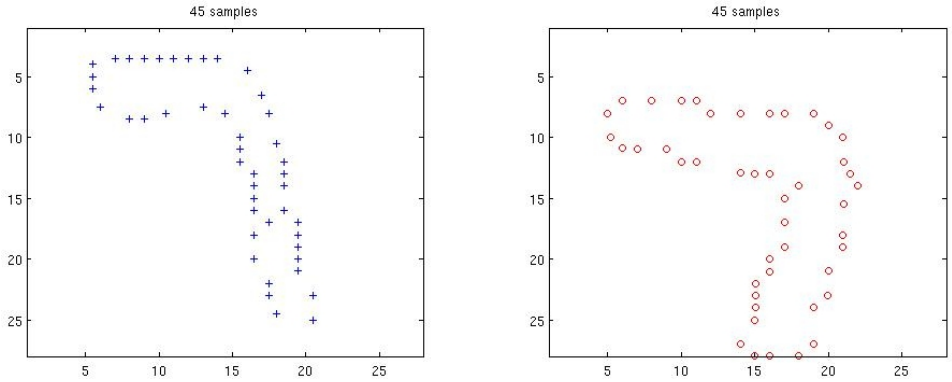


Figuur 5.10: Beste overeenkomst met de volledige MNIST database

de hand van een karakter uit bovenstaand
voorbeeld.

Uitgebreide analyse

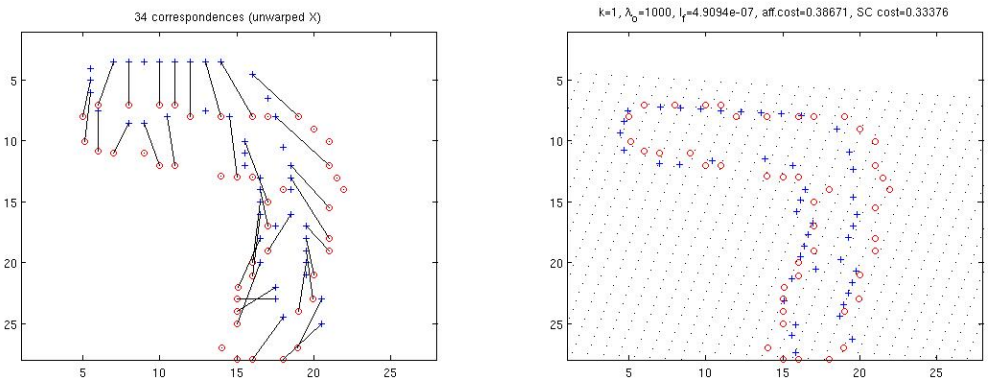
Eerst wordt karakter 7 uit figuur 5.8c geanalyseerd. De beste overeenkomst die gevonden is, is afbeelding nummer 687 uit de MNIST database. Eerst worden beide afbeeldingen gesampled. Dit is te zien in figuur 5.11. De linkse afbeelding is het karakter uit de CAPTCHA en rechts is het referentiekarakter te zien.



Figuur 5.11: Voorbeeld van sampling

Vervolgens moeten de overeenkomsten gezocht worden tussen de samples uit de verschillende vormen. Dit gebeurt zoals uitgelegd in sectie 5.4.1. Het resultaat is te zien in figuur 5.12a. De verschillende vormen worden hier over elkaar gelegd en de samples die overeenkomen worden verbonden met een zwarte lijn.

Na het berekenen van de overeenkomsten tussen de vormen, wordt een transformatie gemodelleerd, zoals uitgelegd in sectie 5.4.3.



(a) (b)

Figuur 5.12: overeenkomsten tussen de verschillende vormen

Door de transformatie te modelleren, kan de kost (D_{sc}) berekend worden om de transformatie uit te voeren. Zoals eerder uitgelegd, moet deze kost zo klein mogelijk gehouden worden. De referentieafbeelding met de laagste kost is de beste overeenkomende vorm. Ter illustratie wordt in figuur 5.12b de getransformeerde afbeelding getoond. Stel nu dat er meerdere iteraties zouden plaatsvinden bij het modelleren van de transformatie. De volgende transformatieberekening zou vertrekken van de reeds bekomen resultaten. De kost gaat dus altijd kleiner worden in de n-de iteratie. Deze kost is dus niet meer representatief ten opzichte van de referentieafbeelding. Met meerdere iteraties zou zelfs een ander karakter gematched kunnen worden met de bovenstaande 7.

D_{sc} is in dit voorbeeld 0.32979. Deze kost verschilt met het resultaat bekomen in figuur 5.9c. Dit komt omdat bij elke analyse de samples willekeurig gekozen worden. Dit zowel bij de referentievorm als bij de originele. De kost gaat dus altijd verschillen tussen verschillende analyses. Over het algemeen gaat dit echter om zeer kleine verschillen. Moest dit niet zo zijn, zou shape context matching geen goede matching techniek zijn.

Hoofdstuk 6

Distance Transform Matching

6.1 Introductie

Distance Transform (DT)[12][13] matching biedt een efficiente oplossing voor vorm-gebaseerde object herkenning. Het houdt het matchen van arbitrair gevormde objecten in alsook het matchen van geparametriseerde objecten. Omdat het algoritme vormen herkent aan de hand van voorbeeld vormen, is het zeer flexibel en aanpasbaar zonder herprogrammatie. Dit is een zeer krachtig voordeel wanneer er gewerkt wordt met CAPTCHA's aangezien er vele soorten van bestaan. Het algoritme werkt met pixel-gebaseerde errorwaarden waardoor het niet meer noodzakelijk is de afbeelding op voorhand te segmenteren. DT werkt dus ook wanneer er veel ruis of andere ongewenste objecten in de afbeelding zitten. Ook is het algoritme snel en goedkoop.

Aan de andere kant heeft DT matching ook enkele negatieve eigenschappen. Het algoritme is gevoelig voor veel geometrische transformaties, waaronder rotaties en schaleringen. Het komt er dus op neer om zeer veel voorbeeld afbeeldingen ter beschikking te stellen zodat het algoritme voldoende diversiteit kan herkennen. Net zoals bij Shape Context Matching wordt voor DT matching in deze thesis de MNIST database gebruikt. Ook hier geldt de beperking dat de database enkel cijfers herkent.

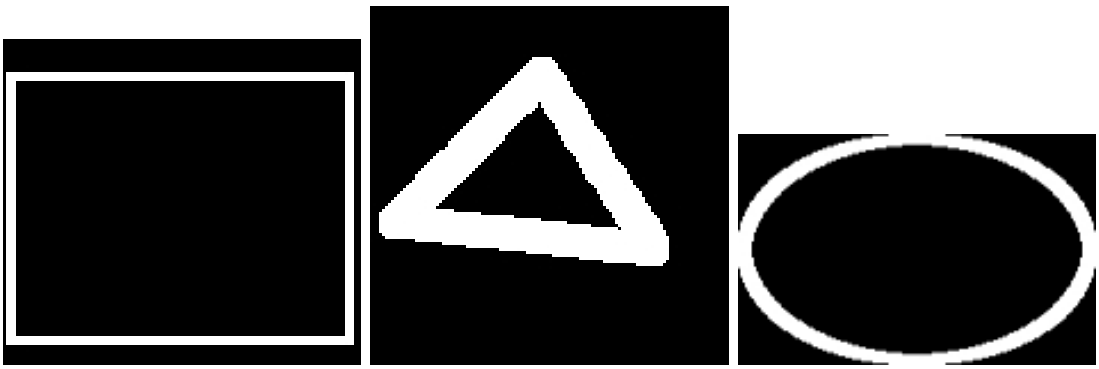
DT matching heeft reeds verschillende toepassingen. Een voorbeeld hiervan is het herkennen van objecten door kleine camera's in voertuigen. Dit kan gaan van het herkennen van voorliggende auto's om botsingen te vermijden, tot het herkennen van verkeersborden voor

de snelheidslimiet te kunnen lezen. Ook wordt deze methode vaak gebruikt in militaire toepassingen. DT matching kan zeer gemakkelijk doelwitten detecteren.

DT matching heeft verschillende varianten, waaronder chamfer matching[13] of Hausdorff matching[14]. Het verschil zal in de volgende sectie worden uitgelegd.

6.2 Algoritme

Het algoritme krijgt als invoer een afbeelding waarin een vorm (of meerdere vormen) moet herkend worden. Dit wordt de doelafbeelding genoemd. Als extra invoer krijgt het algoritme een aantal referentieafbeeldingen ter beschikking. Deze afbeeldingen stellen de vormen voor die gezocht moeten worden in de doelafbeelding. Typisch zijn de referentieafbeeldingen binair. Om de werking van het DT matching algoritme te illustreren, zal er gebruik gemaakt worden van een aantal simpele vormen. Deze zijn te zien in figuur 6.1. Bij het breken van CAPTCHA's worden deze referentievormen natuurlijk karakters.



(a) (b) (c)

Figuur 6.1: Enkele referentievormen

De doelafbeelding kan een volledig willekeurige abeelding zijn. Als de referentievormen niet voorkomen in de doelafbeelding, zal de lokatie gevonden worden die het meeste lijkt op de referentievorm. De doelafbeelding die gebruikt zal worden is te zien in figuur 6.2a. Het is duidelijk dat dit een eenvoudige afbeelding is, en dat de vormen uit figuur 6.1 er gemakkelijk in terug te vinden zijn. Dit is gedaan om gemakkelijker de werking van het algoritme te kunnen tonen.

6.2.1 Converteren naar binair

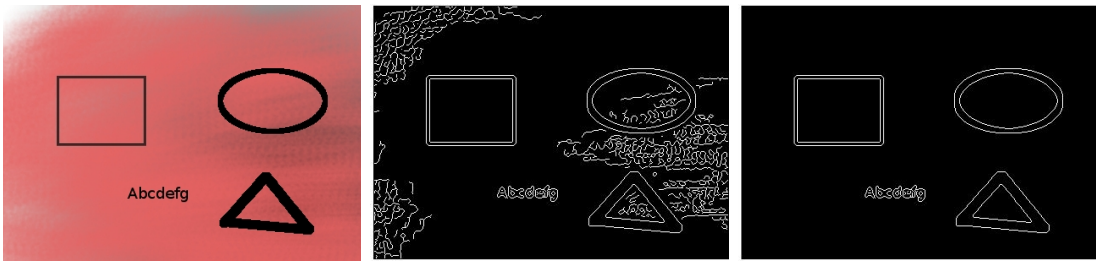
De doelafbeelding wordt als eerste geconverteerd naar een binaire afbeelding. Deze stap is van cruciaal belang indien er veel ruis in de afbeelding zit. Het converteren naar binair gebeurt door de randen op te sporen met behulp van een edge-detection algoritme. Een veel gebruikt algoritme is het Canny edge-detection algoritme[15].

Het Canny edge-detection algoritme is een zeer eenvoudig, doch zeer efficiënt algoritme in verband met edge-detection. De uitvinder van dit algoritme, John F. Canny, probeerde het optimale edge-detection algoritme te bekomen. Optimaal betekent volgende dingen:

- goede detectering: het algoritme moet zoveel mogelijk echte randen aanduiden.
- goede lokalisering: de randen die aangeduid worden moeten zich zo dicht mogelijk bevinden bij de rand van de werkelijke afbeelding.
- minimale output: enkel echte randen mogen voorkomen en ruis moet zoveel mogelijk weggefilterd worden.

Om deze criteria te behalen, wordt er in het Canny algoritme al ruis onderdrukking toegepast. Dit komt zeer goed van pas in het analyseren van CAPTCHA's. De agressiviteit van de ruis onderdrukking in het algoritme is in te stellen door een aantal variabelen, namelijk δ en σ. Er wordt Gaussiaanse vervaging gebruikt door het Canny algoritme, waarbij σ de standaardafwijking is van de Gaussiaanse curve. δ is een drempelwaarde die ingevoerd wordt bij het converteren naar binair.

Bij voorkeur worden de zwakke randen weggelaten uit de doelafbeelding zodat alleen de belangrijke kenmerken overblijven. Dit is niet altijd een voor de hand liggende taak. Door de juiste δ en σ waarden te kiezen kan toch een gewenst resultaat behaald worden. Een voorbeeld van deze procedure is te zien in figuur 6.2. In figuur 6.2b is Canny edge-detection toegepast op de doelafbeelding. Door een δ-waarde van 0.2 te gebruiken, worden de zwakke randen geëlimineerd. De σ werd hier niet aangepast aangezien een standaardwaarde gelijk aan 1 een goed resultaat opleverde.



(a) doelafbeelding (b) Canny edge-detection (c) Canny met $\delta = 0.2$

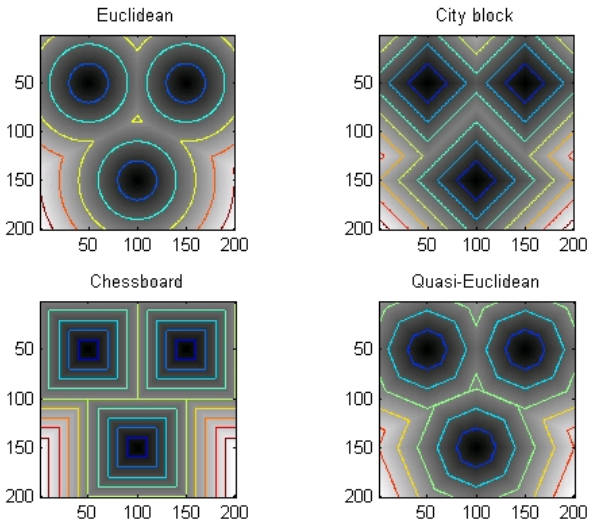
Figuur 6.2: Converteren van doelafbeelding naar een binaire doelafbeelding

6.2.2 Distance Transformation

De volgende stap in de DT matching techniek is het genereren van een *distance map*. Deze distance map wordt berekend aan de hand van de doelafbeelding. De berekening die gedaan wordt, noemt men de *distance transformation*. Deze distance transformation is het hart van de DT maching techniek. De distance map moet per doelafbeelding slechts één maal berekend worden. Dit heeft een grote invloed op de snelheid van het algoritme. Door eenmalig een distance map te creëren, kan zeer veel rekenwerk bespaard en vereenvoudigd worden.

Een distance map bestaat, zoals de naam impliceert, uit afstanden. Deze afstanden kunnen berekend worden op verschillende manieren. Elke afstandsberekening kan hiervoor dienen. Een van de bekendste afstandsberekeningen is de Euclidische afstand. Euclidische afstandsberekeningen zijn echter tijdsintensief. Er bestaan gelukkig efficiëntere algoritmes om de Euclidische afstand te benaderen. Enkele voorbeelden zijn chamfer distance, Hausdorff distance, ... Een visuele voorstelling van de benadering van Euclidische afstanden is te zien in figuur 6.3. Het optimaliseren van distance matching valt echter buiten het bereik van deze thesis. We zullen dus vanaf hier de gewone Euclidische afstand gebruiken. Deze is het nauwkeurigste en zal dus het beste resultaat geven.

De distance map wordt nu als volgt opgesteld. Voor elke pixel in de afbeelding wordt de afstand berekend tot de dichstbijzijnde rand in de doelafbeelding. Wanneer er aan de afstanden een kleurwaarde wordt toegekend, kan de distance map gemakkelijk getoond worden als een nieuwe afbeelding. In figuur 6.4d en 6.4a zijn twee voorbeelden te zien van distance maps. Hoe donkerder de kleur, hoe kleiner de afstand. Figuur 6.4d is de distance map berekend met doelafbeelding 6.2b als invoer. Figuur 6.4a is berekend met

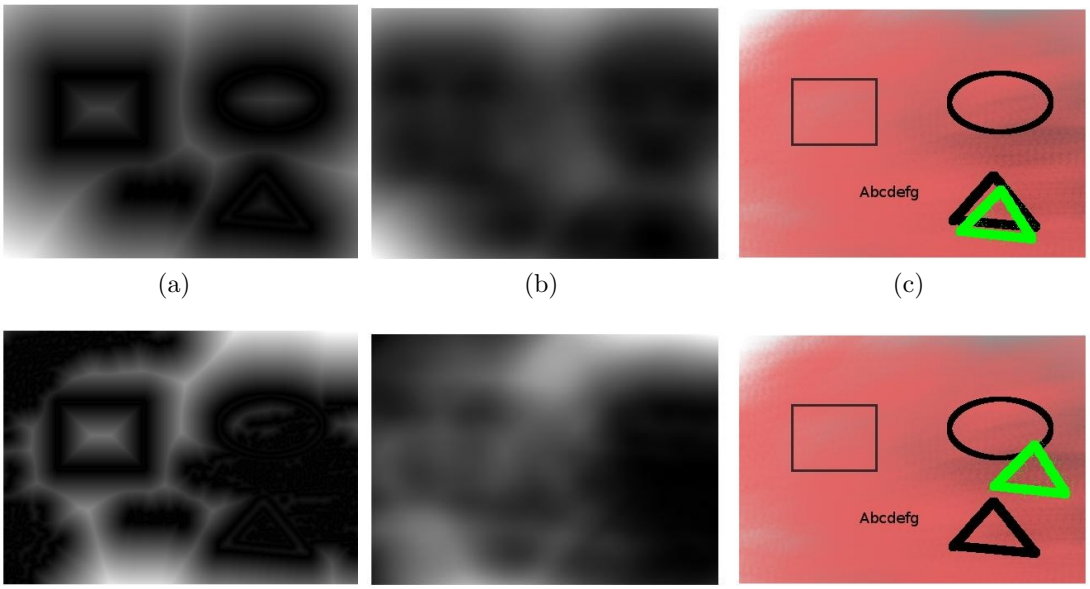


Figuur 6.3: Verschillende distance map berekeningen

figuur 6.2c als invoer. Het is duidelijk te zien dat het noodzakelijk is om overtollige randen te verwijderen uit de doelafbeelding. Wanneer dit niet gedaan wordt, is het veel moeilijk om de verschillende vormen in de doelafbeelding te onderscheiden.

6.2.3 Vorm zoeken

De laatste stap in het matchen van de referentievorm met de doelafbeelding is het zoeken van de beste overeenkomst tussen de referentie- en doelafbeelding. Aangezien er een distance map gegenereerd werd, wordt deze taak aanzienlijk versimpeld. Er wordt een convolutiemasker opgesteld op basis van de referentievorm. Aangezien de referentievorm binair is, kan de referentievorm op zich al dienen als convolutiemaster. Door een twee dimensionale convolutie uit te voeren op de doelafbeelding, wordt een nieuwe map gecreëerd. Deze maps zijn te zien in figuur 6.4b en 6.4e. Donker komt weer overeen met lage waarden. De laagste waarden in de convolutiemap duiden de beste overeenkomsten aan met de referentieafbeelding. Hoe donkerder de map, des te beter de overeenkomst. Het resultaat van de matching is te zien in de laatste kolom van figuur 6.4. Nogmaals is duidelijk te zien dat het resultaat veel beter is wanneer de zwakke randen verwijderd zijn. Aangezien de referentieafbeelding niet volledig overeenkomt met een vorm in de afbeelding, is er een kleine afwijking te zien. Voor het matchen van CAPTCHA's is het dus noodzakelijk om zeer veel referentievormen te gebruiken.



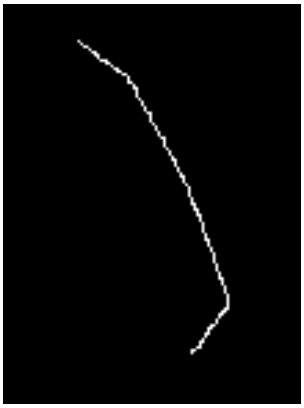
(a) (b) (c)

(d) (e) (f)

Figuur 6.4: (a) en (d): distance maps van doelafbeeldingen uit figuur 6.2, (b) en (e): convolutie met figuur 6.1, (c) en (f): het resultaat van de matching

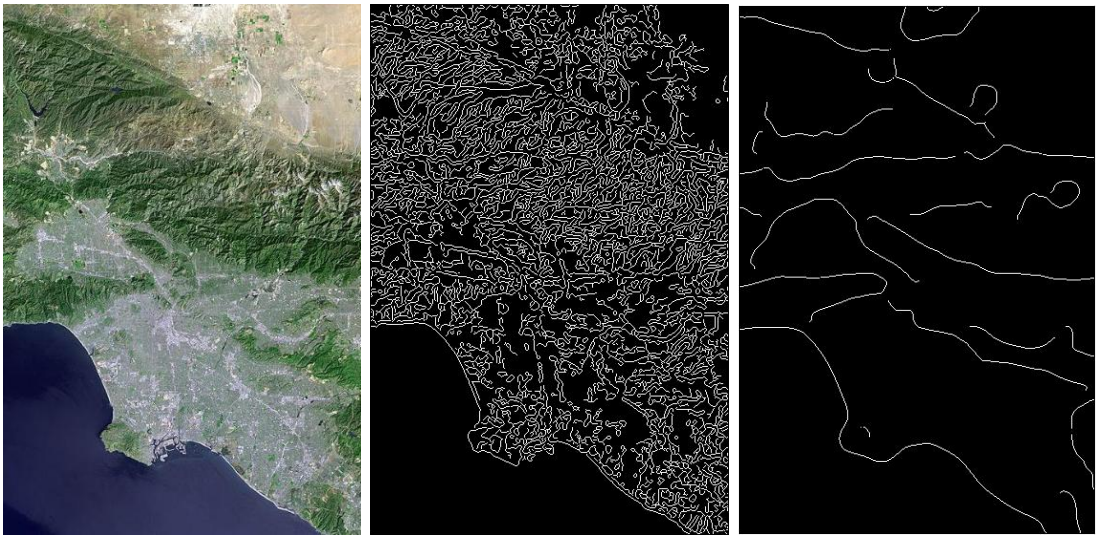
6.3 Voorbeeld

Om aan te tonen dat DT matching ook werkt op willekeurige afbeeldingen, wordt als doelafbeelding figuur 6.6a gebruikt. Het is een deel van een landkaart. De vorm die gezocht moet worden is te zien in figuur 6.5. Zoals te zien komt de vorm overeen met de kust op de doelafbeelding. Deze afbeeldingen hebben helemaal niets te maken met CAPTCHA's. Ze dienen gewoon om aan te tonen dat het een algemene matching techniek is. In hoofdstuk 7 zal de techniek gebruik worden in verband met CAPTCHA's.



Figuur 6.5: Een binaire vorm die gezocht wordt in een doelafbeelding

De eerste uitdaging is het genereren van een binaire afbeelding van de doelafbeelding. Wanneer het Canny algoritme gebruikt wordt zonder aanpassingen, bekomt men



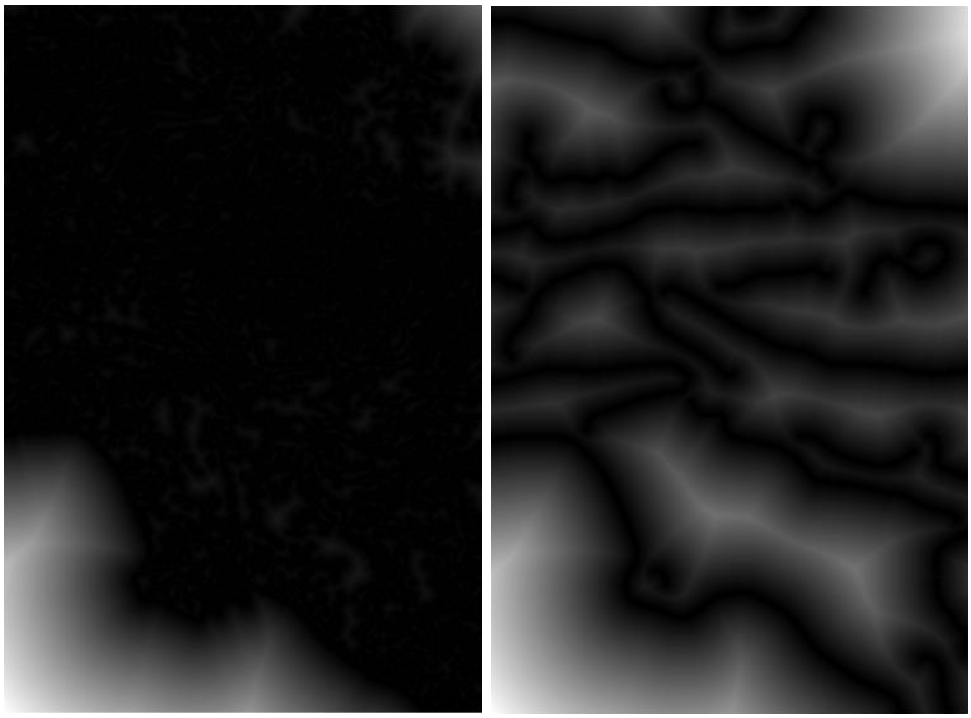
(a) doelafbeelding (b) Canny edge-detection (c) Canny met $\delta = 0.5$ en $\sigma = 7$

Figuur 6.6: Converteren van doelafbeelding naar een binaire doelafbeelding

het resultaat uit figuur 6.6b. Het is vrij duidelijk dat dit geen bruikbaar resultaat is. Er zijn veel te veel randen aanwezig in de binaire afbeelding. Door de Canny parameters aan te passen, kan er een behoorlijk goed resultaat verkregen worden. Door δ gelijk te stellen aan 0.5 en σ aan 7, bekomt men het resultaat zichtbaar in figuur 6.6c. Hier zijn enkel nog de belanrijke en relevante randen overgebleven.

Nadat de edge-detection succesvol beëindigd is, kan de distance map opgesteld worden. In dit voorbeeld werd eveneens gebruik gemaakt van de Euclidische afstandsberekeningen. Nogmaals is duidelijk zichtbaar dat een distance map gebaseerd op teveel randen compleet onbruikbaar is om DT matching mee uit te voeren. Figuur 6.7a is bijna volledig zwart. Hier zal nooit een referentieafbeelding in gevonden kunnen worden.

Vervolgens wordt een convolutiemasker opgesteld met behulp van de referentievorm. Na convolutie met dit masker ziet het resultaat eruit zoals figuur 6.8a. Het is duidelijk te zien dat op de plaats van de kustlijn een donkere streep aanwezig is. Hier is de vorm dus het beste overeengekomen. Figuur 6.8b laat dan ook de positie zien waar de vorm het beste overeenkomt. Dit is op de gewenste plaats.



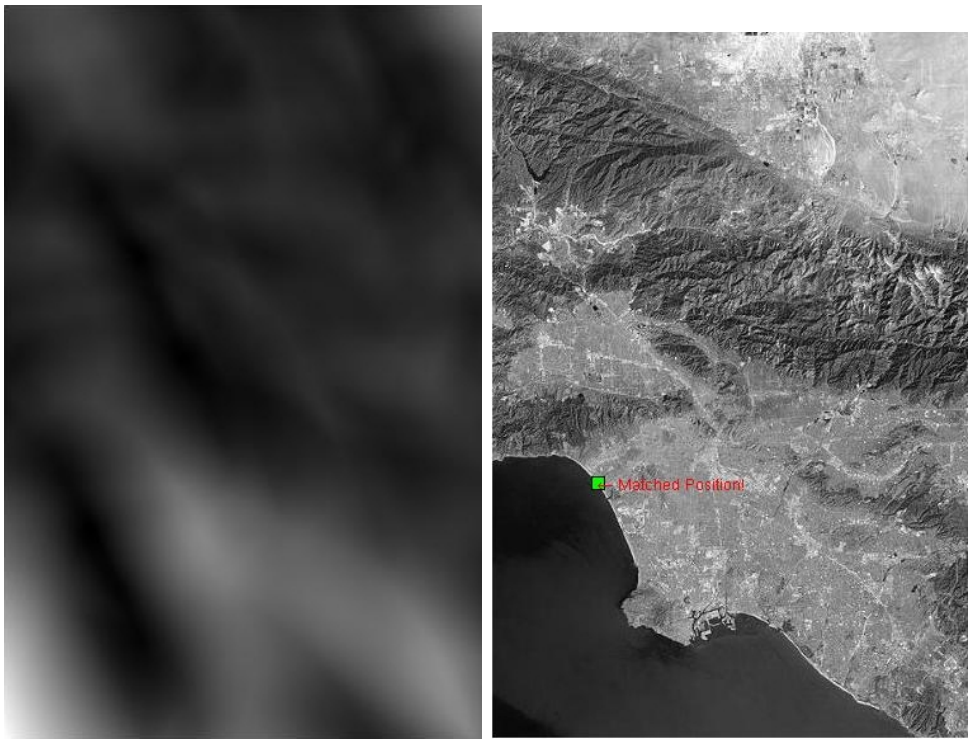
(a) DT van figuur 6.6b (b) DT van figuur 6.6c

Figuur 6.7: Distance transformations van doelafbeeldingen uit figuur 6.6

6.3.1 CAPTCHA voorbeeld

Om aan te tonen dat de techniek niet enkel geldt voor willekeurige afbeeldingen, zal nu een CAPTCHA geanalysserd worden met behulp van het Distance Transform Matching algoritme. Aangezien DT maching referentievormen zoekt in een doelafbeelding, zal er een referentievorm moeten gekozen worden. Als eerste zal een vorm gekozen worden die voorkomt in de CAPTCHA. Dit wordt gedaan om aan te tonen dat het DT matching algoritme de vormen goed kan lokaliseren, zelfs met ruis in de afbeelding. Ten tweede zal een vorm gezocht worden die niet voorkomt in de CAPTCHA. Het zal duidelijk worden dat het algoritme nog steeds een lokatie zal vinden. Dit zal de lokatie zijn van de vorm die het meeste lijkt op de referentievorm.

Als voorbeeld zal opnieuw de phpBB CAPTCHA gebruikt worden uit figuur 4.1. Zoals reeds gezegd, kan het DT matching algoritme enkel vormen herkennen. Figuur 6.9 laat zien welke referentieafbeeldingen gebruikt zullen worden.



(a) convolutiemap (b) resultaat

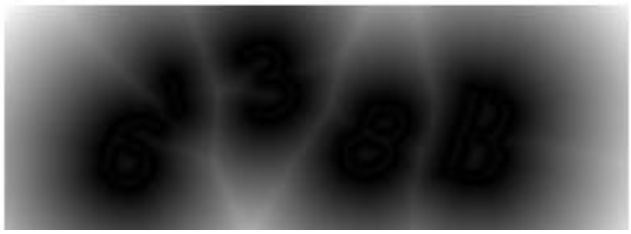
Figuur 6.8: matching van de vormen

(a) (b)

Figuur 6.9: referentieafbeeldingen 1 en 2

Figuur 6.9a zal als eerste gezocht worden in de doelafbeelding. De DT matching procedure is dezelfde als in het vorige voorbeeld. De distance map van de doelafbeelding wordt opgesteld. Het resultaat hiervan is zichtbaar in figuur 6.10. Hier is al duidelijk te zien dat de afbeelding goed gesegmenteerd is. De verschillende karakters zijn volledig gescheiden van elkaar.

Vervolgens wordt convolutie toegepast met het convolutiemaster gelijk aan figuur 6.9a. De convolutiemap is te zien in figuur 6.11a. Na convolutie wordt de beste lokatie gezocht in de afbeelding. Dit komt overeen met het meest donkere stuk uit de convolutiemap. Het resultaat is te zien in figuur 6.11b. Het DT matching algoritme is dus zeer goed in het lokaliseren van een referentieafbeelding in een willekeurige figuur.



Figuur 6.10: Distance map

(a) convolutie map (b) resultaat

Figuur 6.11: Resultaat van referentieafbeelding 1

Als tweede voorbeeld wordt referentieafbeelding 6.9b gezocht in dezelfde doelafbeelding. Het resultaat is te zien in figuur 6.12. Hoewel de referentieafbeelding niet voorkomt in de doelafbeelding, vindt het algoritme toch een lokatie. Deze eigenschap zal in hoofdstuk 7 gebruikt worden om DT matching toch te kunnen gebruiken in het analyseren van CAPTCHA's. Op zich kan het DT matching algoritme gebruikt worden om CAPTCHA's te analyseren. Het grote nadeel is echter dat voor elke referentieafbeeling een mogelijke lokatie gaat gevonden worden. Dit komt omdat de lokatie overeenkomt met de donkerste plaats in de convolutiemap. Deze plaats komt natuurlijk overeen met een concrete waarde. Er kan dus gemakkelijk een restrictie opgelegd worden dat boven een bepaalde waarde geen lokatie gevonden wordt. Dit is echter niet gewenste resultaat in het analyseren van CAPTCHA's. Aangezien een CAPTCHA het typisch moeilijk maakt om een lokatie te vinden, gaat het DT matching algoritme geen enkele lokatie meer vinden indien bovenstaande restrictie wordt toegepast.

(a) convolutie map (b) resultaat

Figuur 6.12: Resultaat van referentieafbeelding 2

Hoofdstuk 7

Combinatie van technieken

In hoofdstukken 5 en 6 zijn twee technieken besproken die individueel vrij efficiënt zijn in het matchen van vormen in doelafbeeldingen. Elk van de algoritmen heeft echter zijn nadelen. Shape Context matching is een zeer robuuste techniek om vormen te matchen, zelfs wanneer de vormen getransformeerd zijn op een of andere manier. Zoals eerder vermeld is dit een groot voordeel bij het breken van CAPTCHA's. Het grote nadeel van deze aanpak is echter dat het algoritme niet in staat is de vormen te lokaliseren in de doelafbeelding. Er zijn dus zeer veel voorgaande bewerkingen nodig op de gegeven CAPTCHA alvorens het resultaat kan gegeven worden.

Het Distance Transform matching algoritme is op zijn beurt zeer goed in het vinden van referentievormen in een doelafbeelding. Mits een grote database van referentievormen, kan het DT matching algoritme zeer robuust vormen uit een afbeelding halen. Het probleem hierbij is dan weer dat CAPTCHA's over het algemeen de vormen van de karakters verbuigen of transformeren zodat ze moeilijk te herkennen zijn. Het blijft echter wel mogelijk om met Distance Transform matching kandidaten te vinden in een afbeelding waar eventueel letters kunnen staan. Door het gebruiken van de MNIST database, zijn er veel referentieafbeeldingen beschikbaar.

Wanneer men nu het DT algoritme kan combineren met het Shape Contexts matching algoritme, kan men de voordelen van beide gebruiken om een CAPTCHA beter te analyseren.

7.1 Lokaliseren van kandidaatposities

(a) Kandidaat lokaties (b) Werkelijke lokaties

Figuur 7.1: Mogelijke kandidaatposities

Als voorbeeld wordt de CAPTCHA uit figuur 4.1 hergebruikt. In hoofdstuk 4 was het nodig om een segmentatiealgoritme toe te passen op de CAPTCHA. Zonder goede segmentatie konden de karakters nie geanalyseerd worden. Er wordt nu aangetoond dat hetzelfde kan gerealiseerd worden door middel van Distance Transform matching.

Omdat de MNIST database bestaat uit grote hoeveelheden afbeeldingen, komt de eerste stap in het DT matching algoritme volledig tot zijn recht. Door op voorhand een distance map te genereren, kunnen nadien alle referentieafbeeldingen gematched worden ten opzichte van die distance map. Het berekenen van de distance map vereist de meeste rekenkracht. Door deze te hergebruiken wordt het algoritme veel performanter.

Vervolgens worden alle referentieafbeeldingen gezocht in de CAPTCHA. Alle gevonden posities worden aangeduid met een groene stip. Na het berekenen van 1000 referentieafbeeldingen, krijgt men het restultaat uit figuur 7.1a. In de 1000 referentieafbeeldingen zitten natuurlijk ook karakters die niet voorkomen in de CAPTCHA. Het cijfer 2 zal bijvoorbeeld niet gevonden worden. Aangezien het DT matching algoritme simpelweg de beste overeenkomst zoekt, zal elk karakter altijd een mogelijke lokatie krijgen. Dit is een voordeel wanneer CAPTCHA's geanalyseerd worden. Als dan grote hoeveelheden referentieafbeeldingen gebruikt worden, is de kans veel groter dat alle karakterposities gevonden worden. Dit gebeurt in dit voorbeeld met de B. Deze zal nergens gevonden worden, maar toch wordt de B aangeduid als een mogelijke kandidaatpositie.

Al deze kandidaatposities worden dan gegroepeerd. De gecreëerde groepen vormen de segmenten waarin de karakters zich bevinden. Het resultaat is te zien in figuur 7.1b. De segmentatiestap die voordien nodig was is nu succesvol omzeild. De volgende stap is het analyseren van de verschillende segmenten.

7.2 Kandidaten analyseren

Nu de CAPTCHA gesegmenteerd is, kan het Shape Context Matching algoritme ingeschakeld worden. De werkwijze is dezelfde als in sectie 5.5. Eerst worden de verschillende segmenten binair gemaakt. Dit gebeurde voordien in de segmentatiestap, maar aangezien DT matching nu wordt gebruikt is dit nog niet gebeurd. Het binair maken van de segmenten gebeurt door een drempelwaarde in te voeren.

Aangezien het geen complexe CAPTCHA is, is het resultaat behoorlijk goed. Alle cijfers zijn succesvol herkend. De eerste 1000 afbeeldingen uit de database werden gebruikt. De B in de CAPTCHA is gematched met het cijfer 8. De oorzaak hiervan is natuurlijk dat er geen letters in de MNIST database zitten. De resultaten zijn te zien in figuur 7.2. Het originele karakter wordt getoond met daarnaast de beste overeenkomst uit de database.

Figuur 7.2: Matching resultaat

Hoofdstuk 8

Case Studies

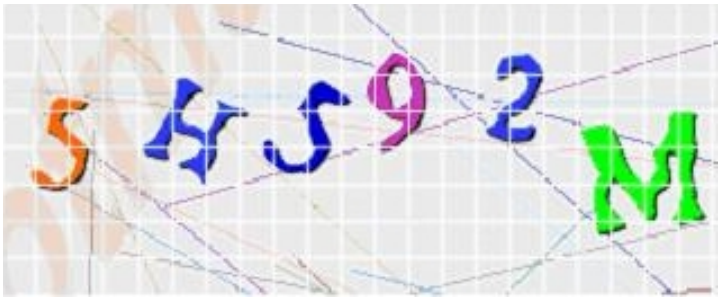
In dit hoofdstuk zullen een aantal CAPTCHA's in detail besproken worden. De technieken uit hoofdstuk 7 zullen hierop worden toegepast. Aan de hand van de resultaten hiervan zullen de sterke en zwakke punten van de CAPTCHA geanalyseerd worden.

8.1 Reeds besproken CAPTCHA's

In deze thesis werden reeds een aantal CAPTCHA's gebruikt als voorbeeld voor de besproken technieken. De phpBB CAPTCHA is hier een voorbeeld van (figuur 4.1). Deze CAPTCHA's zijn over het algemeen gemakkelijk te breken. De phpBB CAPTCHA is gemakkelijk te bewerken zodat de achtergrond ruis wegvalt. Daarnaast is het segmenteren van de afbeelding eenvoudig. Door enkel Shape Context matching toe te passen is de CAPTCHA al breekbaar. Wanneer de combinatie van Distance Transform matching en Shape Context matching wordt toegepast, is het nog eenvoudiger de CAPTCHA te breken. DT matching segmenteert de CAPTCHA feilloos, waarna Shape Context matching de karakters probleemloos herkent.

De CAPTCHA uit figuur 5.7 is ook geen sterke CAPTCHA. Zoals aangetoond in hoofdstuk 5 is deze CAPTCHA zonder veel moeite te breken. Het enige wat deze CAPTCHA doet is een beetje ruis introduceren en de cijfers scheef zetten. Een standaard OCR algoritme zou zelfs in staat zijn hier een behoorlijk resultaat voor te geven.

8.2 Willekeurige CAPTCHA



Figuur 8.1

In dit voorbeeld zal een nieuwe willeurige CAPTCHA geanalyseerd worden. De CAPTCHA is te zien in figuur 8.1. Deze CAPTCHA introduceert drie obstakels die overwonnen moeten worden. Het duidelijkste probleem zijn de strepen door de karakters. Ook wordt er een raster getekend over de CAPTCHA. Als laatste worden de karakters een beetje scheef gezet.

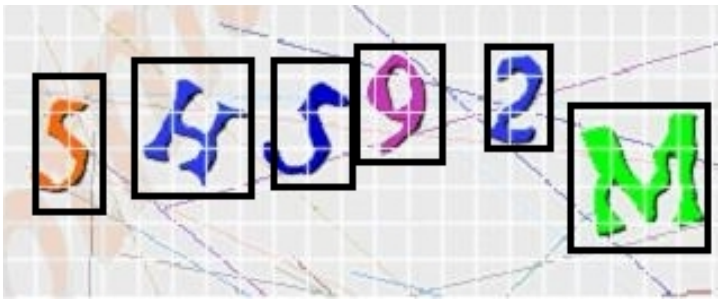
Figuur 8.2

Al deze problemen zijn echter gemakkelijk op te lossen met de combinatie van technieken uit hoofdstuk 7. In figuur 8.2 is te zien wat het DT matching algoritme gedaan heeft met de CAPTCHA.

De kandidaatlokaties zijn te zien in figuur 8.3. Het is duidelijk dat de kakakters goed gesegmenteerd zijn.

In figuur 8.4 zijn de karakters apart te zien zoals ze gesegmenteerd zijn met behulp van DT matching.

Wanneer nu Shape Context matching wordt toegepast, wordt het resultaat bekomen zoals



Figuur 8.3

Figuur 8.4

te zien in figuur 8.5. Enkel de cijfers zijn geanalyseerd aangezien de letter H en M toch niet gevonden zullen worden.

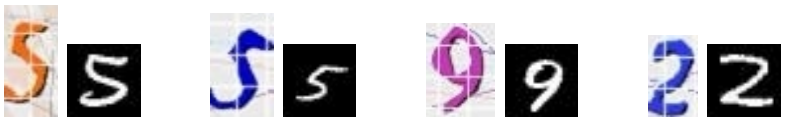
Dit soort CAPTCHA's is duidelijk gemakkelijk te breken. De voornaamste reden is dat de letters gescheiden van elkaar voorkomen in de CAPTCHA. Dit maakt het mogelijk voor het DT matching algoritme om de kandidaatlokaties te lokaliseren.

Aangezien Shape Context Matching zelfs vervormde karakters kan herkennen, zit het zwakke punt van het proces uit hoofdstuk 7 in het segmenteren van de karakters.

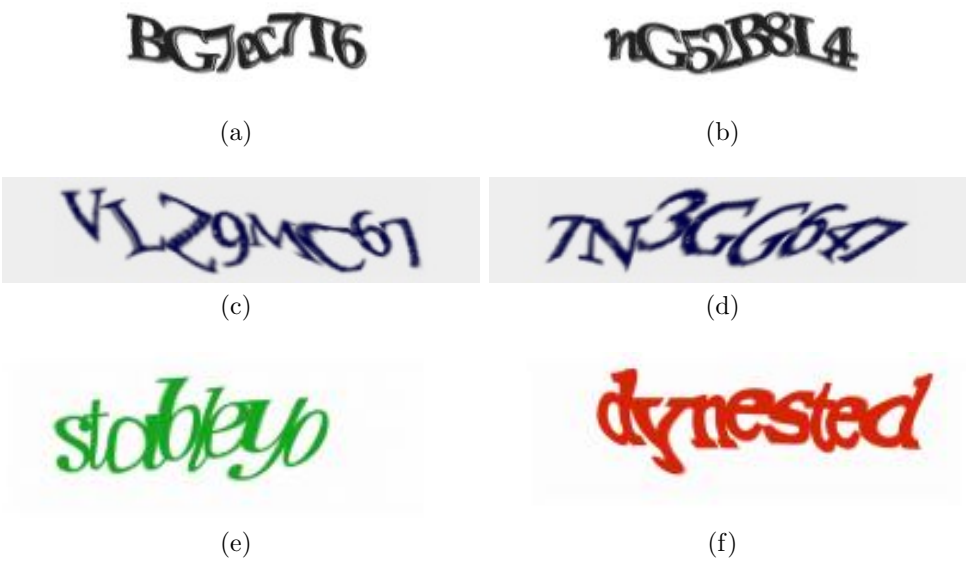
8.3 Bekendere CAPTCHA's

Hier zullen een aantal bekendere CAPTCHA's aan bod komen. Het zal duidelijk worden dat deze CAPTCHA's veel beter en efficiënter zijn, en dus niet gemakkelijk te kraken zijn. In figuur 8.6 zijn verschillende bekende CAPTCHA's getoond. De eerste twee zijn van Yahoo!. De volgende twee van MSN en ten slotte de Google CAPTCHA.

DT matching zal nu worden toegepast op de eerste CAPTCHA (figuur 8.6a). Het resultaat is te zien in figuur 8.7a. Het is onmogelijk om hieruit de aparte karakters te halen. Ter illustratie is het karakter 7 manueel uit de CAPTCHA gehaald. Dit is te zien in figuur



Figuur 8.5



(a) (b)

(c) (d)

(e) (f)

Figuur 8.6: Yahoo!, MSN en Google CAPTCHA

8.7b. Door nu Shape Context Matching toe te passen op deze afbeelding, bekomt men het resultaat uit figuur 8.7c. Shape Context Matching kan dus effectief de karakters herkennen, ook in vervormde omstandigheden.

Door het moeilijk te maken om de karakters uit de CAPTCHA te segmenteren, bekomt men dus een sterke CAPTCHA. In de meeste CAPTCHA's wordt dit bereikt door de karakters dicht tegen elkaar te zetten of zelfs deels in elkaar.

(a)

(b) (c)

Figuur 8.7: Analyse Yahoo! CAPTCHA uit figuur 8.6a

Hoofdstuk 9

Conclusie

In deze thesis zijn twee technieken aan bod gekomen in verband met het analyseren en breken van CAPTCHA's. Beide technieken hebben hun voor- en nadelen.

Als eerste techniek werd het Shape Context Matching algoritme besproken. Dit algoritme kan met behulp van shape contexts vormen met elkaar vergelijken. Het Shape Context Matching algoritme is zeer robuust in het herkennen van karakters. Zelfs wanneer de karakters behoorlijk vervormd zijn, kan het Shape Context Matching algoritme nog een goed resultaat genereren. Het grote nadeel van dit algoritme is het feit dat de karakters op voorhand gesegmenteerd moeten worden. Zonder deze segmentatie, kan het algoritme niet gebruikt worden bij het analyseren van CAPTCHA's.

Vervolgens werd het Distance Transform Matching algoritme voorgesteld. In tegenstelling tot het Shape Context Matching algoritme, kan deze techniek probleemloos vormen herkennen zonder het vooraf segmenteren van de afbeelding. Dit is een zeer nuttige eigenschap bij het analyseren van CAPTCHA's. Het Distance Transform Matching algoritme is afhankelijk van een set van referentiefabeeldingen. Deze afbeeldingen kunnen dan gezocht worden in de betreffende doelafbeelding, in dit geval een CAPTCHA.

Door het combineren van beide technieken, kan het lokaliseren en segmenteren van de afbeelding gebeuren door Distance Transform Matching. Nadien kunnen de verschillende segmenten geanalyseerd worden door het Shape Context Matching algoritme. Door deze combinatie te gebruiken, kunnen de meeste standaard CAPTCHA's gebroken worden.

Zoals besproken in hoofdstuk 8 zijn de complexere CAPTCHA's hier echter imuun voor. Deze CAPTCHA's worden zo opgesteld dat segmentatie van de afbeelding zeer moeilijk wordt. Door karakters tegen elkaar of zelfs over elkaar te plaatsen, wordt het Distance Transform Matching algoritme omzeild. Het analyseren door Shape Context Matching is dus niet meer mogelijk.

Over het algemeen zijn CAPTCHA's dus geen goede beveiliging tegen spambots. De nieuwere en complexere CAPTCHA's bieden meer weerstand, maar 100% veilig zijn ze nooit. De Google CAPTCHA is bijvoorbeeld onlangs gekraakt. Dit werd beschouwd als een van de beste CAPTCHA's op dit moment. Het is dus slechts een kwestie van tijd vooraleer een bepaalde CAPTCHA gebroken wordt.

Bibliografie

[1] www.captcha.net. The official captcha site.

[2] G. Mori and J. Malik. Recognizing objects in adversarial clutter – breaking a visual captcha, 2003.

[3] Shunji Mori, Hirobumi Nishida, and Hiromitsu Yamada. *Optical character recognition*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[4] S. Impedovo, L. Ottaviano, and S. Occhinegro. Optical character recognition-a survey. *IJPRAI*, 5:1–24, 1991.

[5] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004.

[6] M. Hagedoorn. Pattern matching using similarity measures, 2000.

[7] R. Veltkamp and M. Hagedoorn. State-of-the-art in shape matching. Technical Report UU-CS-1999-27, Utrecht University, the Netherlands, 1999.

[8] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts, 2001.

[9] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Dover Publications, July 1998.

[10] J. Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces.

[11] Jean Meinguet. Multivariate interpolation at arbitrary points made simple. *Zeitschrift fr Angewandte Mathematik und Physik (ZAMP)*, 30(2):292–304, May 2005.

[12] Abdul Ghafoor, Rao Iqbal, and Shoab Khan. Image matching using distance transform. pages 212–229. 2003.

[13] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. pages 127–133, 2003.

[14] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the hausdorff distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(9):850–863, 1993.

[15] F. John Canny. A Computational Approach to Edge Detection. 8(6):679–698, 1986.