TECHNISCHE UNIVERSITÄT DRESDEN

BACHELORARBEIT

Reduktion von Ausreißern bei Korrespondenzpunkten für die Stereorektifizierung

Burkhardt, Alexander

Matrikelnummer 3340703

Institution (TUD, Fakultät Informatik, Künstliche Intelligenz, Computervision)

Hochschullehrer: Holger Heidrich

Termin der Abgabe: Dresden,07.10.2014

TECHNISCHE UNIVERSITÄT DRESDEN

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

1 AUFGABENSTELLUNG

Bei Stereoabbildungen mit unkalibrierten Kameras und insbesondere bei Freihand-Stereoaufnahmen, wird eine Rektifizierung gebraucht. D.h. Die beiden Bilder werden mit je einer Homographie so transformiert, dass korrespondierende Punkte auf der gleich Bildzeile liegen. Um diese Homographie zu finden, müssen sehr genau genügend korrespondierende Punkte gefunden werden, die nicht in einer (3D-)Ebene liegen. Für praktische Anwendungen sollte der gesamte Algorithmus möglichst schnell sein.

In der Literatur gibt es zwei interessante Algorithmen für diese Anwendung:

[ORSA] verwendet den „a contrario“ RANSAC Algorithmus (ORSA), SIFT-Punkte und „Quasi-Euclidean uncalibrated epipolar rectification“ während

[SCV] fehlerhafte Korrespondenzen mittels Kosegmentierungen aussortiert und SVM und Wald's „Sequential Probability Ratio Test“ zu Beschleunigung nutzt.

In der Arbeit sollen beide Methoden verglichen und ihre Vorteile zu einem Algorithmus vereinigt werden.

2 ORSA/RANSAC

2.1 RANSAC THEORIE

Der RANSAC Algorithmus wurde 1981 von Martin A. Fischer und Robert C. Bolles auf der „Communications of the ACM“ offiziell vorgestellt. [RANSAC WIKI] Der RANSAC Algorithmus (RANdom SAmple Consensus) ist ein sehr robuster Algorithmus zur Schätzung eines Modells aus einer Reihe von Messdaten, die durch Ausreißer und grober Fehler verrauscht sind. Zur Verwendung von RANSAC ist immer ein überbestimmtes Modell notwendig, das heißt es müssen mehr Messdaten vorliegen als zur Berechnung der Modellparameter zwingend notwendig sind. Bei diesem Verfahren werden zu erst so viel Messdaten zufällig ausgewählt, wie zur Berechnung der Modellparameter notwendig sind. Im zweiten Schritt werden die Modellparameter errechnet und der Abstand aller Punkte zum Modell errechnet. Zuletzt werden alle Punkte, deren Abstand zum Modell kleiner als eine festgesetzte Grenze ist, in das sogenannte Consensus Set gegeben. Das gesamte Verfahren wird mehrfach wiederholt und zum Schluss das größte Consensus Set ausgewählt und auf dessen Basis, die Modellparameter berechnet. Im vorliegend Quellcode werden im RANSAC-Modus einige Abwandlungen vom ursprünglichen Verfahren angewendet. Zum einen gibt es keinen

festgelegten maximalen Abstand vom Modell. Anstelle dessen wird mit Hilfe des NFA(number of false alarms), der im ORSA-Abschnitt genauer erklärt wird, der maximale Abstand vom Modell für jeden Durchlauf errechnet. Zum anderen wird nicht der größte Consensus Set ausgewählt, sondern der Consensus Set mit dem niedrigsten zugehörigen NFA-Grenzwert.

2.2 ORSA THEORIE

Der ORSA Algorithmus [ORSA] wurde von Lionel Moisan, Pascal Monasse und Zhongwei Tang programmiert und beruht auf dem RANSAC Algorithmus[RANSAC]. Die Idee des ORSA ist, dass sobald eine gute Fundamental Matrix gefunden wurde oder 90% der Iterationen durchlaufen wurden, in allen weiteren Iterationen nur noch Messdaten aus den Inlinern der besten Fundamental Matrix für die Berechnung einer neuen Fundamental Matrix gewählt werden.

2.3 ORSA/RANSAC ALGORITHMUS

Der Algorithmus verfügt über 4 verschiedene Modi. Der erste Modus ist ein rein deterministischer Modus, der alle möglichen Kombinationen von Samples durchprobiert. Er ist folglich sehr rechenaufwendig und nur anwendbar, wenn lediglich eine kleine Zahl von Matches verfügbar ist. Da er deterministisch ist, liefert er immer die bestmöglichste Fundamental Matrix. Der zweite Modus ist ein reiner RANSAC-Modus. Er wählt zufällig eine Sample Konfiguration, berechnet die möglichen Fundamental Matrixen und bewertet sie. Dies wiederholt er mehrfach und gibt die beste Fundamental Matrix zurück. Der dritte Modus ist der ORSA-Modus. Er funktioniert wie der RANSAC-Modus, wählt aber ab einem bestimmten Zeitpunkt nur noch Samples aus den Inlinern der bisher besten Fundamental Matrix [ORSA THEORIE]. Sollte der Inputwert stop auf true gesetzt sein, bricht der Algorithmus ab, sobald eine ausreichend gute Fundamental Matrix gefunden wurde. Der letzte Modus ist ein automatischer Modus. Sollte die Anzahl der Iterationen größer der Anzahl der möglichen Sample Konfigurationen sein, funktioniert er wie der deterministische Modus. Ansonsten arbeitet er wie der ORSA-Modus. Der Algorithmus enthält die Inputwerte match, t, verb , mode, stop, logalpha0, inliners und errorMax. Match ist ein Zeiger auf einen Vektor von Matches, der die zu untersuchenden Koordinatenpaare enthält. Die Boolean-Werte verb und stop dienen als flags, wobei verb den verbose mode, also den Ausgabemodus, aktiviert und stop wie oben beschrieben für den ORSA-Modus relevant ist. Die Integer t gibt die maximale Anzahl der Iterationen an und die Integer mode den Modus in dem der Algorithmus läuft. logalpha0 ist ein Float-Wert, der den Logarithmus der Wahrscheinlichkeit eines Fehlers von einem Pixel angibt. Die letzten beiden Variablen Inliers und errorMax dienen als zusätzliche Rückgabewerte. Inliers ist ein Zeiger auf einen Vektor

von size_t, der den Indexwert der Inliner in match zurückgibt und errorMax ist ein Zeiger auf einen Float-Wert, der die Quadratwurzel des höchsten Fehlers des Inliners beinhaltet. Der Rückgabewert der Funktion ist eine 3x3 Matrix von Float-Werten, der die beste Fundamental Matrix zurückgibt. Zur Berechnung einer Fundamental Matrix werden 7 Matches benötigt. Aus ihnen generiert der Algorithmus über epipolare Transformation eine oder drei Fundamental Matrixen. Für jede Fundamental Matrix wird nun der Fehler für jedes Match berechnet und die Matches aufsteigend nach Fehler sortiert. Aus diesem sortierten Vektor von Fehlerwerten mit Index des zugehörigen Matches wird nun die NFA berechnet(die ersten 7 Werte werden ausgelassen, da sie zu den Samplen gehören). Die NFA(number of false alarms) berechnet sich als dem Input logalpha0, dem dekadischen Logarithmus(im weiteren log10 genannt) des Fehlerwertes, loge0 den log10 von 3 mal der Anzahl der Matches minus 7 (Anzahl der Samples), logcn(i) dem log10 des Binomialkoeffizienten (i, Anzahl der Matches) und logc7(i) dem log10 des Binomialkoeffizienten (7, i). [ABB.1] Das Match mit dem niedrigsten NFA, sowie alle Matches mit niedrigerem Fehlerwert als dieses, sind die Inliner. Alle anderen Matches stellen Outliner dar. Sollte der so errechnete minNFA für diese Fundamental Matrix kleiner sein als der bisher kleinste bekannte minNFA, wird diese Fundamental Matrix zur neuen besten Matrix.

$$NFA = \log_{10}(3*(n-7)) + (logalpha0 + 0{,}5*\log_{10}(error[i]))*(i-6) + \log_{10}\binom{i+1}{n} + \log_{10}\binom{7}{i+1}$$

Abb. 1 NFA Formel

3 SCV

Der SCV-Algorithmus von Jan Čech, Jiří Matas und Michal Perd'och [SCV] ist ebenfalls ein Algorithmus zu Verifikation von Bildpunkt Matches . Er wurde speziell für extrem stark verrauschte Bildpaare entwickelt. In ersten Schritt führt der Algorithmus eine Grown Funktion aus, welche aus den bisher bekannten Matches versucht weitere neue Matches zu erzeugen. Im nächsten Schritt wird eine Support Vector Maschine benutzt. Diese erhält vom der Grow-Funktion die durchschnittliche Anzahl an neuen Matches pro Grown-Step, den durchschnittlichen Korrelationswert der neuen Matches und die Wahrscheinlichkeit dass ein Punkt des linken Bildes auf einen Punkt im rechten Bild gematched wird, der bereits durch einen anderen Punkt im linken Bild besetzt ist. Des weiteren erhält er noch ein externes Modell und eine Sift ratio. Die Werte der Grown-Funktion werden mit Hilfe der distance ratio sr des nächsten und zweitnächsten SIFT discriptor optimiert. Er generiert daraus eine skalare Qualität mit deren Hilfe die likelihood Wahrscheinlichkeiten für korrekte und inkorrekte Correspondence Klassen leichter abgeschätzt werden kann. Im letzten Schritt wird

ein Wald's Sequential Probability Ratio Test(SPRT) auf die Likelihood Wahrscheinlichkeiten angewendet. Sollte der SPRT nicht terminieren, so wird der Algorithmus mit einer höheren Zahl an Grow-Steps wiederholt.

4 KOMBINIERTER ALGORITHMUS

4.1 IDEE

Als Basis habe ich den ORSA/RANSAC Algorithmus[ORSA] gewählt. Um ihn zu optimieren, habe ich versucht den Algorithmus mit Hilfe von SPRT zu beschleunigen. Zudem wurde eine zusätzliche Grown-Funktion eingefügt, die an die Grown-Funktion des SCV-Algorithmus angelehnt ist. Sobald der ORSA/RANSAC Algorithmus eine akzeptable Fundamental Matrix gefunden hat, werden deren Werte verwendet um bereits bei der Fehlerberechnung einer neuen Fundamental Matrix mit SPRT einen vorzeitigen Abbruch zu ermöglichen, sollten zu viele Fehlerwerte höher sein als der maximale Fehler der Inliner der bisher besten Fundamental Matrix. Falls ein solcher vorzeitiger Abbruch eintritt, wird die weitere Fehlerberechnung sowie die Berechnung des NFA ausgelassen und die Fundamental Matrix verworfen. Diese Optimierung ist für alle Modi des ORSA/RANSAC Algorithmus verwendbar. Vor allem wenn viele Matches verfügbar sind, sollte diese Optimierung eine deutliche Geschwindigkeitsverbesserung bei sehr geringer Fehlererhöhung erreichen. Die neu eingefügte Grow-Funktion kann dem ORSA/RANSAC Algorithmus vorgeschaltet werden und erhöht die Anzahl der verfügbaren Matches.

4.2 WALD'S SPRT THEORIE

$$\frac{\ln\left(\frac{\beta}{1-\alpha}\right)}{\ln\left(\frac{\tau_1(1-\tau_0)}{\tau_0(1-\tau_1)}\right)}+i*\frac{\ln\left(\frac{1-\tau_0}{1-\tau_1}\right)}{\ln\left(\frac{\tau_1(1-\tau_0)}{\tau_0(1-\tau_1)}\right)}<y<\frac{\ln\left(\frac{1-\beta}{\alpha}\right)}{\ln\left(\frac{\tau_1(1-\tau_0)}{\tau_0(1-\tau_1)}\right)}+i*\frac{\ln\left(\frac{1-\tau_0}{1-\tau_1}\right)}{\ln\left(\frac{\tau_1(1-\tau_0)}{\tau_0(1-\tau_1)}\right)}$$

Abb. 2 Wald's SPRT Fortsetzungsbereich

1942 von Abraham Wald [WALD] in den USA entwickelt, zielt der SPRT Algorithmus darauf ab, nach möglichst wenigen Stichproben zu entscheiden, ob eine Nullhypothese H_0 oder die Alternativhypothese H_1 für den Test akzeptiert wird. Für H_0 wird eine höchste Irrtumswahrscheinlichkeit von alpha und für H_1 eine maximale Irrtumswahrscheinlichkeit beta festgelegt. Hierfür errechnet der Algorithmus den Likelihood Quotienten L und die Entscheidungsgrenzen A und B. Sollte L größer-gleich A sein wird H_1 akzeptiert und der

Algorithmus terminiert. Ist L kleiner-gleich B wird H_0 akzeptiert und der Algorithmus terminiert. Sollte L größer B und kleiner A sein, wird der Algorithmus fortgesetzt und weitere Stichproben zur Entscheidung benötigt. Grundsätzlich gilt das A größer als B ist. Durch Umformung kann man den Term [Abb.2] für den Fortsetzungsbereich bekommen. Die Anzahl an negativen Stichproben wird hier durch y angegeben. τ_0 ist der maximale Anteil der negativen Stichproben für die Akzeptanz von H_0. τ_1 ist der minimale Anteil an negativen Stichproben für die Akzeptanz von H_1. α und β sind die oben bereits erwähnten Irrtumswahrscheinlichkeiten für H_0 und H_1. i ist die Größe der bisher berücksichtigten Stichprobe. Sollte y kleiner den linken Term sein, wird H_0 akzeptiert. Ist y jedoch größer dem rechten Term wird H_1 akzeptiert.

4.3 SPRT OPTIMIERUNG

Der SPRT Algorithmus benötigt einen Grenzwert für das Verwerfen und einen Grenzwert für das Akzeptieren der Fundamental Matrix. Des weiteren noch die falsepositiv und falsenegativ Wahrscheinlichkeiten alpha und beta und einen Grenzwert für den maximalen akzeptablen Fehlerwert. Zu Beginn des ORSA/RANSAC Algorithmus sind nur alpha und beta bekannt. Die weiteren Variablen werden erst nach dem Errechnen der ersten akzeptablen Fundamental Matrix gesetzt. Infolgedessen ist der Durchlauf für die erste Fundamental Matrix identisch zum normalen ORSA/RANSAC. Nachdem die Fundamental Matrix ermittelt wurde, wird der maxerror der Inliner dieser Matrix verwendet um zu entscheiden, ob ein Match wahrscheinlich ein Inliner oder Outliner ist. Sollte der Fehler eines Matches unter einer Fundamental Matrix größer als maxerror sein, wird dieses Match vom SPRT als Outliner betrachtet. Der Anteil von Inlinern an den gesamten Matches unter der bisher besten Fundamental Matrix, sowie alpha und beta werden an die Funktion calSPRT übergeben. Diese Funktion berechnet daraus den festen, von der Größe der Stichprobe unabhängigen Teil der Entscheidungsgrenze B, den festen Teil der Entscheidungsgrenze für A und den variablen, von der Größe der Stichprobe abhängigen Teil, der Entscheidungsgrenzen A und B. Letzterer ist für A und B gleich. Da nun alle benötigten Werte gesetzt sind, wird für alle weiteren Fundamental Matrixen nicht mehr die ursprüngliche ORSA/RANSAC Funktion sortErrors verwendet, sondern die modifizierte Funktion sprtSortErrors. Diese Funktion erhält neben den Inputwerten der sortErrors Funktion noch den double Wert maxerror, der den maximalen Fehler eines vermuteten Inliners übergibt, sowie einen Vektor von double Werten varSPRT, der die von calSPRT berechneten Werte enthält, als Inputwerte. Sie hat zudem einen Boolean Rückgabewert, der angibt ob die Fundamental Matrix positiv entschieden wurde. Die sprtSortErrors Funktion berechnet jetzt für jedes Match den zugehörigen Fehler. Sollte der Fehler größer als maxerror sein, wird der Outliner counter outlnum um eins erhöht und geprüft, ob outlnum größer-gleich A ist. Sollte dies der Fall sein, wird sprtSortErrors mit

dem Rückgabewert false beendet. Sollte der Fehler kleiner-gleich maxerror sein, wird überprüft ob outlnum kleiner-gleich B ist. Ist dies der Fall, werden die Fehler für alle verbleibenden Matches berechnet, sortiert und sprtSortErrors mit dem Rückgabewert true beendet. Sollten nach der Berechnung der Fehler aller Matches outlnum immer noch größer B sein, wird die Funktion mit dem Rückgabewert false beendet. Wurde die Funktion sprtSortErrors mit dem Rückgabewert false beendet, wird die Fundamental Matrix verworfen, ansonsten wird der minNFA der Fundamental Matrix berechnet. Ist der minNFA besser als der bisher beste, wird diese Fundamental Matrix zur neuen besten Matrix und der maxerror und die Entscheidungsgrenzen A und B auf Basis dieser Matrix neu berechnet.

4.4 GROW FUNKTION

Die Grown-Funktion basiert auf der Grow-Funktion des SCV-Algorithmus und muss vor dem eigentlichen ORSA/RANSAC Algorithmus angewendet werden. Die Funktion hat als Inputwerte die Grayscale Images des linken und rechten Bildes, sowie den Vektor der bisherigen Matches, die Fenstergröße w, die Anzahl der Grow-Steps my, den Korrelation Threshold c_thr und den Mindestabstand zwischen der besten und zweitbesten Korrelation epsilon. Der Rückgabewert der Funktion ist der neue Vektor der Matches. Zuerst wird die Korrelation jedes ursprünglichen Matchs berechnet und die Matches nach dem Korrelationswert geordnet. Da die Matches des ORSA/RANSAC keinen local affine Frame haben muss dieser erst berechnet werden. Für jedes Match wird daher mithilfe eines linearen Gleichungssystems und unter Berücksichtigung des in dem Vektor vorhergehenden und nachfolgenden Matchs ein solcher local affine Frame berechnet. Solange die maximale Anzahl von Grow-Steps noch nicht erreicht ist und noch nicht alle Matches verarbeitet wurden, wählt der Algorithmus das beste noch nicht verwendete Match aus. Für jeden der vier Nachbarn des Punktes im linken Bild wird nun der Punkte im rechten Bild ausgerechnet, der die beste Korrelation hat und der zweitbeste Korrelationswert gespeichert. Die zum Punkt im linken Bild potenziell gehörigen Punkte im rechten Bild werden mit Hilfe des local affine Frame[Abb. 3] berechnet.

$$N_1(s)=\{(x-1,y,A_{i-1,j}) \mid i,j \epsilon \{-1,0,1\}\}$$
$$N_2(s)=\{(x+1,y,A_{i+1,j}) \mid i,j \epsilon \{-1,0,1\}\}$$
$$N_3(s)=\{(x,y-1,A_{i,j-1}) \mid i,j \epsilon \{-1,0,1\}\}$$
$$N_4(s)=\{(x,y+1,A_{i,j+1}) \mid i,j \epsilon \{-1,0,1\}\}$$
$$\mathbf{A}_{i,j}=\begin{bmatrix} a_1 & a_2 & a_3+a_1 i+a_2 j \\ a_4 & a_5 & a_6+a_4 i+a_5 j \end{bmatrix}$$

Abb. 3 Berechnung des LAF für Nachbarn

Sollte der beste Korrelationswert eines Nachbarn größer c_thr sein, die Differenz des besten und zweitbesten Korrelationswertes größer als epsilon sein und es noch keinen Match für den entsprechenden Punkt im linken Bild geben, wird das neue Match dem Match Vektor hinzugefügt. Die ursprünglichen Matches zusammen mit den neu errechneten Matches werden dann als Rückgabewert der Funktion ausgegeben.

5 TEST DES KOMBINIERTEN ALGORITHMUS

5.1 TEST ALLGEMEIN

Für die Tests wurden jeweils 50000 Durchläufe der orsa Funktion pro Bild und Konfiguration durchgeführt. Dies war notwendig da der ORSA/RANSAC Algorithmus an sich im allgemeinen schon nicht deterministisch ist und auch der zur Optimierung verwendete SPRT Algorithmus ein statistischer Algorithmus ist. Nur durch große Mengen an Durchläufen kann daher ein repräsentatives Ergebnis erzielt werden.

5.2 TEST RANSAC VS RANSAC SPRT

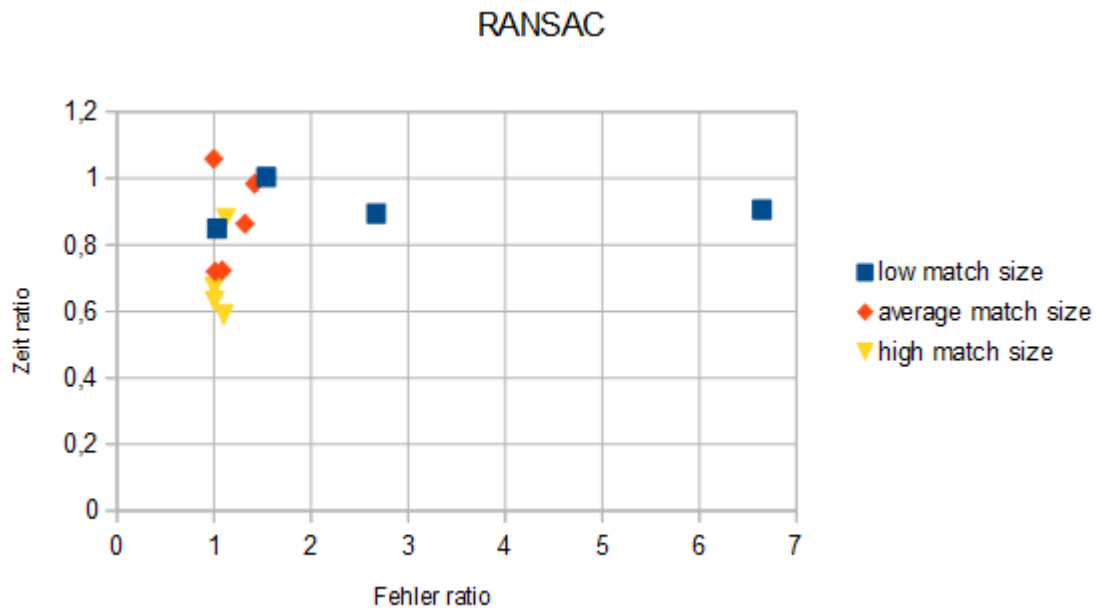

Abb. 4 Diagramm RANSAC versus RANSAC SPRT

Der X-Achse Fehler ratio stellt den RANSAC SPRT Fehler geteilt durch den RANSAC Fehler dar. Die Y-Achse Zeit ratio stellt die Laufzeit des RANSAC SPRT geteilt durch die Laufzeit des RANSAC dar. Die blauen Punkte „low match size“ stellen Bilder dar bei denen weniger als 120 Matches vorlagen. Die gelben Punkte „high match size“ stellen Bilder dar, die über 180 Matches besaßen und die orangen Punkte alle Bilder der Anzahl an Matches zwischen

120 und 180. [ANH 1.1]Über alle Bilder gesehen, beträgt die Laufzeit des RANSAC SPRT 82,91% des RANSAC und der Fehler des RANSAC SPRT 168,75% des RANSAC. [ANH 1.4] Der minNFA des RANSAC ist durchschnittlich 6,3% besser als der des RANSAC SPRT.

5.3 TEST ORSA VS ORSA SPRT

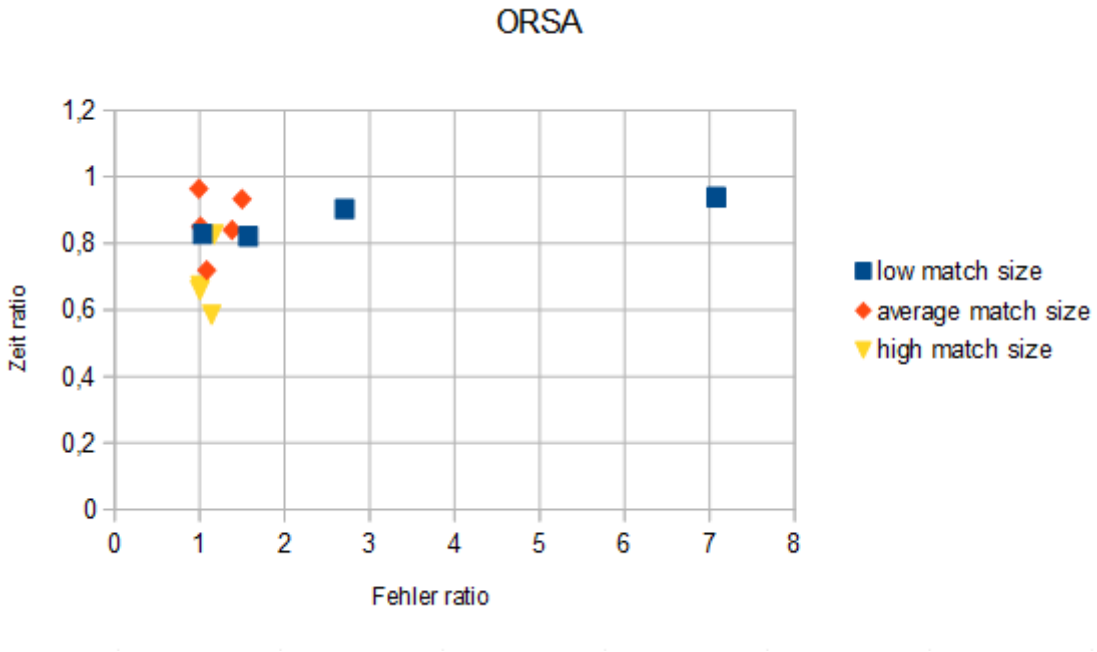

Abb. 5 Diagramm ORSA vs ORSA SPRT

Der X-Achse Fehler ratio stellt den ORSA SPRT Fehler geteilt durch den ORSA Fehler dar. Die Y-Achse Zeit ratio stellt die Laufzeit des ORSA SPRT geteilt durch die Laufzeit des ORSA dar. Die blauen Punkte „low match size“ stellen Bilder dar, bei denen weniger als 120 Matches vorlagen. Die gelben Punkte „high match size“ stellen Bilder dar, die über 180 Matches besaßen und die orangen Punkte alle Bilder der Anzahl an Matches zwischen 120 und 180. [ANH 1.2] Über alle Bilder gesehen, beträgt die Laufzeit des ORSA SPRT 81,08% des ORSA und der Fehler des ORSA SPRT 174,46% des ORSA. [ANH 1.4]Der minNFA des ORSA ist durchschnittlich 6,7% besser als der des ORSA SPRT.

5.4 TEST GROW FUNKTION

[ANH 1.3]Die Grow Funktion erreichte in mehreren Tests ein durchwachsenes Ergebnis. Der Algorithmus erreichte bei einigen Bildern ein gerade zu fantastisches Ergebnis, bei einem der Testbilder senkte er den durchschnittlichen Maximalfehler der Inliner von 2,32 auf 0. Bei anderen Bildern erzeugte er jedoch einen deutlich höheren Maximalfehler als ohne Grown Funktion. Der Test wurde mit 1000 Grow-Steps durchgeführt und verdoppelte durchschnittlich die Anzahl der Matches.

6 FAZIT

Die Verwendung der SPRT Optimierung für den ORSA/RANSAC Algorithmus erscheint nur dann sinnvoll, wenn ein ausreichend großes Set an Matches für das Bild verfügbar ist. Unter dieser Voraussetzung ist der SPRT verbesserte Algorithmus recht zuverlässig, was sich vor allem aus der geringen Differenz der minNFAs des verbesserten Algorithmus zum ursprünglichen ablesen lässt. Streicht man alle Bildpaare mit weniger als 60 Matches, nimmt der minNFA bei Verwendung des verbesserten Algorithmus nur um rund 2% zu, während sich die Laufzeit auf 80% der ursprünglichen Laufzeit verkürzt. Jedoch kann es bei manchen Match Sets unabhängig von der Größe zu einer verringerten Effizienz kommen. Der Grund hierfür scheint in der Struktur des Match Sets zu liegen, das in diesen Fällen ein falsches vorzeitiges Verwerfen einer Fundamental Matrix begünstigt. Man kann diesem Problem mit einer veränderten Einstellung für alpha und beta entgegen wirken, jedoch muss mit einem Geschwindigkeitsverlust gerechnet werden.

Die Grow Funktion erfüllt ihre Aufgabe sehr gut und erzeugt weitere Matches für den ORSA/RANSAC Algorithmus, jedoch kann es vorkommen, dass der ORSA/RANSAC Algorithmus ein 8 Punkte großes Inliner Set mit einem NFA von – Infinite auswählt. Auch im Originalcode kommt dieser Fall manchmal vor, jedoch deutlich seltener. Ob diese Inlinerauswahl wirklich sinnvoll ist, muss geprüft werden. Leider machen diese -Infinit-Fälle die Auswertung der Effizienz der Grow Funktion praktisch unmöglich. Sobald man genug Durchläufe des ORSA/RANSAC Algorithmus auf einer mit Grow vergrößerten Match Menge ausführt, kommt es fast immer dazu das der DurchschnittsminNFA gleich -Infinit wird und damit keine Aussagekraft mehr hat. Auch ohne Grow Funktion kommt dies vor, jedoch nur etwa bei 10% der Fälle, so dass die verbleibenden 90% durchaus noch Aussagekraft besitzen.

7 LITERATURVERZEICHNIS

SCV, Fast Sequential Correspondence Verification by Cosegmentation **,** 2008, http://cmp.felk.cvut.cz/~cechj/SCV/

ORSA, Quasi-Euclidean Epipolar Rectification, MissStereo mit RANSAC und ORSA, 13.09.2011, http://www.ipol.im/pub/art/2011/m_qer/

WALD, Wikipedia Artikel „Sequential Probability Ratio Test“[Online], Stand 07.10.2014, http://de.wikipedia.org/wiki/Sequential_Probability_Ratio_Test

RANSAC WIKI, Wikipedia Artikel „RANSAC-Algotihmus“[Online], Stand 07.10.2014, http://de.wikipedia.org/wiki/RANSAC-Algorithmus

8 ANHANG

8.1 MESSERGEBNIS TABELLEN

8.1.1 RANSAC vs RANSAC SPRT

	Zeit R	Zeit RS	Zeit ratio	Error R	Error RS	Error ratio	Match
IMP 1	59,81	53,5	0,89	0,87	2,32	2,67	54
IMP 2	78,57	71,21	0,91	4,05	26,89	6,64	56
IMP 3	236,86	237,79	1	6,53	10,04	1,54	99
IMP 4	122,24	103,75	0,85	0,82	0,84	1,03	117
IMP 5	531,94	523,53	0,98	2,67	3,79	1,42	126
IMP 6	84,06	60,8	0,72	0,94	1,02	1,08	129
IMP 7	89,17	64,17	0,72	3,24	3,28	1,01	130
IMP 8	613,19	529,65	0,86	1,64	2,16	1,32	132
IMP 9	378,04	400,39	1,06	3,3	3,29	1	150
IMP 10	147,91	93,52	0,63	2,42	2,43	1	298
IMP 11	400,03	352,33	0,88	2,49	2,79	1,12	302
IMP 12	308,82	182,31	0,59	1,33	1,46	1,1	549
IMP 13	729,69	490,72	0,67	1,03	1,03	1	1026

IMP = Image pair

R = RANSAC

RS = SPRT optimierter RANSAC

Match = Anzahl der Matches

ratio = WERT für SPRT optimierter RANSAC geteilt durch Wert für RANSAC

8.1.2 ORSA vs ORSA SPRT

	Zeit O	Zeit OS	Zeit ratio	Error O	Error OS	Error ratio	Match
IMP 1	54,71	49,41	0,9	0,87	2,35	2,71	54
IMP 2	121,07	113,6	0,94	4,11	29,09	7,08	56
IMP 3	207,25	170,13	0,82	6,83	10,75	1,57	99
IMP 4	103,99	86,11	0,83	0,83	0,85	1,03	117
IMP 5	334,14	311,84	0,93	2,51	3,76	1,5	126
IMP 6	79,41	57,11	0,72	0,96	1,04	1,08	129
IMP 7	139,93	118,96	0,85	3,24	3,27	1,01	130
IMP 8	426,18	358,1	0,84	1,52	2,1	1,38	132
IMP 9	612,11	590,08	0,96	3,49	3,46	0,99	150
IMP 10	158,17	104,21	0,66	2,41	2,42	1	298
IMP 11	803,92	664,47	0,83	2,32	2,71	1,17	302
IMP 12	297,14	173,96	0,59	1,28	1,46	1,14	549
IMP 13	725,66	487,92	0,67	1,05	1,06	1,01	1026

IMP = Image pair

O = ORSA

OS = SPRT optimierter ORSA

Match = Anzahl der Matches

ratio = WERT für SPRT optimierter ORSA geteilt durch Wert für ORSA

8.1.3 Grow Algorithmus

	Error G	Error NG	Match NG	Match G
IMP 1	2,21	0,87	54	108
IMP 2	0	4,11	56	126

IMP 3	14,03	6,83	99	126
IMP 4	0	0,83	117	657
IMP 5	5,14	2,51	126	155
IMP 6	2,19	0,96	129	182
IMP 7	0	3,24	130	1113
IMP 8	2,21	1,52	132	156
IMP 9	5,23	3,49	150	177
IMP 10	3,8	2,41	298	407
IMP 11	0	2,32	302	1032
IMP 12	0,02	1,28	549	1048
IMP 13	1,76	1,05	1026	1253

G = ORSA mit Grow Funktion

NG = ORSA ohne Grow Funktion

Match = Anzahl der Matches

IMP = Image pair

8.1.4 minNFA von ORSA/RANSAC mit und ohne SPRT

	NFA O	NFA OS	Ratio O/OS	NFA R	NFA RS	Ratio R/RS	Match
IMP 1	-20141	-16549	1,22	-20037	-16656	1,2	54
IMP 2	-197339	-144433	1,37	-212928	-156260	1,36	56
IMP 3	-367594	-350586	1,05	-374129	-357121	1,05	99
IMP 4	-50048	-49074	1,02	-50234	-49468	1,02	117
IMP 5	-175558	-165991	1,06	-173284	-165971	1,04	126
IMP 6	-77809	-76702	1,01	-79670	-78642	1,01	129
IMP 7	-594418	-587602	1,01	-593692	-587947	1,01	130

IMP 8	-170882	-164786	1,04	-174219	-168340	1,03	132
IMP 9	-285291	-282534	1,01	-293297	-290724	1,01	150
IMP 10	-1526180	-1523110	1	-1525880	-1523710	1	298
IMP 11	- infinit	-1540110	---------	-1548920	-1536740	1,01	302
IMP 12	-1226030	-1211050	1,01	-1233510	-1224850	1,01	549
IMP 13	-3549930	-3542920	1	- infinit	-3570630	---------	1026

IMP = Image pair

O = ORSA

OS = SPRT optimierter ORSA

R = RANSAC

RS = SPRT optimierter RANSAC

Match = Anzahl der Matches

ratio = WERT für Wert für O oder R geteilt durch SPRT optimierter O oder R

8.2 ERKLÄRUNG ZUM QUELLCODE

Der Quellcode basiert auf dem MissStereo Quellcode[ORSA]. Die Dateien grow.cpp und grow.h wurden neu geschrieben und sind kein Teil des MissStereo Quellcodes. Auch die TestMain.cpp wurde neugeschrieben, dient aber nur zu Testzwecken. Die Quellcode Dateien orsa.cpp und orsa.h wurden modifiziert und erweitert. Alle Modifikationen und Erweiterungen wurden mit Hilfe von Kommentaren in diesen Dateien gekennzeichnet.