KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Augmented Reality Bordspellen

Wouter Franken

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Mens-machine communicatie

Promotor:
Prof. dr. ir. Philip Dutre

Assessor:
TODO

Begeleider:
Ir. J. Baert

Academiejaar 2014 – 2015

Voorwoord

DANKWOORD

Wouter Franken

Inhoudsopgave

Samenvatting

ABSTRACT

Lijst van figuren en tabellen

Lijst van figuren

Lijst van tabellen

Lijst van afkortingen en symbolen

Afkortingen

Symbolen

Hoofdstuk 4

Legoblokdetectie op basis van CAD modellen

In het voorgaande hoofdstuk werd een naïeve methode aangebracht om legoblokken te detecteren. Deze methode worstelde echter met verschillende nadelen waardoor het niet mogelijk was om constructies van legoblokken te detecteren die bestond uit meerdere niveau's. Om dit soort constructies wel te kunnen detecteren moeten we op zoek naar een meer generiek algoritme dat op basis van alle geometrische informatie een legoblok kan detecteren (zodanig dat de blok kan gevonden worden in eender welke legoconstructie). Daarom wordt in dit hoofdstuk een algoritme behandeld dat op basis van features, geëxtraheerd uit CAD modellen, legoblokken kan detecteren in een videoframe.

Eerst schetsen we kort in sectie 4.1 het algoritme. Vervolgens bespreken we in sectie 4.2 welke evaluatiemethodes verder in dit hoofdstuk worden gebruikt. Verschillende feature types en classificatie methodes voor dit algoritme worden besproken en vergeleken in secties 4.3 en 4.4 respectievelijk. Tenslotte geven we in sectie 4.5 aan waarom deze methode niet kan worden gebruikt in een AR spel om legoblokken te detecteren.

4.1 Inleiding

CAD modellen zijn collecties van punten die de geometrie van een 3 dimensionaal object beschrijven, meestal worden ze gebruikt om objecten in een virtuele 3D wereld voor te stellen. Omdat zulk model alle geometrische informatie over een object bevat, kan deze informatie gebruikt worden om een object in een afbeelding te detecteren. Het grote voordeel is dan dat, aangezien dit model alle geometrische informatie bevat, het object in om het even welke pose kan worden gedetecteerd. Dit is de methode die werd aangebracht in [1].

Om CAD modellen te kunnen gebruiken voor object detectie moet eerst nuttig informatie uit deze modellen worden gehaald. In een afbeelding is echter slechts een deel van het object te zien (de afbeelding is namelijk 2D), daarom is het beter eerst afbeeldingen te genereren van de verschillende poses van het object. Hierna kunnen

deze afbeeldingen vergeleken worden met de testafbeelding en afhankelijk van een goede score blijkt welke pose het object in de afbeelding heeft.

Om te bepalen waar het object zich in de afbeelding bevindt wordt gebruik gemaakt van window sliding. Hierbij wordt een rechthoekig window geschoven over de afbeelding en op elke plaats worden de pixels in het window vergeleken met de pixels van de verschillende gegenereerde poses van het object. Om daarenboven ook de schaal van het object te bepalen wordt dit meermaals gedaan met een testafbeelding van verschillende groottes (mbv. een gaussiaanse piramide).

Het vergelijken van het window met de verschillende poses wordt gedaan door van beide features te berekenen en dan via een classificatie methode te vergelijken met elkaar. Hiervoor kunnen verschillende soorten features worden gebruikt en om aan te tonen waarom de keuze in het soort feature zo belangrijk is worden verschillende features in dit hoofdstuk vergeleken met elkaar. Verder worden ook twee verschillende classificatie methodes met elkaar vergeleken.

4.2 Evaluatiemethode

De evaluatie van de verschillende features gebeurt door eerst een classifier te trainen voor alle features en vervolgens de geleerde classifiers te gebruiken voor detectie.

Voor de **training** werden een 3000-tal positieve en 2946 negatieve samples gebruikt. De positieve samples zijn geconstrueerd door een set van 128 positieve afbeeldingen te transformeren op een willekeurig gekozen negatieve achtergrond (uit de negatieve samples). Deze transformaties helpen om een groter aantal aan positieve samples te genereren en om het effect van belichting en rotaties kleiner te maken. De set van 128 positieve afbeeldingen bestaat uit verschillende zichtpunten op een CAD model van een 2x2 legoblok en enkele foto's van zulke legoblok, allemaal op een witte achtergrond. De zichtpunten zijn gegenereerd van slechts een kwart van de legoblok

Parameter	Waarde
Training	
# classif. in cascade	20
Min. hit rate	0.999
Max. false alarm rate	0.5
Mode (Haar-like)	ALL
Detectie	
Schaalfactor	1.1
Min. # buren	15
Min. grootte	10x10 (px)
Max. grootte	100x100 (px)

Tabel 4.1: Parameters gebruikt tijdens training en detectie van features met cascade classificatie.

Parameter	Waarde
Training	
Padding (HOG)	0x0 (px)
Wind. stride (HOG)	8x8 (px)
Wind. size (HOG)	64x64 (px)
Detectie	
Schaalfactor	1.1
Hit threshold	0.4
Padding (HOG)	0x0 (px)
Wind. stride (HOG)	8x8 (px)
Wind. size (HOG)	64x64 (px)

Tabel 4.2: Parameters gebruikt tijdens training en detectie van features met SVM.

omdat de legoblok symmetrisch is. De foto's helpen de invloed van belichting op de classifier te minimaliseren aangezien ze werden getrokken in een meer realistische belichting. De parameters die tijdens de training werden gebruikt zijn aangegeven in tabellen 4.1 en 4.2.

De features werden steeds getraind door een cascade classifier te modelleren (zie sectie 4.4.1), dit om geen invloed op de resultaten te krijgen door een andere keuze in classificatie methode. Bij de vergelijking tussen de classificatie methodes werd steevast voor de HOG features gekozen om dezelfde reden (zie sectie 4.3.3).

De **detectie** gebeurt op twaalf afbeeldingen van een legoblok in verschillende poses. Dit geeft een idee over hoe robuust de features en classificatie methodes zijn tegen wijzigingen in pose. De parameters die gebruikt werden tijdens de detectie zijn ook aangegeven in tabellen 4.1 en 4.2.

Bij het vergelijken van features zal eerst naar performantie en robuustheid worden gekeken. Vervolgens wordt aangetoond wat de invloed is van de schaalparameter op de detectie van een legoblok. De vergelijking tussen classificatie methodes focust zich eerst ook op performantie en robuustheid, vervolgens wordt de impact van de *hit threshold* parameter (enkel voor SVM, zie sectie 4.4.2) bekeken. Voor de experimenten die focussen op performantie werd de detectie 10x uitgevoerd op alle twaalf testafbeeldingen. Omdat de training vaak een heel stuk meer tijd vraagt werd deze telkens maar één maal uitgevoerd om toch een groffe indicatie te geven van hoe lang zo een training duurt.

De experimenten werden uitgevoerd op een **machine** met een Intel Core 2 Duo 2,4 GHz processor, een NVIDIA GeForce 320M 256 MB grafische kaart, 8 Gb RAM en Mac OS X 10.10.2. Alle implementaties maken gebruik van OpenCV 2.4.9 (zowel training als detectie) en implementaties met SVM maken gebruik van SVMlight [6].

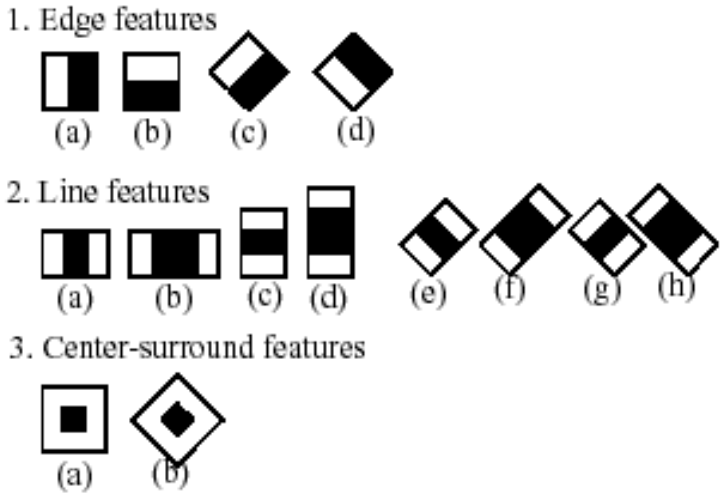
4.3 Features

In deze sectie worden drie verschillende soorten features besproken en met elkaar vergeleken: Haar-like [11], Multiscale Block Local Binary Patterns [7] (MB-LBP) en Histogram of Oriented Gradients [3] (HOG). Tenslotte wordt ook een parts-based feature besproken (gebaseerd op HOG), dat gebruikt werd in [1], en waarom deze niet is opgenomen in de vergelijking [4].

4.3.1 Haar-like

Bij de berekening van een Haar-like feature vector wordt het verschil genomen van de som van pixels tussen verschillende rechthoekige regio's. Oorspronkelijk werden vier soorten features gebruikt: twee die elk bestonden uit twee rechthoekige gebieden, één soort met drie rechthoekige gebieden en één soort met vier rechthoekige gebieden [11]. Later werden extra features toegevoegd tot de uiteindelijke featureset bestond uit 14 soorten features [8], zie figuur 4.1.

Om goede features te vinden wordt eerst een enorme hoeveelheid aan features aangemaakt. Aangezien dit aantal features een heel stuk hoger is dan het aantal pixels in het window moet een efficiënte methode gebruikt worden om de juiste

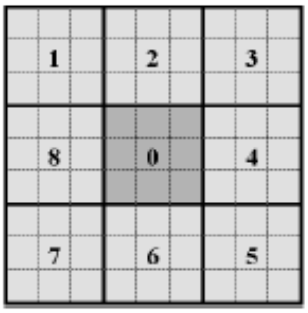


Figuur 4.1: Uitgebreide Haar-like featureset. Het verschil wordt bepaald door de som te nemen van pixels in de witte regio's en af te trekken van de som van pixels in de zwarte regio's.

features te selecteren. Hiervoor wordt AdaBoost gebruikt, een methode die origineel wordt gebruikt om een classifier sterker te maken. AdaBoost selecteert telkens die feature die de positieve en negatieve samples scheidt zodanig dat de fout zo klein mogelijk is [11]. Dit impliceert dat elke feature ook een threshold bevat die aangeeft wanneer deze feature een detectiewindow juist of onjuist classificeert. De uiteindelijke feature vector van het volledige detectiewindow is de set van de geselecteerde features.

4.3.2 MB-LBP

Een MB-LBP feature is een uitbreiding op de originele LBP feature waarin elke pixel werd vergeleken met zijn acht buren. Deze acht buren werden steevast in dezelfde volgorde overlopen en telkens een buur een grotere intensiteit had dan de pixel zelf



Figuur 4.2: Een 9x9 MB-LBP operator: elk blok bestaat uit 9 pixels.

werd een '1' genoteerd, in de andere gevallen een '0'. Zo wordt voor elke pixel een getal verkregen van acht bits dat de gradiënt van de pixel voorstelt in de verschillende pixels [9].

Hier wordt de MB-LBP feature gebruikt in plaats van de oorspronkelijke LBP omdat deze meer robuust is. In de MB-LBP variant worden, in plaats van een pixel met zijn buren te vergelijken, blokken van pixels met hun buren vergeleken (zie figuur 4.2). Hierdoor wordt is de feature meer robuust en encodeert het bovendien, naast microstructuren, ook macrostructuren in een afbeelding [7].

Om de feature vector van een volledig window te berekenen wordt het window in cellen opgesplitst en vervolgens histogrammen van de MB-LBP features binnen een cel. De histogrammen van elke cel worden ten slotte geconcateneerd om de feature vector te bekomen.

4.3.3 HOG

Bij een HOG feature worden eerst gradiënten van alle pixels in een window berekend. Vervolgens wordt het window onderverdeeld in cellen en wordt voor elke cel een histogram van de gradiënten gemaakt. Om problemen met belichting en contrast te voorkomen worden cellen gecombineerd tot blokken waarover deze histogrammen worden genormaliseerd. Ten slotte worden alle histogrammen geconcateneerd (net als bij MB-LBP) om een feature vector van het window te bekomen [3].

4.3.4 Deformed parts-based

Deformed parts-based modellen is een type feature dat bestaat uit twee onderdelen: een HOG feature van de volledige window (root filter) en aantal kleinere HOG features van een subwindow (part filters). Deze verschillende part filters worden gedefinieerd door een anker punt ten opzichte van de plaats van de root filter en een functie die alle mogelijke plaatsen van deze part filter beschrijft relatief ten opzichte van het anker punt. De score van de hypothese dat een window het te detecteren object bevat wordt berekend door de sommatie te nemen van het verschil tussen de score van de filter (HOG feature) en een kost voor de vervorming van de filter. Dit model maakt het HOG feature een stuk meer flexibel voor vervormingen of andere poses van een object [4].

Een groot nadeel van deze methode is dat ze nogal wat rekenwerk vraagt om uit te voeren en dus ongeschikt is voor gebruik in AR. Dit is zelfs eerder aangetoond in andere state-of-the-art papers die de performantie hebben kunnen verhogen. Hoewel bijvoorbeeld in paper [12] wordt aangegeven dat 40 FPS haalbaar is, is dit slechts na parallellisatie en bovendien op een desktop computer. Zulke resultaten zullen dus (voorlopig) niet gehaald kunnen worden op een mobiel apparaat om te gebruiken voor AR. Wegens tijdsbeperkingen hebben we dit niet aangetoond.

4.3.5 Vergelijking

In deze sectie worden de eerste drie besproken feature-types met elkaar vergeleken. Ze worden eerst vergeleken qua performantie en robuustheid. Tenslotte wordt

onderzocht wat de impact is op de performantie en robuustheid bij het wijzigen van de schaalfactor.

Performantie De vergelijking van performantie tussen verschillende soorten features bestaat uit twee onderdelen: enerzijds de snelheid bij training van de classifier en anderzijds de snelheid bij detectie.

Tabel 4.3 toont dat een classifier op basis van HOG features sneller kan worden getraind dan een classifier op basis van LBP features en dat deze laatste sneller kan worden getraind dan een classifier op basis van Haar-like features. We merken vooral op dat er een groot verschil is tussen de tijd nodig om een HOG classifier te trainen en tijd nodig om een LBP of Haar-like classifier te trainen. Dit valt als volgt te verklaren: bij het trainen van een Haar-like classifier wordt eerst voor elke window een enorme hoeveelheid aan features gegenereerd die vervolgens wordt uitgeselecteerd met AdaBoost [5]. Deze hoeveelheid aan features is vele malen groter dan het aantal pixels in het window, in tegenstelling tot MB-LBP en HOG waarbij voor elk window enkel een histogram wordt opgesteld (in feite worden voor MB-LBP meerdere histogrammen geconcateneerd).

Uit de snelheid in detectie kunnen we vooral opmerken dat de detectie van MB-LBP een stuk sneller gaat dan detectie van Haar-like en HOG. De reden hiervoor is simpelweg dat bij MB-LBP gewoon integer waarden worden vergeleken terwijl bij Haar-like en HOG features floating point getallen moeten worden vergeleken.

Robuustheid Uit de resultaten van figuur 4.3 kan worden afgeleid dat detectie met behulp van HOG, getraind via een SVM de beste resultaten geeft: slechts op één afbeelding uit de testset was een vals positief resultaat aanwezig en op 2 afbeeldingen uit de testset na werd de legoblok correct gevonden door de classifier.

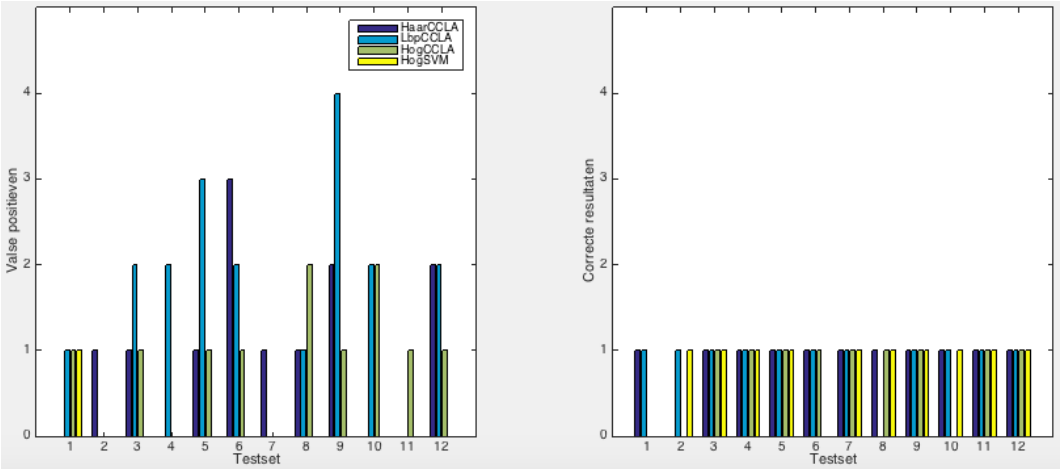
Figuur 4.3 toont dat de HOG features erg slechte resultaten vertoont: slechts 4/12 keer wordt de legoblok werkelijk gevonden in de frame.

4.4 Classificatie methodes

In deze sectie worden de volgende classificatie methodes besproken: Cascade Classifiers [11] (CCLA) en Support Vector Machines [6] (SVM). Vervolgens worden beide methodes vergeleken op vlak van robuustheid en performantie. Ten slotte wordt

	Gemiddelde performantie	
Feature	Training	Detectie (ms)
Haar-like	$>$ 2 dagen	TODO
MB-LBP	TODO	43.71
HOG	TODO	189.36

Tabel 4.3: Feature performantie.



FIGUUR 4.3: Feature robuustheid.

de impact bekeken van de *scale factor* parameter op robuustheid en performantie CCLA en hetzelfde wordt gedaan met de *hit threshold* parameter voor SVM.

4.4.1 CCLA

Bij een **CCLA** wordt een classifier getraind door een aantal feature classifiers te combineren. Elk van deze classifiers bestaat uit een aantal features op basis waarvan elke subwindow van een frame wordt geclassificeerd. Zulke cascade classifier werkt door eerst elke subwindow van de frame te laten evalueren door de eerste classifier, indien de eerste classifier de subwindow aanvaardt triggert dit een evaluatie door de volgende classifier in de cascade. Enkel wanneer een window door alle classifiers is aanvaard, kan er vanuit worden gegaan dat dit window het te detecteren object bevat. Door een cascade van classifiers te gebruiken moet elke classifier van de cascade niet al te sterk zijn waardoor negatieve windows al sneller kunnen worden afgevoerd, wat de detectie des te sneller maakt.

4.4.2 SVM

Een **SVM** classificeert op een andere manier: elk subwindow wordt voorgesteld als een p-dimensionele vector in een p-dimensionele ruimte. Met behulp van (p-1)-dimensionele hyperplane worden de subwindows gesplitst in twee groepen: de ene groep wordt aanvaard, de andere niet. Deze lineaire classifier was het oorspronkelijke algoritme [10], later werd ook een methode ontwikkelt voor non-lineaire classificatie [2].

4.4.3 Vergelijking

Robuustheid Figuur 4.4 toont de robuustheid van de twee classifier methodes. Merk op dat de grafiek met valse positieven is weggelaten omdat deze in beide

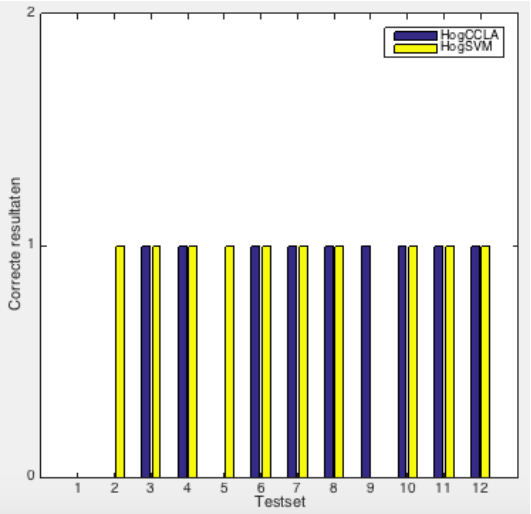
gevallen overal nul was. Er is duidelijk weinig verschil tussen de twee methodes qua robuustheid: de SVM methode detecteert één legoblokje meer.

Performantie Tabel 4.4 toont een vergelijking van de performantie tijdens de training en de detectie. Dit toont aan dat de snelheid van CCLA hoger ligt bij detectie maar niet bij training. Dat de detectie langer duurt bij een SVM is logisch aangezien het een lineaire classifier is (geen cascade die de snelheid verhoogt). De training snelheid ligt lager bij een SVM omdat deze geen cascade moet opstellen, terwijl CCLA dit wel moet doen.

Schaalfactor De schaalfactor is een parameter die bepaalt hoe sterk de schaal van het window wordt gewijzigd tussen een minimum en maximum grootte. Dus wanneer deze kleiner wordt zullen normaal gezien meer iteraties nodig zijn, terwijl een nauwkeurigere detectie plaatsvindt.

Om de impact van de schaalfactor te bepalen, wordt de performantie en robuustheid van de HOG feature vergeleken wanneer de schaalfactor 1.01, 1.05, 1.10, 1.15, 1.20 en 1.25 bedraagt. Uit figuur 4.5 is het duidelijk dat aan de verwachtingen voldaan is: de performantie is hoger (het aantal ms daalt) bij een hogere schaalfactor, terwijl het aantal correcte detecties daalt. Het aantal valse positieven werd ook gemeten maar deze was 1 bij schaalfactor 1.01 en 0 voor de rest.

Opmerkelijk is echter dat het aantal correcte poses hoger ligt bij SVM dan bij CCLA. Maar hoewel de detectie beter is laat de performantie echter te wensen over. De oorzaken hiervoor kunnen als volgt verklaard worden: Een SVM is een lineaire classifier die vergeleken kan worden met één enkele classifier uit een cascade. Om een voldoende goede detectie te behalen moet een SVM dus erg correct kunnen classificeren, dit vergt meer tijd aangezien een cascade juist is ontwikkeld om efficiënter



Figuur 4.4: Robuustheid classifier methodes.

	Gemiddelde performantie	
Feature	Training	Detectie (ms)
CCLA	TODO	189.36
SVM	1m55s	244.94

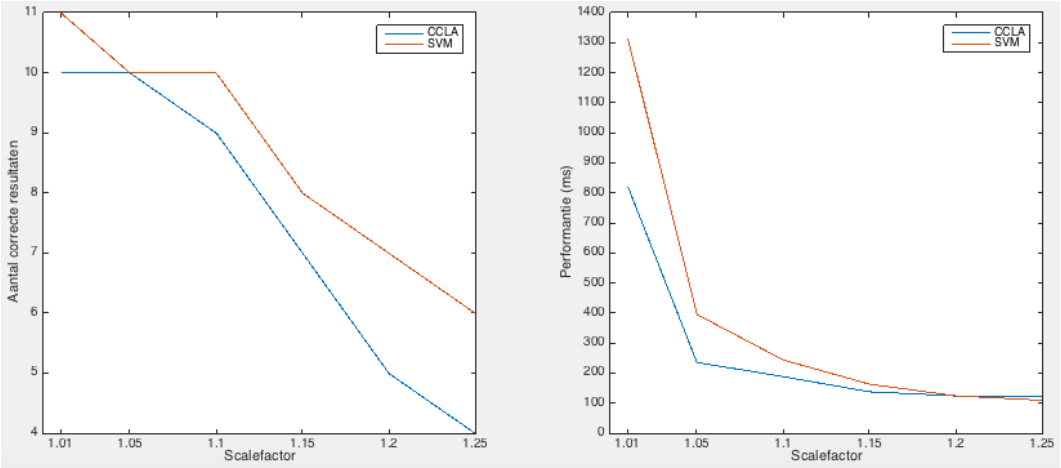
Tabel 4.4: Performantie classifier methodes.

te zijn dan een enkele classifier. Aangezien de SVM een hogere robuustheid haalt is het dan ook logisch dat de performantie hoger is.

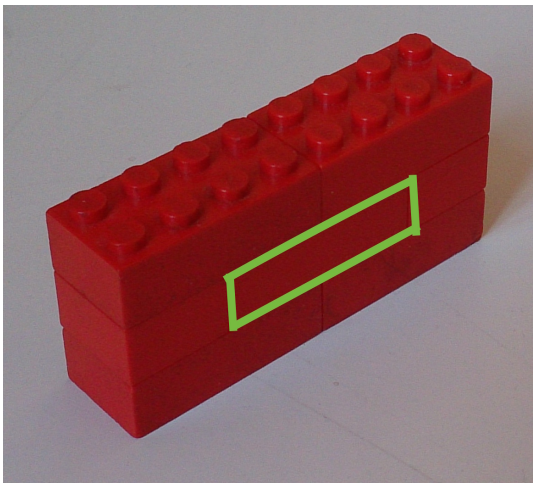
4.5 Discussie / Besluit

In dit hoofdstuk werd onderzocht welke feature types en classifier methodes beter werken voor detectie van legoblokken. Zo is aangetoond dat HOG de meer robuuste feature type is en dat MB-LBP de meest performante van de drie is. Verder werd aangegeven dat een vierde type feature (parts-based deformable modellen) te traag is om nuttig te gebruiken in een AR applicatie. Ten slotte toonde de vergelijking tussen classifier methodes aan dat detectie met een SVM classifier vaak trager is maar tegelijk ook meer robuust. De methode die in dit hoofdstuk werd onderzocht is echter niet nuttig om te gebruiken in een AR spel om legoblokken te detecteren en hier is waarom:

Hoewel het wel mogelijk is om een enkele legoblok te detecteren is het erg moeilijk om in een constructie van legoblokken de blokken afzonderlijk te detecteren. Dit resulteerde telkens in erg slechte resultaten, zelfs wanneer het beste type feature en de beste classificatie methode werd gekozen. Dit wordt hoofdzakelijk veroorzaakt door de enorme hoeveelheid aan occlusie die een legoblok in een constructie grotendeels



Figuur 4.5: Impact schaalfactor op performantie en robuustheid van HOG feature.



Figuur 4.6: Een voorbeeld van een legoblokconstructie waarin het groen omlijnde vlak het enige is wat we van die legoblok zien.

verbergt. Hierdoor is slechts een erg beperkt deel van de legoblok (in sommige gevallen slechts één vlak, zie figuur 4.6) zichtbaar wat het erg moeilijk maakt om met features een legoblok te detecteren. Zelfs indien op voorhand een optimale pose zou worden gekozen kan voor sommige legoblokken enkel één vlak zichtbaar zijn in de afbeelding.

Deze redenering leidt tot een volgend resultaat: het is erg moeilijk, zo niet onmogelijk, om in een legoconstructie alle legoblokken apart te detecteren. Daarom moeten we proberen om, in plaats van de blokken individueel te proberen detecteren, ze als groter geheel te detecteren. Zulke legoconstructie kunnen we zien als een flexibele legoblok die eender welke hoogte, diepte en breedte kan hebben. Dat is exact waarvoor parts-based deformable modellen voor worden gebruikt maar zoals aangehaald in sectie 4.3.4 is deze methode veel te traag om te gebruiken in AR. Dat is de reden waarom in de volgende sectie een sneller alternatief wordt voorgesteld dat wel gebruik maakt van het idee dat lego constructies moeten gedetecteerd worden in plaats van individuele legoblokken. Hierbij wordt echter wel afgestapt van het idee om CAD modellen te gebruiken.

Bijlagen

Bibliografie

[1] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3762–3769. IEEE, 2014.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[5] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[6] T. Joachims. Svmlight: Support vector machine. *SVM-Light Support Vector Machine http://svmlight. joachims. org/, University of Dortmund*, 19(4), 1999.

[7] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Z. Li. Learning multi-scale block local binary patterns for face recognition. In *Advances in Biometrics*, pages 828–837. Springer, 2007.

[8] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900. IEEE, 2002.

[9] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[10] V. Vapnik. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963.

[11] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[12] J. Yan, Z. Lei, L. Wen, and S. Z. Li. The fastest deformable part model for object detection. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2497–2504. IEEE, 2014.

Fiche masterproef

Student: Wouter Franken

Titel: Augmented Reality Bordspellen

Engelse titel: Augmented Reality Boardgames

UDC: 681.3

Korte inhoud:
Hier komt een heel bondig abstract van hooguit 500 woorden. LaTeX commando's mogen hier gebruikt worden. Blanco lijnen (of het commando) zijn wel niet toegelaten!

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Mens-machine communicatie

Promotor: Prof. dr. ir. Philip Dutre

Assessor: TODO

Begeleider: Ir. J. Baert