Humboldt-Universität zu Berlin
Institut für Informatik

Evaluierung von Feature Deskriptoren

Christoffer Fuss[1]

28. Oktober 2011

Gutachter: Prof. Peter Eisert

[1] fuss@informatik.hu-berlin.de

Inhaltsverzeichnis

Kapitel 1

Einleitung

Das Finden von Interessenpunkten in Bildern, sogenannter *Feature Punkte*, spielt in vielen Bereichen der *Computer Vision* eine wichtige Rolle. Sie werden vor allem zum Wiedererkennen von Objekten oder Szenen in einzelnen Bildern oder ganzen Bilddatenbanken benutzt. Weitere Anwendungen sind Objektverfolgung, Bildregistrierung, das Erstellen von 3D Modellen aus mehreren Bildern, Kamera Kalibrierung, Roboter Lokalisierung und viele mehr. Zur Berechnung solcher Feature Punkte gibt es viele verschiedene Verfahren. Ziel dieser Arbeit ist es, diese Verfahren auf Testdaten anzuwenden und miteinander zu vergleichen und zu evaluieren. Ausführlicher betrachtet werden SIFT, SURF und ASIFT. Weitere Verfahren werden kurz erläutert. Dabei sind vor allem die zahlreichen Parameter der einzelnen Verfahren interessant, welche auf die Testdaten optimiert werden sollten, um bessere Ergebnise zu bekommen.

Die Suche nach Beziehungen zwischen Bildern lässt sich in 3 Teile einteilen. Als erstes werden beide Bilder nach potentiellen Interessenpunkten hoher Einzigartigkeit durchsucht, was mit einem *Detektor* erreicht wird. Die wichtigste Eigenschaft des Detektors ist die *Wiederholbarkeit* (repeatability). Diese sagt aus, wie zuverlässig der Detektor die selben Interessenpunkte unter anderen Blickwinkeln, Perspektiven oder Skalierungen findet. Findet der Detektor z.B. unter allen Skalierungsänderungen die selben Interessenpunkte, so nennt man diesen invariant gegenüber Skalierung. Um eine hohe Wiederholbarkeit zu gewährleisten, sollte der Detektor möglichst invariant gegen sämtliche Bildänderungen sein, also invariant gegenüber Rotationen, Skalierungen und Änderungen in Helligkeit und Kontrast. Des Weiteren sollten sie eine hohe Einzigartigkeit aufweisen, so dass selbst in Datenbanken von mehreren tausend Bildern eine eindeutige Zuordnung möglich ist. Im zweiten Schritt wird die Umgebung um jeden Interessenpunkt betrachtet und durch einen Merkmals-Vektor bzw. *Feature Vektor* beschrieben, was mit einem *Deskriptor* erreicht wird. Auch der Deskriptor sollte in den meisten Anwendungsbereichen invariant gegen sämtliche Bildänderungen und unempfindlich gegenüber Rauschen sein.

Die Detektoren und Deskriptoren werden in Kapitel 2 erläutert. Im dritten Teil werden die Deskriptoren bzw. Feature Vektoren der beiden Bilder miteinander verglichen bzw. *gematcht*, um so korrespondierende Punkte zu ermitteln, was mit einem *Matcher* erreicht wird. Die verschiedenen Matching Strategien werden in Kapitel 3 erläutert. In Kapitel 4 werden die Testdaten beschrieben, mit denen die einzelnen Verfahren getestet wurden. Die Ergebnisse dieser Untersuchungen werden in Kapitel 5 vorgestellt und in Kapitel 6 ausgewertet und diskutiert.

Kapitel 2

Verfahren

In diesem Kapitel werden die Verfahren SIFT, ASIFT und SURF näher beschrieben. Über weitere Verfahren wird ein kurzer Überblick gegeben.

2.1 SIFT

SIFT - Scale Invariant Feature Transform wurde von David G. Lowe an der University of British Columbia entwickelt und erstmals 1999 veröffentlicht. Das Verfahren lässt sich grob in 4 Teile teilen: Lowe (2004)

1. **Extremwertsuche im Skalenraum:** Suche nach potentiellen Interessenpunkten über allen Skalierungen und allen Bereichen des Eingangsbildes. Dazu wird eine Difference of Gaussian Pyramide, kurz DoG, erzeugt. Diese Pyramide wird anschließend nach Extrempunkten durchsucht, welche dann die potentiellen Interessenpunkte darstellen. Da über alle Skalierungen und Bildbereiche gesucht wird, sind die gefundenen Extrempunkte invariant gegenüber Skalierung und Orientierung. Die Skalierung ist vor allem dafür interessant, um unterschiedlich weit entfernte Objekte zu erkennen.

2. **Lokalisierung der Interessenpunkte:** Die Extrempunkte aus dem vorigen Teil werden in diesem Teil lokalisiert und ihre Skalierung bzw. Größe wird bestimmt. Danach werden sie auf Stabilität untersucht und die Punkte verworfen, welche nicht stabil genug sind. Ein Punkt ist stabil, wenn er umempfindlich gegenüber kleinen Änderungen des Bildes bzw. Rauschen ist. Bei Unstabilen Feature Punkten ist die Gefahr groß, dass diese falsch gematcht werden.

3. **Orientierungsberechnung:** Für die verbleibenden Interessenpunkte wird eine charakteristische Richtung berechnet, welche auf lokalen Gradienten Richtungen basiert.

4. **Deskriptor Berechnung:** Die Umgebung um jeden Interessenpunkt wird anhand der lokalen Gradienten beschrieben. Anschließend werden die charakteristischen Merkmale der Interessenpunkte und deren Umgebung in einen Vektor geschrieben und für eine spätere Wiedererkennung gespeichert.

In den folgenden Abschnitten werden die 4 Schritte ausführlicher beschrieben.

2.1.1 SIFT-Detektor

Die Feature Punkte werden in mehreren Schritten bzw. Etappen berechnet. Zuerst werden potentielle Kandidaten für Interessenpunkte im Eingangsbild gesucht, welche dann weiter untersucht und gefiltert werden. Um Skalierungsinvarianz zu erhalten ist es erforderlich, das Eingangsbild auf verschiedenen Skalierungen zu untersuchen. Dafür wird eine kontinuierliche Skalierungsfunktion benutzt, welche auch als Skalenraum bezeichnet wird.
Der Skalenraum ist definiert als eine Funktion $L(x, y, \sigma)$, welche durch eine Gaußfaltung des Eingangsbildes $I(x, y)$ entsteht:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.1}$$

wobei $*$ der Faltungsoperator in x und y ist und

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.2}$$

Um nun die Interessenpunkte aus dem Skalenraum zu bestimmen, wird eine weitere Funktion $D(x, y, \sigma)$ benötigt, welche als Difference of Gaussians oder kurz DoG bezeichnet wird. Die Funktion wird berechnet, indem die benachbarten Skalierungen des Skalenraumes voneinander subtrahiert werden. Die benachbarten Skalierungen des Skalenraumes unterscheiden sich in einem konstanten Faktor k , welcher den Grad der Glättung bzw. Unschärfe bestimmt:

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \tag{2.3}$$
$$= L(x,y,k\sigma) - L(x,y,\sigma) \tag{2.4}$$

Abbildung 2.1 zeigt die Konstruktion eines Skalenraumes. Der Skalenraum besteht aus mehreren Octaven, welche wiederum aus mehreren Bildern bestehen. Die erste Octave besteht aus dem Ausgangangsbild in Orginalgröße und wiederholt mit dem Gaußoperator gefaltenden Bildern der selben Skalierung. Das letzte Bild der ersten Octave wird dann verkleinert, indem die Auflösung des Bildes halbiert wird, und dient dan als erstes Bild der zweiten Octave und so weiter. Eine Octave ist also quasi eine Ebene aus mehreren Bildern gleicher Skalierung. Wie viele Bilder eine Octave enthält bzw. wie oft das Bild gefaltet wird, hängt von einem Parameter ab. Wie viele Octaven der Skalenraum enthält hängt ebenfalls von einem Parameter ab (2.1.3). Für jede Oktave des Skalenraums wird das Eingangsbild wiederholt mit dem Gaußoperator gefaltet (linke Seite). Auf der rechten Seite sieht man die DoG Bilder, welche durch Subtraktion entstehen. Ist eine Oktave komplett abgearbeitet, also wiederholt gefaltet, wird die Auflösung des letzten Bildes der vorherigen Oktave halbiert und der Faltungsvorgang wird erneut durchgeführt. Die Halbierung der Bilddimension wird erreicht, indem jedes zweite Pixel jeder Zeile und Spalte weggelassen wird.

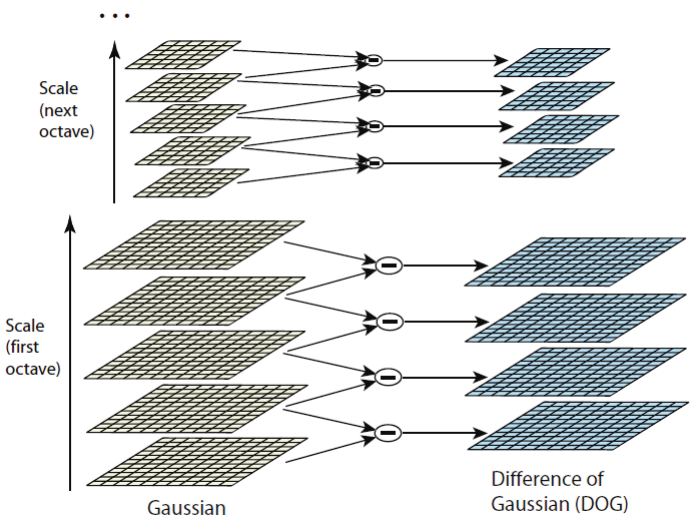

Abbildung 2.1 Konstruktion eines Skalenraums Lowe (2004)

Ist der Skalenraum gebildet, wird die DoG Funktion $D(x,y,\sigma)$ nach Extrempunkten überprüft. Dazu wird jeder Punkt mit seinen acht Nachbarn auf der gleichen Ebene, sowie seinen neun Nachbarn der darüber und darunter

liegenden Oktaven verglichen. Ist der Punkt größer oder kleiner als all seine 26 Nachbarn, so liegt ein Extrempunkt vor.

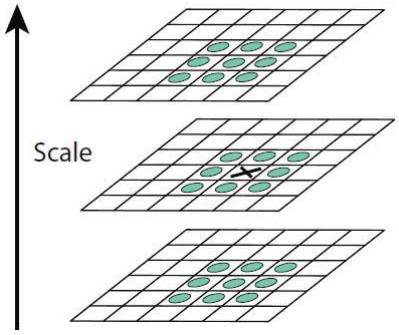

Abbildung 2.2 Extremwertbildung: Jeder Punkt wird mit seinen 8 Nachbarn der gleichen bzw. aktuellen Oktave und seinen jeweils 9 Nachbarn der beiden benachbarten Oktaven verglichen Lowe (2004).

Die gefundenen Extrempunkte bilden die Kandidaten für potentielle Interessenpunkte. In den nächsten Schritten werden diese Punkte weiter untersucht und gefiltert. So werden z.B. Punkte mit zu niedrigem Kontrast heraus gefiltert, indem der Unterschied zwischen dem Mittelpunkt des Punktes und dessen Nachbarn mit einem bestimmten Schwellwert verglichen wird. Auch Punkte, die auf Kanten liegen, sind unerwünscht und werden heraus gefiltert. Um zu überprüfen, ob ein Interessenpunkt auf einer Kante liegt, wird die Krümmung des Punktes über die Fläche bestimmt, welche sich durch eine $2x2$ Hesse Matrix berechnen lässt.

2.1.2 SIFT-Deskriptor

Um Rotationsinvarianz zu gewährleisten, wird jedem vom Detektor ermittelten Interessenpunkt bzw. Feature Punkt, eine Orientierung zugewiesen. Um die Orientierung zu bestimmen, wird zunächst die Oktave L aus dem Skalenraum gewählt, welche dem Feature Punkt am nächsten liegt. Des weiteren wird für jeden Samplepunkt $L(x,y)$ dieser Oktave der Gradient $\Theta(x,y)$ und dessen Größe $m(x,y)$ berechnet:

$$m(x,y) = \sqrt{L((x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.5)$$

$$\Theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \quad (2.6)$$

Anschließend wird ein Histogramm der Gradienten in der Umgebung des Feature Punktes bestimmt. Dieses Histogramm deckt 360° um den Feature Punkt ab und ist in 36 Teile geteilt. Jeder Wert des Histogramms wird nach der Größe des Gradienten $m(x,y)$ und der Entfernung vom Zentrum des Feature Punktes gewichtet, welche durch eine Gaußfunktion mit einem σ von 1.5 mal der Größe des Feature Punktes gewichtet wird. Somit fallen Änderungen in den äußeren Bereichen des Feature Punktes nicht so stark ins Gewicht. Um nun die Richtung zu bestimmen, wird im Histogramm nach dem stärksten Orientierungsgradienten gesucht. Ist die Länge des zweitstärksten Gradienten des Histogramms 80% von der Länge des stärksten Gradienten, so wird ein zweiter Feature Punkt erzeugt, was allerdings laut Lowe nur in 15% der Fälle vorkommt. So wird die Stabilität erhöht, da der Punkt beim späteren matchen auch in beide Richtungen erkannt werden kann.
Die verbliebenen Feature Punkte sind nun durch ihre Position, Größe und Richtung eindeutig bestimmt und jedem Feature Punkt wurde eine Region zugeordnet. Im nächsten Schritt werden die Deskriptoren erzeugt, welche anhand von $m(x,y)$ und $\Theta(x,y)$ berechnet werden, was mit der folgenden Abbildung 2.3 gezeigt wird.

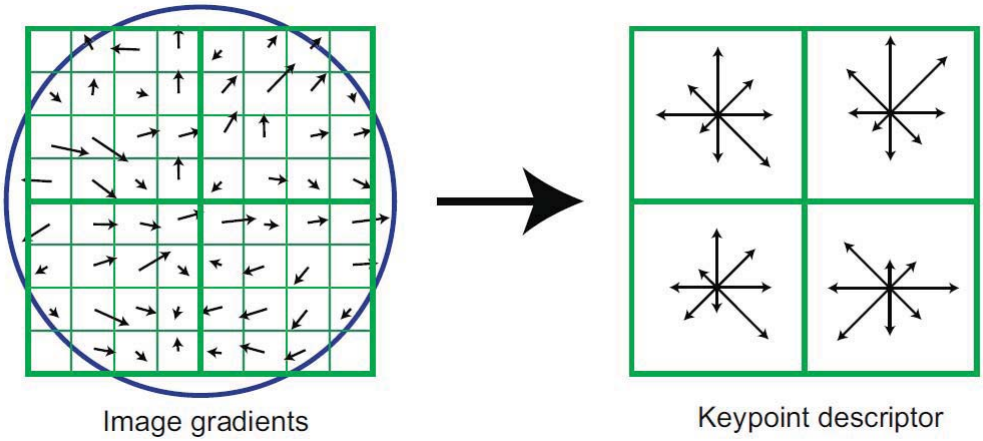

Abbildung 2.3 **Linke Seite:** Aus den lokalen Gradienten $\Theta(x,y)$ und deren Größen $m(x,y)$ werden um jeden Samplepunkt $L(x,y)$ einer Region eines Feature Punktes die Deskriptoren gebildet.
Rechte Seite: Für die 8x8 Samplepunkte wird dann ein Gradientenhistogramm mit je acht Richtungen erstellt und daraus ein 2x2 Deskriptor array gebildet Lowe (2004).

Bessere Resultate erhält man mit 4x4 Deskriptor Arrays und 8 möglichen Richtungen, welche dann aus 16x16 Samplepunkten berechnet werden und nicht nur aus 8x8 Samplepunkten. Das führt zu Deskriptoren bzw. Feature Vektoren mit $4x4x8 = 128$ Einträgen für jeden Feature Punkt. Der Deskriptor ist auch invariant gegenüber Änderungen von Helligkeit und Kontrast. Um Invarianz gegenüber Kontrast zu erhalten, werden die Feature Vektoren normalisiert.

2.1.3 Implementierung und Parameter

Von SIFT gibt es zahlreiche Implementierungen. Für diese Arbeit wurde nicht die Implementierung von Lowe Lowe (2005) sondern die vlfeat Implementierung von Andrea Vedaldi und Brian Fulkerson betrachtet Vedaldi and Fulkerson (2011). Bei vlfeat handelt es sich um eine open source Sammlung von Computer Vision Algorithmen, wozu neben SIFT auch MSER (Maximally Stable Extremal Regions) gehört. Wie die Abbildung 2.4 zeigt, liefern die Implementierungen von vlfeat und Lowe fast identische Ergebnisse:

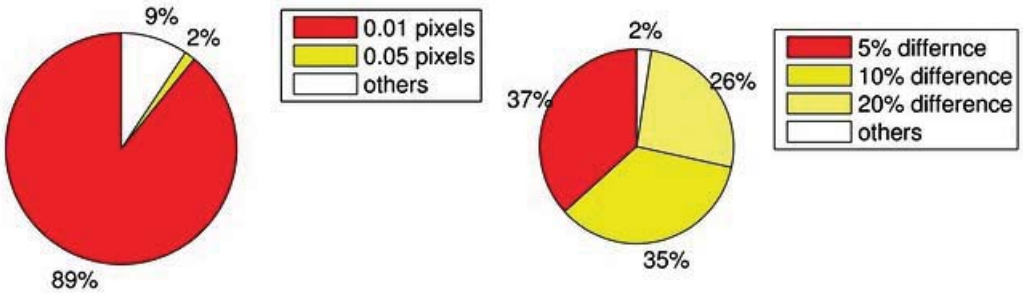

Abbildung 2.4 **Linke Seite:** Prozentualer Anteil der von vlfeat und Lowe gefundenen Feature Punkte, welche bis auf 0.01 bzw. 0.05 Pixel übereinstimmen. Das bedeutet, dass wenn beide Implementierungen auf das selbe Bild angewendet werden, 89% der gefundenden Feature Punkte bis auf 0.01 Pixel übereinstimmen.
Rechte Seite: Prozentualer Anteil der von vlfeat und Lowe ermittelten Deskriptoren, deren Abstand kleiner ist als 5%, 10% und 20% vom durchschnittlichen Deskriptor Abstand. Das bedeutet, dass wenn beide Implementierungen auf das selbe Bild angewendet werden, 37% der gefundenden Deskriptoren einen Abstand haben, welcher sich nur um 5% vom durchschnittlichen Deskriptor Abstand unterscheidet Vedaldi and Fulkerson (2011).

Die vlfeat Implementierung verfügt über zahlreiche Parameter, welche im

folgenden näher beschrieben werden. Um Skalierungsinvarianz zu erreichen, wird ein Skalenraum konstruiert. Die Konstruktion dieses Skalenraumes wird von folgenden Parametern beeinflusst. Die Zahl in eckigen Klammern zeigt den Standard Wert von vlfeat:

- **Number of octaves [Maximum]:** Dieser Wert gibt die Anzahl der Oktaven im Skalenraum an. Fügt man dem Skalenraum eine Oktave hinzu, hat das den Effekt, dass die Auflösung des Bildes halbiert wird. Da man immer auf möglichst vielen Größen bzw. Skalierungen suchen möchte, sollte die Anzahl der Oktaven möglichst groß oder am besten maximal sein.

- **First octave index [0]:** Dieser Wert gibt den Index der ersten Oktave an. Ein Wert von 0 bedeutet, dass die erste Oktave das Bild in voller Auflösung darstellt. Ist der Wert größer als 0, so fängt die erste Oktave mit einer geringeren Auflösung des Bildes an. Ein Index von 1 bedeutet zum Beispiel, dass die erste Oktave das Bild in halbierter Auflösung enthält. Auch negative Werte sind möglich, ist der Index -1, so wird mit doppelter Auflösung begonnen, was zur Folge hat, dass sehr kleine Feature Punkte gefunden werden.

- **Number of levels per octave [3]:** Dieser Wert gibt an, wie oft jede Oktave abgetastet bzw. gefaltet wird. Erhöht man diesen Wert, werden mehr Feature Punkte gefunden, welche allerdings in der Praxis an Stabilität gegenüber Rauschen verlieren.

Instabile Feature Punkte, welche z.B. auf Kanten liegen oder in Bildbereichen mit niedrigem Kontrast, sollen heraus gefiltert werden. Die folgenden 2 Parameter regeln diese Filterung:

- **Peak threshold [0]:** Dieser Wert gibt den minimalen Kontrast an, damit ein Feature Punkt akzeptiert wird. Punkte mit zu niedrigem Kontrast können heraus gefiltert werden, indem der Unterschied zwischen dem Mittelpunkt des Feature Punktes und den Nachbarpunkten mit dem peak threshod verglichen wird. Ist der Kontrast zu klein, ist die Wahrscheinlichkeit hoch, dass der Feature Punkt durch Bildrauschen entstanden und somit instabil und unerwünscht ist. Erhöht man den peak threshold, werden mehr Feature Punkte verworfen.

- **Edge threshold [10]:** Dieser Parameter ist dazu da, Feature Punkte auf Kanten zu filtern. Feature Punkte auf Kanten sind nicht sehr stabil, da

eine spätere Wiedererkennung des Punktes an einen beliebigen anderen Punkt auf der Kante geschehen könnte. Es lässt sich durch eine $2x2$ Hesse Matrix berechnen, ob ein Feature Punkt auf einer Kante liegt, indem die Krümmung des Punktes über die Fläche bestimmt wird. Zu dieser Filterung wird ein Schwellwert gesetzt, der Edge threshold. Dieser liefert laut Lowe (2004) die besten Ergebnisse für $Edgethreshold = 10$. Erhöht man diesen Wert, erhält man mehr Feature Punkte auf Kanten, verringert man ihn, werden mehr Feature Punkte auf Kanten heraus gefiltert und somit verworfen.

Im folgenden werden die Parameter für die Berechnung der Deskriptoren beschrieben:

- **Magnification factor [3]:** Um die Größe des Deskriptors zu ermitteln, wird die Skalierung bzw. Größe des Feature Punktes mit dem magnification factor multipliziert. Der magnification factor gibt an, wie viel größer ein Deskriptor Abschnitt im Vergleich zur Skalierung des Feature Punktes ist. Erhöht man diesen Wert, wird die Bildregion vergrößert.

- **Gaussian window size:** Jedes image sample im Skalenraum wird durch ein Gaußfenster gewichtet, um Gradienten mit einer großen Entfernung vom Zentrum des Feature Punktes zu vernachlässigen. Diese Entfernung wird durch eine Gaußfunktion mit einer Standartabweichung von 1.5 mal der Größe des Feature Punktes gewichtet. Vergrößert man das Gaußfenster (Gaussian window size), dann werden Gradienten mit großen Entfernungen zum Zentrum des Feature Punktes stärker gewichtet. Wird der Wert verkleinert, dann fallen die Gradienten im Zentrum stärker ins Gewicht

2.2 Affine-SIFT (ASIFT)

Wie oben beschrieben ist SIFT invariant gegenüber Zoom, Rotation und Translation und damit invariant gegen 4 der 6 Parameter einer affinen Transformation. ASIFT erweitert SIFT, sodass Invarianz gegenüber allen 6 Parametern erreicht wird. Diese 2 fehlenden Parameter sind die Winkel, welche die Ausrichtung der Kameraachse festlegen. Indem diese beiden Winkel immer wieder variiert werden, können alle möglichen Bildansichten simuliert werden, welche von SIFT nicht betrachtet werden. Nach dieser Simulation der beiden Winkel, wird SIFT angewendet. Abhängig von der Anzahl der simmulierten affinen Abbil-

dungen wird ASIFT invarianter gegenüber affinen Transformationen. Morel and Yu (2008)

2.2.1 Transition tilts

Jede affine Transformation A = $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ hat eine eindeutige Zerlegung

$$A = H_\lambda R_1(\psi) T_t R_2(\phi) = \lambda \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} t & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \tag{2.7}$$

wobei $\lambda > 0$ ist, λt die Determinante von A ist, R_i Rotationen sind, $\phi \in [0, \pi)$ und T_t eine Neigung ist, also eine Diagonalmatrix mit dem ersten Eigenwert $t > 1$ und dem 2. Eigenwert $t = 1$.
Abbildung 2.5 zeigt 1.8 schematisch.

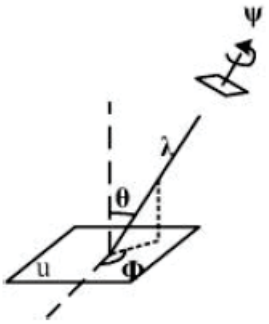

Abbildung 2.5 Das kleine Parallelogramm in der Ecke rechts stellt eine Kamera da, welche auf das Bild u schaut. Die Winkel ϕ und θ sind die oben erwähnten Winkel und beschreiben den Blickpunkt der Kamera. ϕ wird *longitude angle* genannt und $\theta = \arccos(1/t)$ *latitude angle*. Ein dritter Winkel ψ stellt die Drehung der Kamera dar und λ den Zoom der Kamera. Die Kamera kann sich also um ihre eigene optische Achse drehen (ψ) und näher an das Bild heranfahren (λ) Morel and Yu (2008).

Der Parameter t in (1.7) wird *absolut tilt* bzw. *absolute Neigung* genannt und gibt den Unterschied zwischen der frontalen Ansicht und einer geneigten Ansicht an. In der Realität sind aber so gut wie immer beide Bilder geneigt. Daher ist also nicht die absolute Neigung von Interesse, sondern der Unterschied zwischen zwei geneigten Bildern. Es wird ein Maß benötigt, welches angibt,

wie stark sich zwei Bilder nach Anwendung einer affinen Transformation unterscheiden. Dieses Maß wird *transition tilt* oder *Übergangsneigung* genannt. Die Übergangsneigung $\tau(u_1, u_2)$ zwischen 2 Bildern u_1 und u_2 hängt nur von der absoluten Neigungen t und t' und der Differenz von ϕ und ϕ' ab:

$$\tau(u_1, u_2) = \tau(t, t', \phi - \phi') \tag{2.8}$$

Des Weiteren gilt:

$$t/t' \leq \tau(u_1, u_2) = \tau(u_2, u_1) \leq tt' \tag{2.9}$$

mit $t \geq t'$.

Wie die Abbildung 2.6 zeigt, kann die Übergangsneigung um einiges größer sein, als die absolute Neigung.

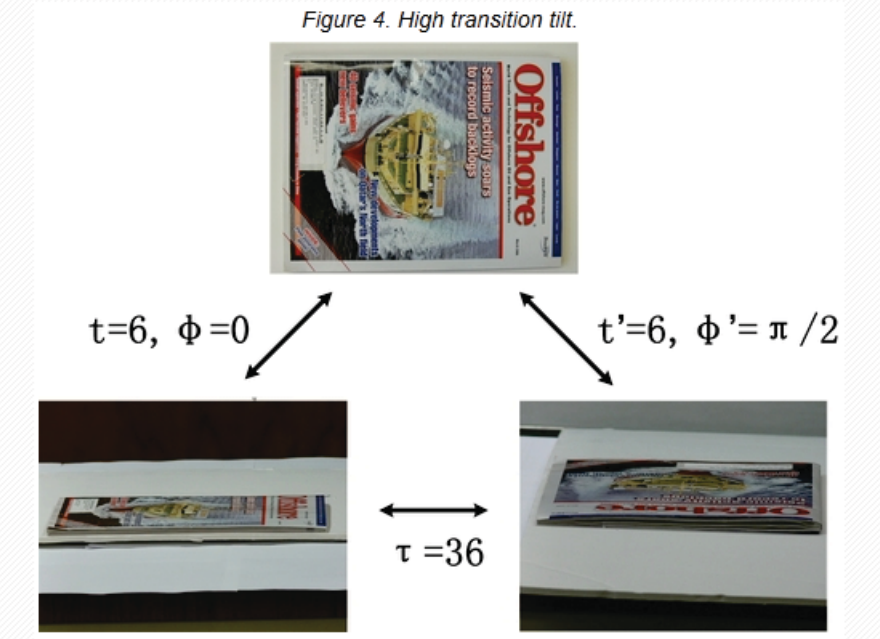

Abbildung 2.6 Die Abbildung zeigt eine hohe Übergangsneigung. Das Bild oben in frontaler Ansicht wird einmal in eine Richtung geneigt (links) und einmal in orthogonale Richtung geneigt(rechts). Die absolute Neigung ist in beide Richtungen 6, daraus ergibt sich eine Übergangsneigung von $tt' = 6 \cdot 6 = 36$ Morel and Yu (2008).

Daraus folgt, dass ein Matcher invariant gegen möglichst große Übergangsneigungen sein sollte. ASIFT liefert auch bei einer hohen Übergangsneigung

von über 32 noch gute Resultate, wo andere Verfahren wie SIFT schon bei Übergangsneigungen von größer als 2 kaum noch oder gar keine Matches mehr finden.

2.2.2 Implementierung und Parameter

Verwendet wurde die IPOL Implementierung in der Version 2.1.2 von Guoshen Yu und Jean-Michel Morel (Morel and Yu, 2011).
ASIFT simmuliert 2 Kameraparameter und verwendet anschließend direkt SIFT. Da sich die einzelnen SIFT Parameter dieser Implementierung zum teil etwas von der vlfeat Implementierung unterscheiden, werden diese hier noch einmal aufgeführt und die wichtigsten kurz erläutert:

- **OctaveMax [100000]:** äquivalent zu *number of octaves* von vlfeat
- **DoubleImSize [0]** Ist dieser Wert gleich 1, wird die Größe des Ausgagsbildes verdoppelt.
- **InitSigma [1.6]:** Dieser Parameter gibt den Grad der Glättung des Eingangsbildes in jeweils der ersten Ebene einer jeden Oktave an. Gute Werte liegen laut den ASIFT Autoren zwischen 1.2 und 1.8. Diese Werte wurden experimentell bestimmt.
- **BorderDist [5]** Dieser Wert gibt die minimale Anzahl an Samplen an, welche zwischen den Extremwerten im DoG Skalenraum und dem Rand des Bildes mindestens liegen müssen. Liegt ein Feature Punkt Nahe am Rand (BorderDist < 15), so liegen Teile des Deskriptors außerhalb des Bildes, welche dann durch das nächste Pixel approximiert werden.
- **Scales [3]:** äquivalent zu *Number of levels per octave* von vlfeat
- **PeakThresh [255.0 * 0.04 / 3.0]:** siehe vlfeat
- **EdgeThresh [0.12]:** siehe vlfeat
- **EdgeThresh1 [0.08]:** Der EdgeThresh der ersten Okatve ist leicht verschieden von dem der anderen Oktaven.
- **OriBins [36]:** Dieser Parameter gibt die Anzahl der Teile bzw. *bins* im Histogramm an. Bei Lowe und vlfeat ist dieser Wert ebenfalls 36.

- **OriSigma [1.5]:** Dieser Parameter dient der Berechnung der Orientierung eines Feature Punktes und wird dabei einfach mit octScale multipliziert. OctScale ist die Skalierung bezüglich der Oktave.

- **OriHistThresh [0.8]:** Für jeden lokalen Extremwert im Histogramm wird überprüft, ob der Extremwert größer oder gleich OriHistThresh mal dem maximalen Wert im Histogramm liegt. Ist das der Fall, wird ein Feature Punkt erzeugt.

- **MaxIndexVal [0.02]**: Dieser Wert ist dafür da, den Einfluss von einzelnen großen Magnituden zu reduzieren, welche die Normalisierung der Vektoren beeinflusst haben.

- **MagFactor [3]** siehe vlfeat

- **IndexSigma [1.0]** Die einzelnen Teilwerte des Histogramms werden nach ihrer Entfernung vom Zentrum des Feature Punktes gewichtet. Diese Entfernung wird durch eine Gaußfunktion mit IndexSigma mal der Größe des Feature Punktes gewichtet.

- **IgnoreGradSign [0]:** Ist dieser Wert gleich 1, werden Gradienten mit entgegengesetzten Vorzeichen gleich behandelt. Das erhöht die Invarianz gegenüber Beleuchtung.

2.3 SURF

SURF - Speeded-UP Robust Features verwendet ebenfalls Integralbilder zur Ermittlung der Feature Punkte, wobei ein auf Hesse-Matrix basierter Detektor verwendet wird. Der Deskriptor bzw. Feature Vektor hat eine Länge von 64. Detektor und Deskriptor werden in den folgenden Abschnitten beschrieben. SURF ist invariant gegenüber Skalierung und Rotation. Ziel von SURF ist es, die Berechnung im Vergleich zu SIFT zu beschleunigen. Dafür werden die Gaußfilter von SIFT durch Mittelwertfilter ersetzt Bay et al. (2008).

2.3.1 Detektor

SURF verwendet zur Detektion der Feature Punkte die Determinante der Hesse-Matrix. Die Hesse-Matrix ist folgendermaßen definiert:

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(\mathbf{x},\sigma) L_{xy}(\mathbf{x},\sigma) \\ L_{xy}(\mathbf{x},\sigma) L_{yy}(\mathbf{x},\sigma) \end{bmatrix} \tag{2.10}$$

wobei $\mathbf{x} = (x, y)$ ein Punkt im Bild I , σ die Standartabweichung der Gaußfunktion und $L_{xx}(x,\sigma)$ die Gaußfaltung des Bildes I ist. Die Determinante der Hesse-Matrix kann mit Hilfe von Richtungsableitungen berechnet werden:

$$\det(H_{aapprox}) = D_{xx}D_{yy} - (\omega D_{xy})^2 \tag{2.11}$$

wobei D_{xx}, D_{xy} und D_{yy} die Box-Filter sind, welche Antworten bzw. Ausgaben mit ω gewichtet werden. Diese approximierten Determinanten werden in einer so genannten *Blob-Response Map* gespeichert und auf lokale Maxima und Minima untersucht, um so invariante Schlüsselpunkte zu erhalten. Die Extrempunkte müssen wie bei SIFT auf verschiedenen Skalierungen gesucht werden, um Skalierungsinvarianz zu erreichen. Bei SURF wird allerdings im Vergleich zu SIFT nicht wiederholt das Bild mit einem Gaußfilter geglättet und dann um einen konstanten Faktor verkleinert, sondern die Filtermatrizen werden vergrößert und immer wieder auf das Originalbild angewendet. Dieses Vorgehen hat des weiteren den Vorteil, dass keine Aliasing-Effekte auftreten können, was bei der Verkleinerung der Bilder allerdings der Fall sein. Abbildung 2.7 zeigt diesen Unterschied:

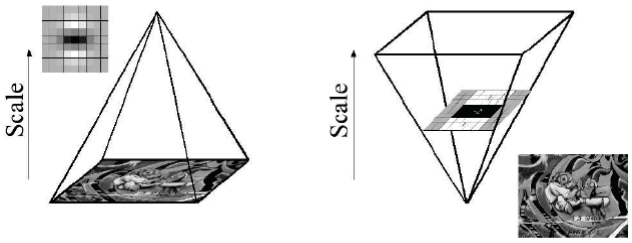

Abbildung 2.7 Auf der linken Seite wird die Größe des Bildes immer weiter verkleinert, auf der rechten Seite werden die Filter nach und nach vergrößert Bay et al. (2008).

2.3.2 Deskriptor

Der Deskriptor beschreibt die Intensitätsverteilung der Pixel um den Feature Punkt, ähnlich dem Deskriptor von SIFT. Mit Hilfe von *Haar-Wavelets* wird die

erste Ableitung in x und y Richtung in der Region bzw. Umgebung des Feature Punktes berechnet. Daraus wird dann der Deskriptor bzw. Feature Vektor erstellt, welcher eine Länge von 64 hat. Zunächst wird, wie bei SIFT, jedem Feature Punkt eine Orientierung bzw. Ausrichtung zugewiesen, um Invarianz gegenüber Rotation zu erhalten. Alle weiteren Berechnungen basieren dann auf dieser Orientierung. In einer Umgebung vom Radius $6s$, wobei s die Skalierung des Feature Punktes ist, werden die Filterantworten von Haar-Wavelets berechnet, welche eine Ableitung in x und y Richtung darstellen.
Zur Berechnung der Feature Vektoren wird ein Quadrat über den Schlüsselpunkt gelegt, welches abhängig von der Skalierung ist und eine Größe von $20s$ hat, wobei s die Skalierung des ermittelten Feature Punktes ist. Dieses Quadrat wird nach der Orientierung des Feature Punktes ausgerichtet und in 4x4 Subregionen eingeteilt. Für jede der 16 Subregionen werden 5x5, also 25 Punkte bewertet, welche über die Region gleichmäßig verteilt werden. Abbildung 2.8 verdeutlicht die Berechnung des Deskriptors:

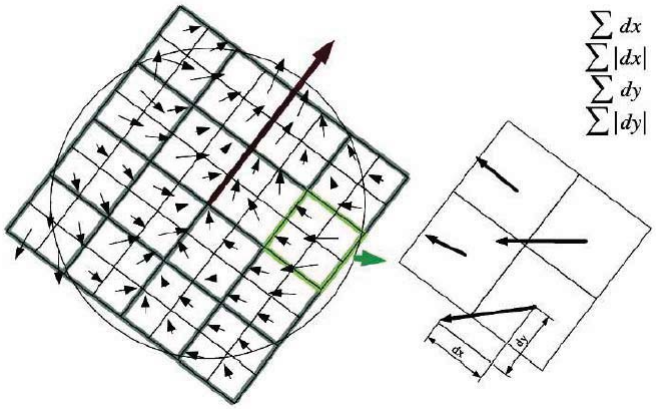

Abbildung 2.8 Berechnung des Deskriptors Bay et al. (2008)

dx und dy bezeichnen die Werte, welche sich für jeden Sample Punkt ergeben (Antworten der Haar Wavelets). Um eine bessere Wiederholbarkeit zu erhalten und unempfindlicher gegen Bildrauschen zu sein, werden die Werte dx und dy mit einem Gaußfilter mit $\sigma = 3.3s$ geglättet. Für den Feature Vektor wird die Summe der Filterantworten für jeden Punkt in der Subregion berechnet und die Beträge dieser Werte summiert.

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{2.12}$$

Aus diesen Vektoren der einzelnen Subregionen setzen sich dann die Feature

Vektoren für jeden Feature Punkt zusammen, was eine gesamte Vektorlänge von 64 Einträgen ergibt Bay et al. (2008).

2.3.3 Implementierung und Parameter

Auch von SURF gibt es bereits viele Implementierungen. Für diese Arbeit wurde die *SURFmex* Implementierung von Peter Strandmark der LUND Universität genommen Strandmark (2010). Dabei handelt es sich um ein Matlab Interface, welches OpenCV benutzt.
Die Implementierung bietet folgende Parameter:

- nOctaveLayers [2]: Anzahl der Ebenen in einer Oktave
- nOctaves [4]: Anzahl der Oktaven
- hessianThreshold [500]: Dieser Wert regelt die Anzahl der gefundenden Feature Punkte, je höher er ist, um so weniger Feature Punkte werden zugelassen.
- extended [0] : Wird dieser Wert auf 1 gesetzt, hat der SURF Deskriptor bzw. Feature Vektor eine Länge von 128 Elementen, statt einer Länge von 64 im default Fall.

2.4 Weitere Verfahren

Dieser Abschnitt gibt einen groben Überblick über weitere Detektoren und Deskriptoren.

2.4.1 SKB

Der SKB Feature Deskriptor wurde am Fraunhofer Institute for Telecommunications Heinrich Hertz Institute entwickelt. SKB steht dabei für *Semantic Kernels Binarized*.Zilly et al. (2011) Solche *Kernels* bzw. Kerne bestehen aus grundlegenden, geometrischen Strukturen wie z.B. Ecken, Kanten oder bestimmten Flächen. Die Hauptidee des Verfahrens ist, dass eine Reihe von Faltungen mit

verschiedenen gefalteten Kernen, auf die Region um einen Feature Punkt angewendet wird. Um jeden Feature Punkt werden 16 Kerne an 16 verschiedenen Positionen berechnet, was $16 * 16 = 256$ Ausgaben ergibt, welche anschließend binarisiert werden. Auf Grund dieser Binarisierung hat der Deskriptor trotz der hohen Dimension von 256 nur eine Größe von 32 Byte. SIFT z.b. hat eine Dimension von nur 128, aber eine Größe von $4 * 128 = 512$ Bytes.

Detektor

Beim SKB Detektor handelt es sich um einen *blob* Detektor im Skalenraum. Unter blobs versteht man Gebiete, dessen Graustufenwerte im Zentrum, sich stark von den umgebenden Pixeln unterscheiden. Um solche Gebiete zu ermitteln, kann die *Mexican hat* Funktion oder die Gauß-Laplace Funktion verwendet werden, welche dann nach Extremwerten durchsucht werden. Der Detektor benötigt nur eine einzige Filter Operation, welche auf Integralbildern basiert. Diese dienen der schnellen Berechnung von Pixelsummen innerhalb rechteckiger Ausschnitte von Bildern. Nach Anwendung des Filters werden dessen Ergebnisse in einer Skalenraum Pyramide gespeichert, welche aus n Oktaven besteht. Die Oktaven bestehen aus m Intervallen bzw. m verschiedenen Skalierungen. Die Skalenraum Pyramide wird wie schon bei SIFT nach Extremwerten durchsucht, indem jeder Punkt mit seinen 26 Nachbarn verglichen wird (siehe 1.1.1).

Deskriptor

2 Varianten des Deskriptor werden vorgestellt:

- **Variante A:** Die *support region* bzw. Region um einen Feature Punkt, hat eine Größe von 12x12 Pixeln. In diesen Regionen überlagern bzw. überlappen sich die einzelnen Kerne. Wie Abbildung 2.9 zeigt, ist diese Variante gut für Feature Punkte mit komplexen Gradienten im Zentrum der Region und daher gut für Eck-Detektoren.

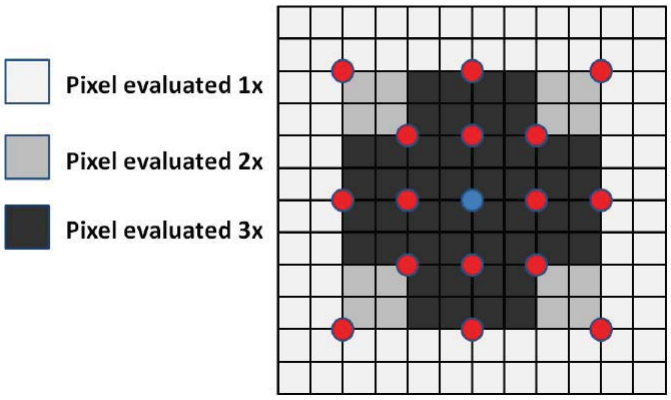

Abbildung 2.9 Die Abbildung zeit einen Feature Punkt (blauer Punkt in der Mitte) mit seiner 12x12 Pixel großen Region. Die roten Punkte stellen die 16 Kerne da, welche eine Größe von 4x4 Pixeln haben und sich in der Abbildung überlagern. Die Region im Zentrum wird 3 mal berechnet, die Pixel am Rand hingegen nur einmal. Das Zentrum wird also stärker gewichtet Zilly et al. (2011).

- **Variante B:** Bei dieser Variante ist die Region 16x16 Pixel groß und die einzelnen Kerne überlagern sich nicht. Variante B ist gut für Feature Punkte mit vielen Gradienten am Rand der Region und somit für blob Detektoren, siehe Abbildung 2.10.

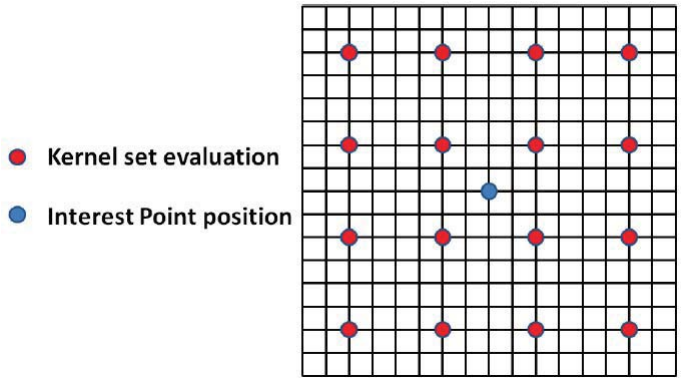

Abbildung 2.10 Hier ist die Region 16x16 Pixel groß. Die roten Punkte stellen wieder die Kerne da, welche sich bei Variante B jedoch nicht überlagern Zilly et al. (2011).

Bei blobs befinden sich die wichtigen Gradienten am Rand der Region und nicht im Zentrum, daher funktioniert Variante B gut mit blob Detektoren.

2.4.2 Weitere Detektoren

- **Harris - Laplace Detektor**
  Dieser Detektor ist invariant gegenüber Rotation und Skalierung und ermittelt Feature Punkte in Ecken, in welchen sich die Intensität des Bildes deutlich in verschiedene Richtungen ändert. Die Feature Punkte werden durch eine angepasste Harris Funktion ermittelt und durch den Gauß-Laplace Operator im Skalenraum ausgewählt Mikolajczyk and Schmid (2004).

- **Hessian - Laplace Detector**
  Dieser Detektor ist ebenfalls invariant gegenüber Rotation und Skalierung. Er ist ähnlich zum oben beschriebenen DoG Detektor, allerdings bei der Lokalisierung der Punkte im Skalenraum genauer als der DoG Detektor und auch genauer bei der Auswahl der Skalierung als der Harris-Laplace Detektor. Feature Punkte werden anhand von lokalen Maxima der Hesse Determinante (Determinante der Hesse Matrix) im Raum lokalisiert und die Skalierung ergibt sich aus lokalen Maxima der Gauß-Laplace-Pyramide Mikolajczyk and Schmid (2004).

- **Harris - Affine Detektor**
  Der Harris-Affine Detektor ist, wie der Name schon sagt, invariant gegenüber affinen Transformationen. Er benutzt den Harris-Laplace Detektor um potentielle Feature Punkte zu ermitteln und erreicht affine Invarianz, indem eine affine Anpassung der Region durchgeführt wird (*affine shape adaption*, siehe Mikolajczyk and Schmid (2004))

- **Hessian-Affine Detektor**
  Dieser Detektor ist ebenfalls affin invariant und unterscheidet sich nur dadurch vom Harris-Affine Detector, dass nicht der Harris-Laplace Detector, sondern der Hessian-Laplace Detektor verwendet wird Mikolajczyk and Schmid (2004).

- **MSER - Maximally stable extremal regions**
  Auch MSER ist affin invariant aber anders als bei den meisten anderen Detektoren, haben MSER Regionen keine geometrische Form wie Rechtecke oder Kreise. Für verschiedene Schwellenwerte wird das Bild neu berechnet. Jeder Pixel des Bildes wird mit dem Schwellenwert verglichen. Ist dieser kleiner als der Schwellenwert, wird die Farbe des Pixels schwarz, ansonsten weiß. So bilden sich nach und nach schwarze Bereiche. In diesen Bereichen befinden sich lokale Intensitätsminima, welche nach und nach miteinander verschmelzen, bis am Ende ein komplett

schwarzes Bild entsteht. Die Vereinigung aller dieser schwarzen Bereiche bzw. Regionen bildet die Menge der *extremal regions*. Diese sind *maximal stabil*, wenn sie sich für eine größere Anzahl an Schwellenwerten, kaum bzw. gar nicht verändern. Morel and Yu (2008)

2.4.3 Weitere Deskriptoren

- **PCA - SIFT:**
  Dieser Deskriptor von Yan Ke und Rahul Sukthankar ist eine Erweiterung von SIFT. PCA steht dabei für *Principal Components Analysis*, wobei es sich um eine Technik zur Reduktion der Dimension handelt. Ein SIFT Feature Vektor besteht aus 128 Elementen. Mit PCA werden die Feature Vektoren auf eine Größe von 36 Elementen reduziert Ke and Sukthankar (2003).

- **GLOH - Gradient Location-orientation histogram:**
  GLOH ist ebenfalls eine Erweiterung von SIFT und wurde von K. Mikolajczyk und C. Schmid entwickelt. Der Deskriptor wird berechnet für 17 Lokalisierungsteile und 16 Teile für die Orientierung, was ein Histogramm von $16 * 17 = 272$ Teilen ergibt. Die Dimension ist also viel höher als bei SIFT und wird deshalb noch mit PCA reduziert. Die Feature Vektoren haben dann am Ende die selbe Dimension wie SIFT, nämlich 128 Mikolajczyk and Schmid (2005).

- **HOG - Histogram of Oriented Gradients**
  Der HOG Deskriptor ist dem SIFT Deskriptor sehr ähnlich, da beide auf Histogrammen lokaler Gradienten basieren. Ein Bild wird in viele kleine rechteckige Teilbereiche unterteilt, welche als '*cells*' bezeichnet werden. Für jedes Pixel einer jeden Zelle wird ein Histogramm von Gradientenrichtungen berechnet. Die vereinigten Einträge dieser Histogramme bilden dann den Deskriptor. Durch Kontrast - Nomalisierung der lokalen Histogramme wird die Invarianz gegenüber Beleuchtung erhöht Dalal and Triggs (2005).

Kapitel 3

Matching

In diesem Kapitel geht es darum, die gefundenden Feature Punkte der verschiedenen Bilder miteinander zu vergleichen bzw. zu matchen. Es werden verschiedende *Matching Strategien* kurz vorgestellt. Wurden in beiden zu vergleichenden Bildern die Deskriptoren bzw. Feature Vektoren berechnet, müssen diese verglichen bzw. gematcht werden, um korrespondierende Punkte zu ermitteln. Beim matchen wird jeder Deskriptor von Bild A mit jedem Deskriptor des anderen zweiten Bildes B verglichen. Wann ein Match akzeptiert wird, hängt von der jeweiligen *matching strategy* ab. Im folgenden werden verschiedene Strategien kurz beschrieben.
Im einfachsten Fall wird die euklidische Distanz zwischen 2 Deskriptoren D_A und D_B gebildet und der Match akzeptiert, wenn diese Distanz bzw. Differenz kleiner als ein bestimmter Schwellenwert ist, welcher vorher festgelegt werden muss. Bei diesem Verfahren ist es möglich, dass ein einzelner Deskriptor mehrere Matches hat, wovon auch mehrere richtig sein können.
Im Fall von *nearest neighbor matching*, wird ein Match nur dann akzeptiert, wenn der Deskriptor D_B der nächste Nachbar (nearest neighbor) von D_A ist, und die Distanz bzw. Differenz zwischen beiden unter einem Schwellenwert ist. Der nächste Nachbar von einem Deskriptor D_B ist dabei der Deskriptor, welche den kleinsten euklidischen Abstand zu D_B hat. Bei diesem Ansatz kann ein einzelner Deskriptor nur einen einzigen Match haben.
Ein weiterer, ähnlicher Ansatz zum nearest neighbor matching, betrachtet nicht nur den nächsten Nachbarn, sondern auch noch den zweit nächsten Nachbarn. Sei D_B der nächste Nachbar zu D_A und D_C der zweit nächste Nachbar und *distRatio* der Schwellenwert, dann wird der Match akzeptiert wenn gilt:

$$\frac{||D_A - D_B||}{||D_A - D_C||} < distRatio \tag{3.1}$$

oder in anderer Schreibweise

$$distRatio * ||D_A - D_B|| < ||D_A - D_C|| \tag{3.2}$$

Dieser dritte Ansatz ist deshalb so gut, weil bei korrekten Matches der Abstand zum nächsten Nachbarn sehr viel kleiner sein sollte, als der Abstand zum zweit nächsten Nachbarn. Bei falschen Matches kommt es oft vor, dass viele Nachbarn einen sehr ähnlichen Abstand haben. Je kleiner distRatio gewählt wird, desto mehr Matches werden gefunden. Wählt man distRatio allerdings zu klein, werden viele falsche Matches dabei sein. Es stellt sich also die Frage ob man viele Matches haben möchte und somit auch viele falsche Matches in Kauf nimmt oder ob man lieber wenige Matches möchte und dafür möglichst nur korrekte Matches erhält, allerdings auch viele andere korrekte Matches verpasst. Lowe schlägt einen Wert von distRatio von 0.8 vor, welcher laut Lowe 90% der falschen Matches eliminiert und nur 5% der korrekten Matches ausschließt.
Die in Kapitel 1 beschriebenen Verfahren bringen alle einen eigenen Matcher mit. In dieser Arbeit wurden jedoch alle Matches mit dem Matcher von vlfeat berechnet, damit sich die Ergebnisse besser vergleichen lassen. Vlfeat verwendet den 3. Ansatz und setzt den Parameter distRatio standardmäßig auf 1.5.

Kapitel 4

Testdaten

Für alle Untersuchungen in dieser Arbeit wurde ein MultiViewVideo Datensatz als Grundlage genommen INRIA (2007). Der Datensatz besteht aus Bildern von 8 Kameras mit jeweils 500 Frames.

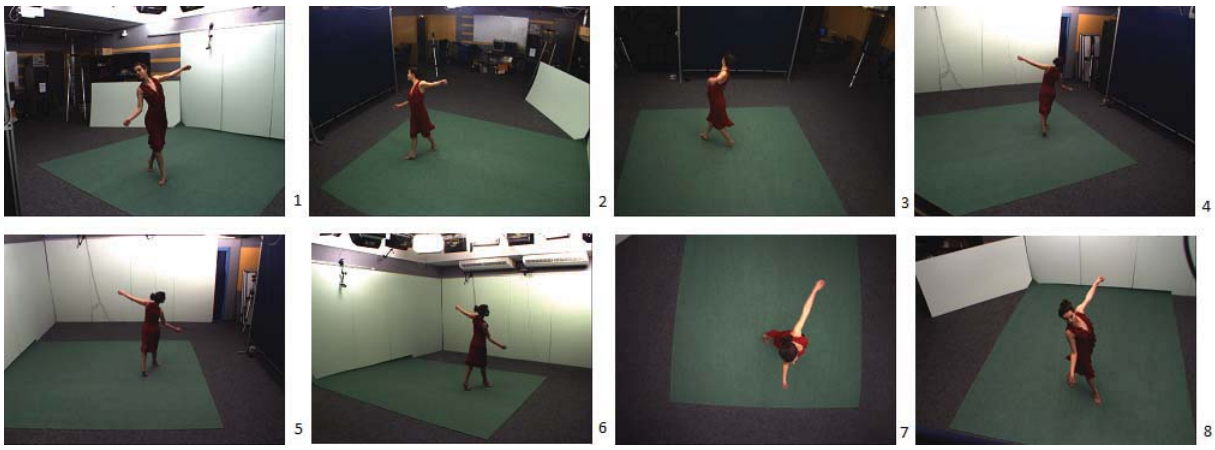

Abbildung 4.1 Die Abbildung zeigt das erste Bild des Datensatzes aller 8 Kameras

Die Bilder wurden nach den beschriebenen Verfahren nach Feature Punkten durchsucht und gegeneinander gematcht. Da die Kameras eine feste Position haben und sich die einzelnen Bilder einer Kamera dadurch nur durch die Position der Tänzerin unterscheiden, werden Matches hauptsächlich im Hintergrund gefunden. Diese sind aber nicht von Interesse und müssen heraus gefiltert werden, sodass nur Matches der Tänzerin übrig bleiben. Zu dieser Filterung werden Segmentierungsmasken eingesetzt.

Folgende Szenarien sind zu untersuchen:

Abbildung 4.2 Segmentierungsmaske des 1.Bildes der 1. Kamera

- **Szenario 1:** Bilder der selben Kamera, welche einen kleinen Abstand von nur 1, 3 oder 5 Frames haben.
- **Szenario 2:** Bilder der selben Kamera, welche einen größeren Abstand von 10, 20 oder 50 Frames haben.
- **Szenario 3:** Bilder verschiedener Kameras zum gleichen Zeitpunkt
- **Szenario 4:** Bilder verschiedener Kameras zu verschiedenen Zeitpunkten

Kapitel 5

Ergebnisse

In diesem Kapitel werden die Ergebnisse der einzelnen Verfahren in den verschiedensten Konfigurationen in den oben beschriebenen 4 Szenarien vorgestellt und miteinander verglichen.

5.1 Szenario 1

Zunächst werden 2 aufeinanderfolgende Bilder einer Kamera gegeneinander gematcht, in diesem Fall die ersten beiden Bilder der Kamera 0.
Die folgenden Abbildungen zeigen die Ergebnisse aller oben beschriebenen Verfahren in default Konfiguration (alle oben beschriebenen Parameter auf default Wert). Verwendet wurde der Matcher von vlfeat Vedaldi and Fulkerson (2011). Werden mehr als 50 Matches gefunden, dann werden nur die 50 'besten' Matches auf Korrektheit überprüft. Dazu werden alle Matches nach ihrem quadrierten euklidischen Abstand sortiert und die 50 mit dem kleinsten Abstand überprüft. Das bedeutet aber nicht, dass die 50 Matches mit dem kleinsten euklidischen Abstand, alle korrekt sind. (siehe 6.1)

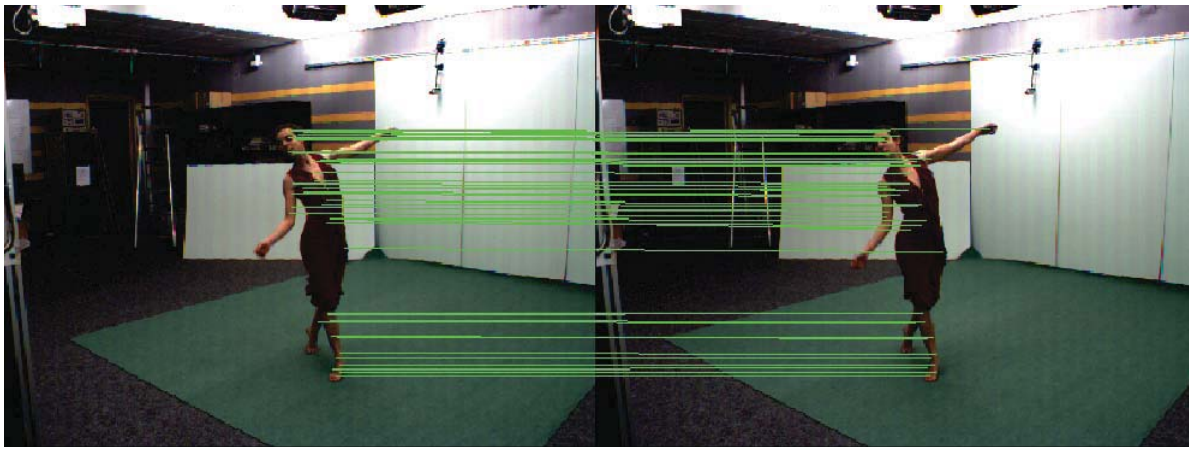

Abbildung 5.1 SIFT in default Konfiguration: 45 Matches werden gefunden, wovon alle 45 Matches korrekt sind

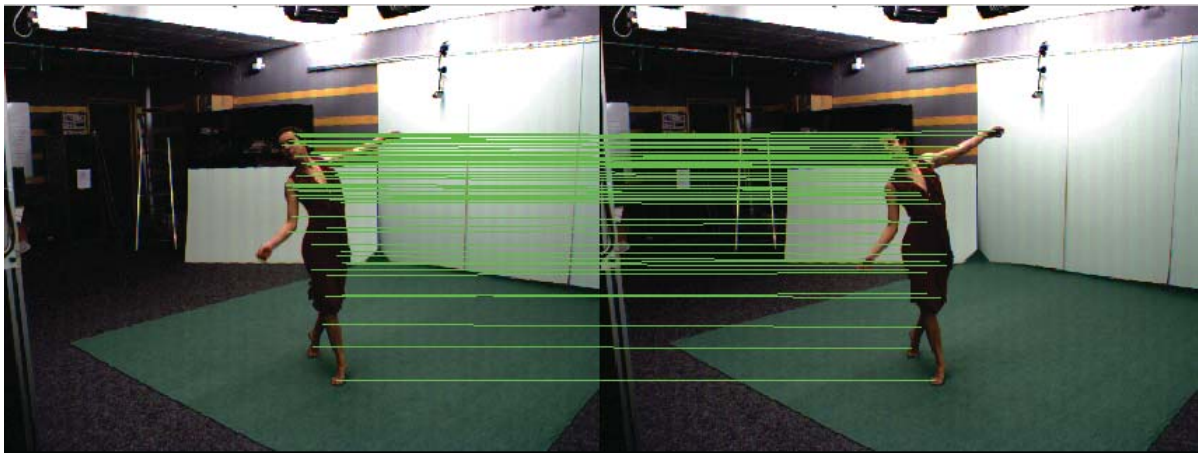

Abbildung 5.2 SURF in default Konfiguration: 54 Matches werden gefunden, wovon alle 54 Matches korrekt sind

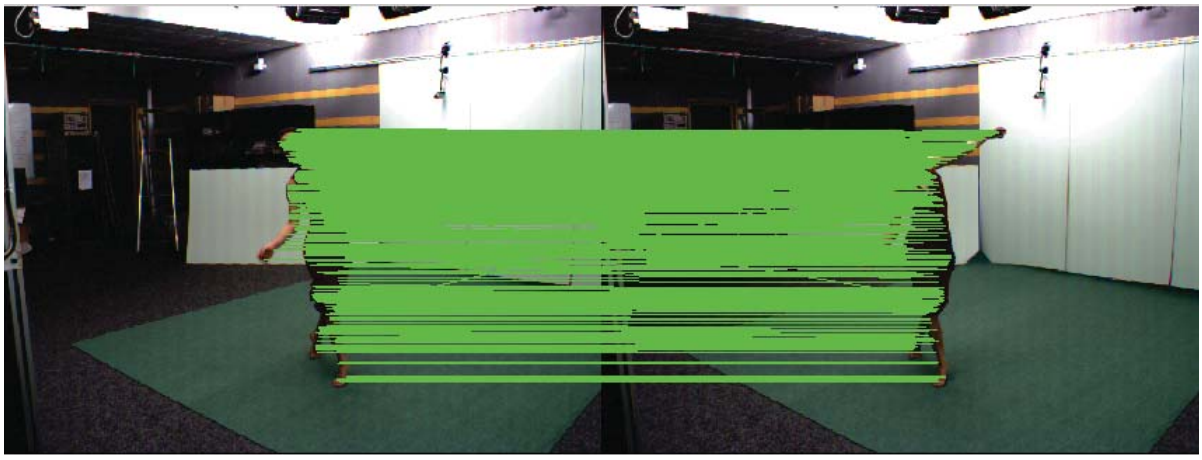

Abbildung 5.3 ASIFT in default Konfiguration: 983 Matches werden gefunden, wovon die 50 besten Matches auf Korrektheit überprüft wurden. Alle 50 sind korrekt.

Im folgenden wird ein Frame übersprungen und das 1. Bild gegen das 3. Bild von Kamera 0 gematcht.

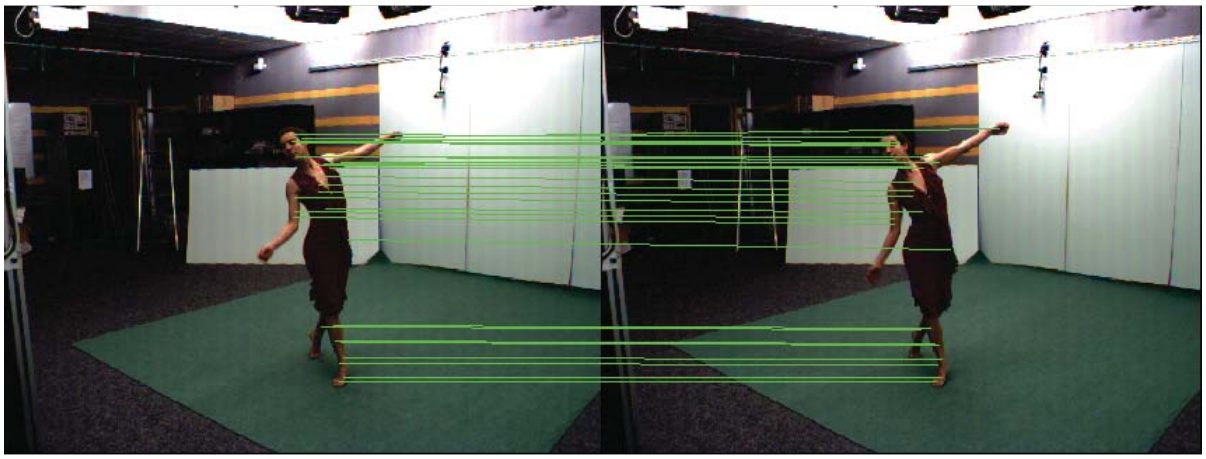

Abbildung 5.4 SIFT in default Konfiguration: 29 Matches werden gefunden, wovon alle 29 Matches korrekt sind

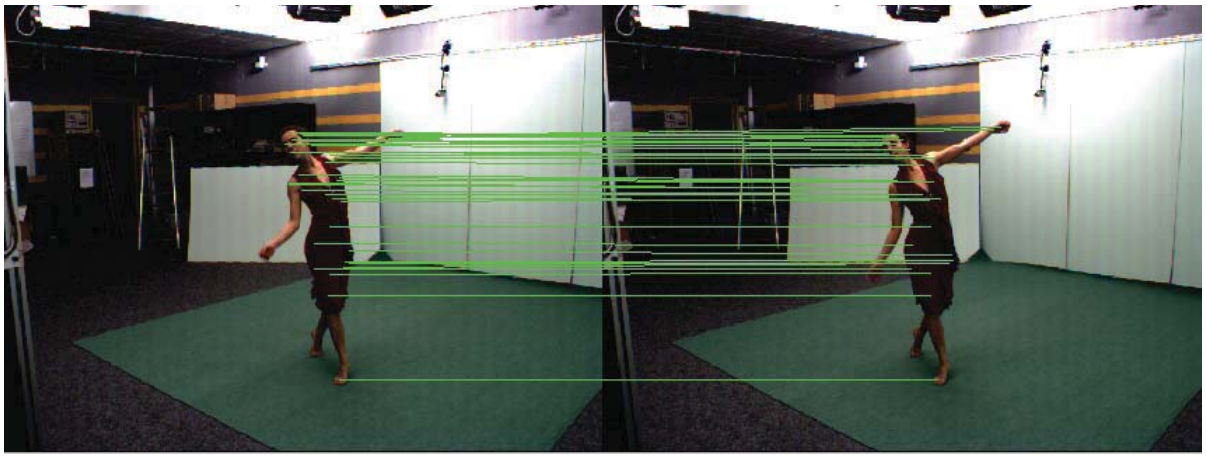

Abbildung 5.5 SURF in default Konfiguration: 34 Matches werden gefunden, wovon alle 34 Matches korrekt sind

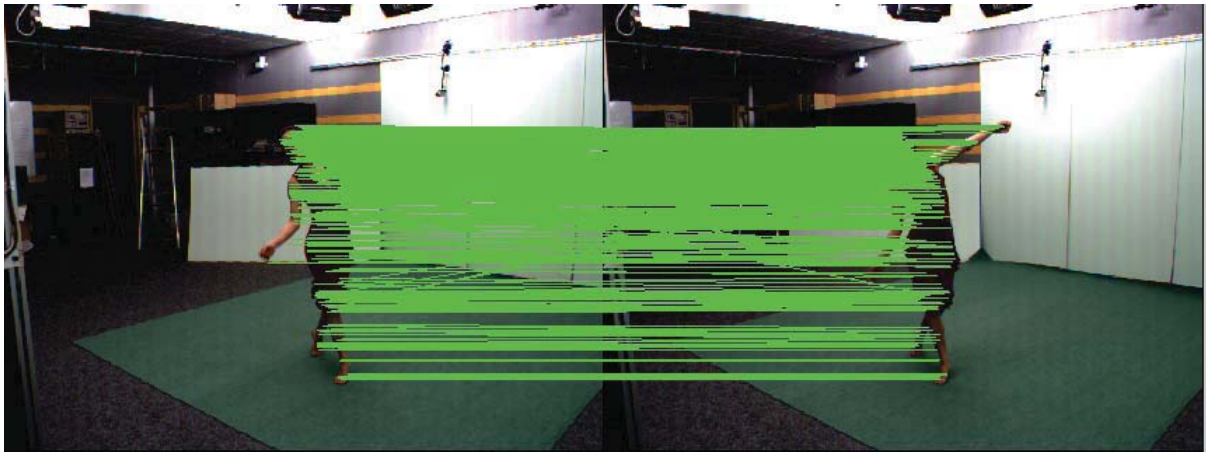

Abbildung 5.6 ASIFT in default Konfiguration: 594 Matches werden gefunden, wovon die 50 besten Matches korrekt sind.

Auch hier liefern alle Verfahren noch fehlerfreie Ergebnisse in default Konfiguration. Überspringt man nochmal 2 Frames, vergleicht also Bild 1 mit Bild 5, erhält man folgende Ergebnisse:

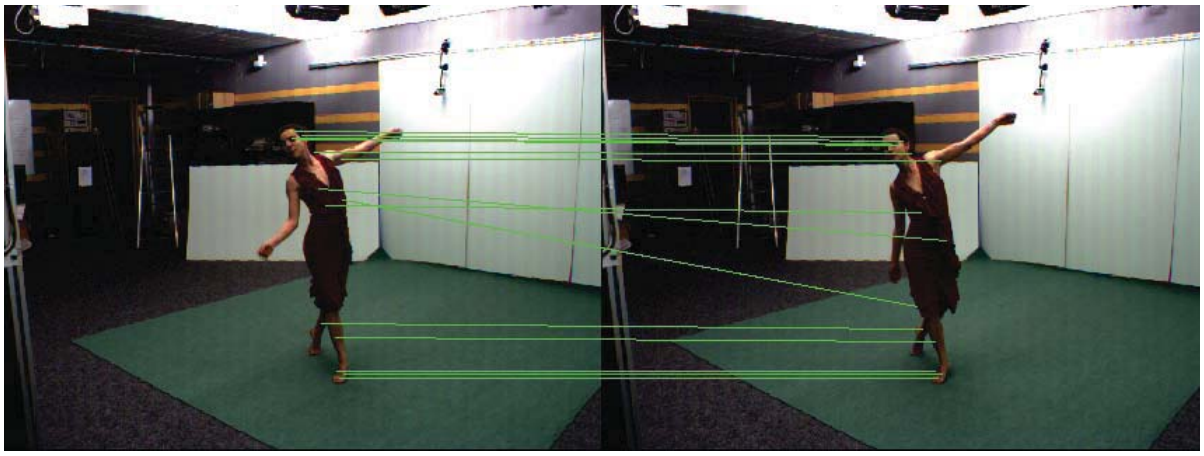

Abbildung 5.7 SIFT in default Konfiguration: 15 Matches werden gefunden, wovon 12 Matches korrekt sind

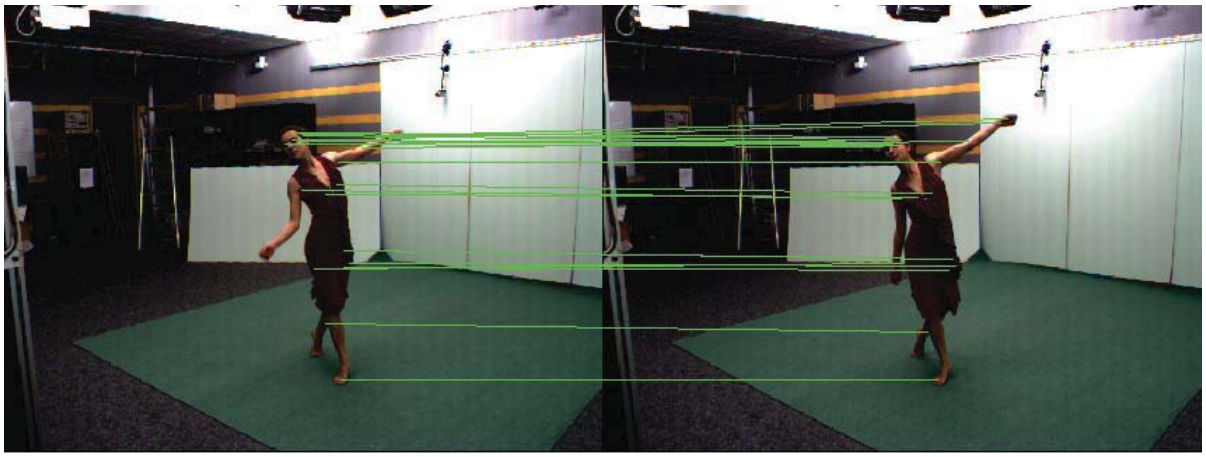

Abbildung 5.8 SURF in default Konfiguration: 20 Matches werden gefunden, wovon 19 Matches korrekt sind

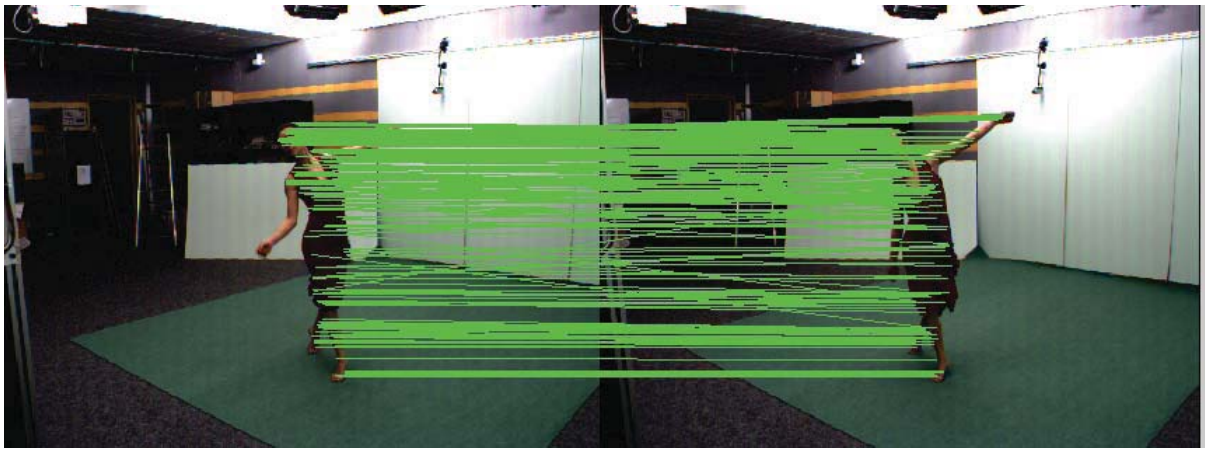

Abbildung 5.9 ASIFT in default Konfiguration: 273 Matches werden gefunden, wovon 44 der 50 besten Matches korrekt sind.

Die Anzahl der Matches wurde wieder deutlich kleiner und es sind bei allen Verfahren nicht mehr alle Matches korrekt, die fehlerhaften Matches halten sich allerdings in Grenzen. Möchte man die Anzahl der gefundenen Feature Punkte erhöhen, hat man bei SIFT verschiedene Möglichkeiten. Einmal kann man die die Anzahl der 'Level' erhöhen z.b. auf 50, oder den 'edge thresh' erhöhen , dann werden weniger Punkte auf Kanten heraus gefiltert und man erhält mehr Feature Punkte. Des Weiteren führt ein höherer 'magnification factor' zu mehr Feature Punkte im Bereich der Tänzerin, da dadurch die Bildregion vergrößert wird. Verringert man die 'match ratio', werden natürlich sehr viel mehr Matches gefunden, wovon aber viele Matches falsch sein können. Ein hoher 'Level' Wert von 50 führt allerdings zu einer deutlich längeren Berechnungszeit.

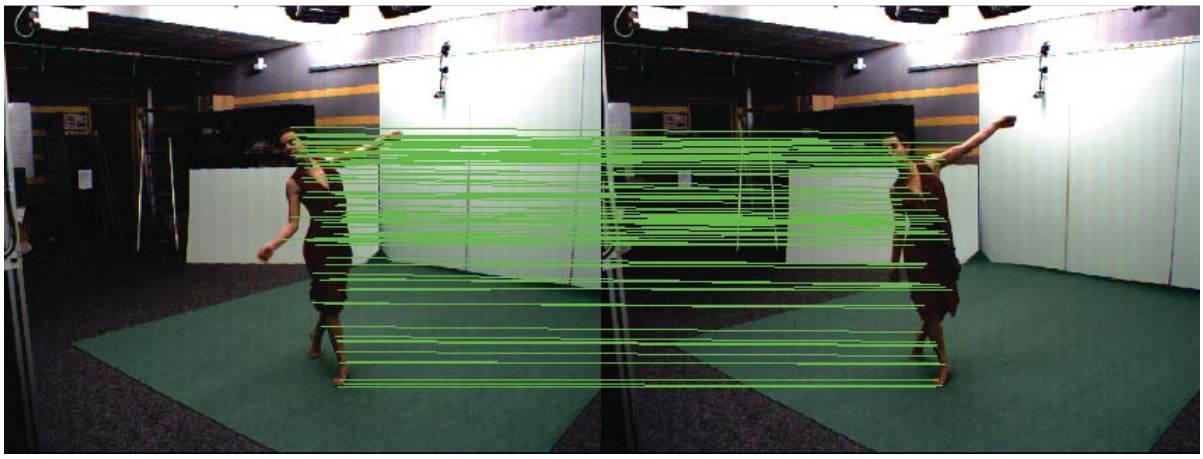

Abbildung 5.10 SIFT mit Levels = 50, magnif = 8 und edge thresh = 100: 88 Matches werden gefunden, wovon 44 der 50 besten Matches korrekt sind.

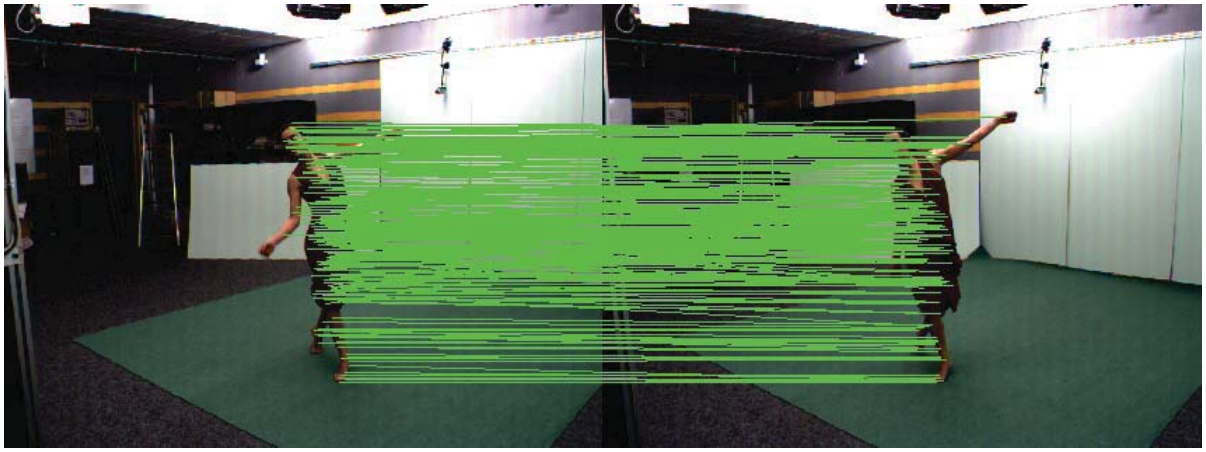

Abbildung 5.11 SIFT mit Levels = 50, magnif = 8 und edge thresh = 100 und match ratio = 1: 310 Matches werden gefunden, wovon 42 der 50 besten Matches korrekt sind.

Erhöht man bei SURF die Anzahl der Oktaven und die Anzahl der Ebenen und setzt den 'hessianThreshold' runter, findet SURF ein paar Matches mehr, wovon fast alle korrekt sind. Setzt man dazu die 'match ratio' auf 1, werden deutlich mehr Matches gefunden, allerdings auch viele falsche.

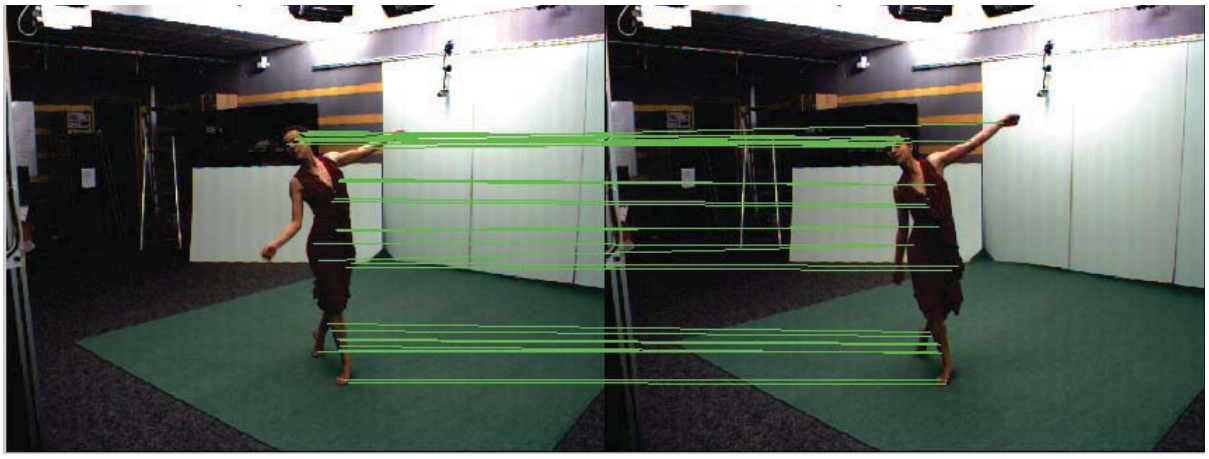

Abbildung 5.12 SURF mit doppelter Oktaven Anzahl und doppelter Anzahl von Ebenen pro Oktave und hessianThreshold = 5 : 32 Matches werden gefunden, wovon 31 korrekt sind.

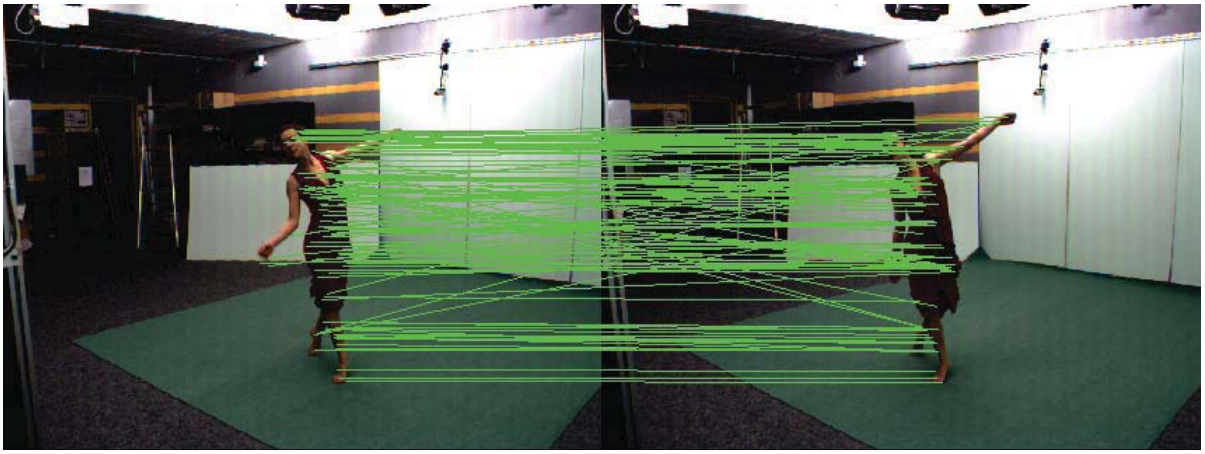

Abbildung 5.13 SURF mit doppelter Oktaven Anzahl und doppelter Anzahl von Ebenen pro Oktave und hessianThreshold = 5, match ratio = 1: 126 Matches werden gefunden, wovon 38 der 50 besten korrekt sind.

ASIFT lifert bereits in default Konfiguration sehr gute Ergebnisse, sodass die Parameter in diesem Fall nicht optimiert werden müssen. In den folgenden Abbildungen werden kurz die Ergebnisse der oben beschriebenen weiteren Detektoren und Deskriptoren betrachtet. Als Testbilder dienen wieder die Bilder 1 und 5 der Kamera 1.

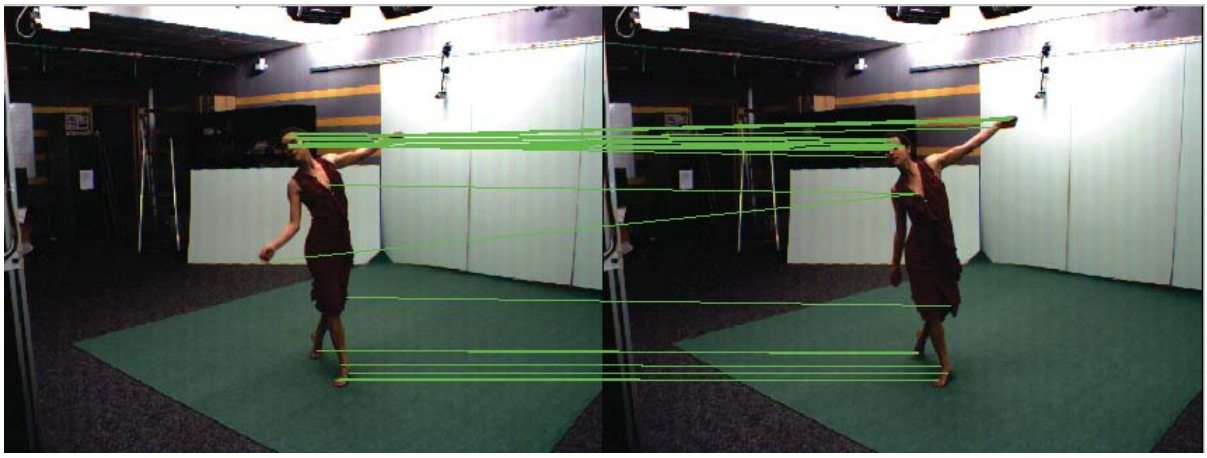

Abbildung 5.14 Harris-Affine Detektor mit Sift Deskriptor: 48 Matches werden gefunden, davon 47 korrekt.

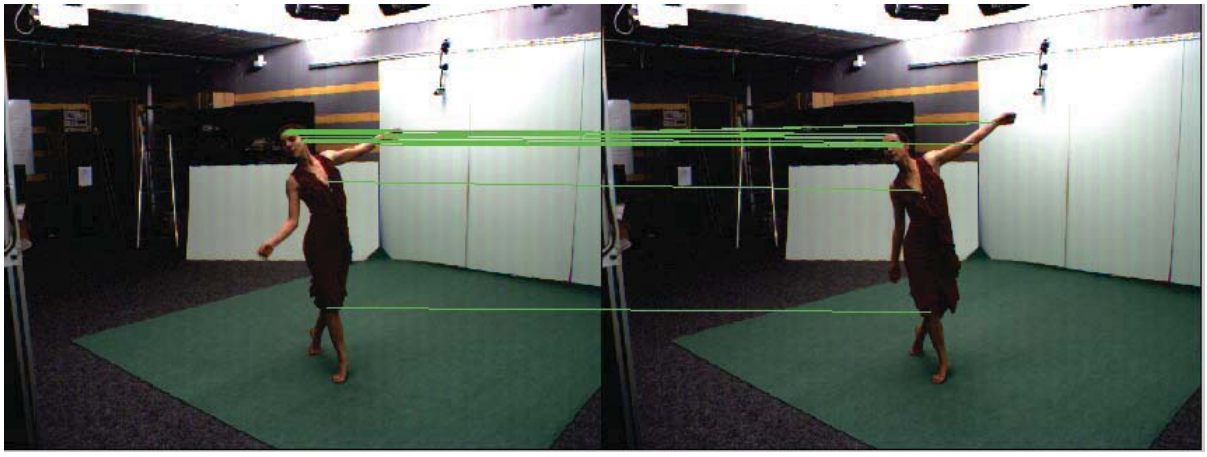

Abbildung 5.15 Harris-Affine Detektor mit GLOH Deskriptor: 24 Matches werden gefunden, davon alle 24 korrekt.

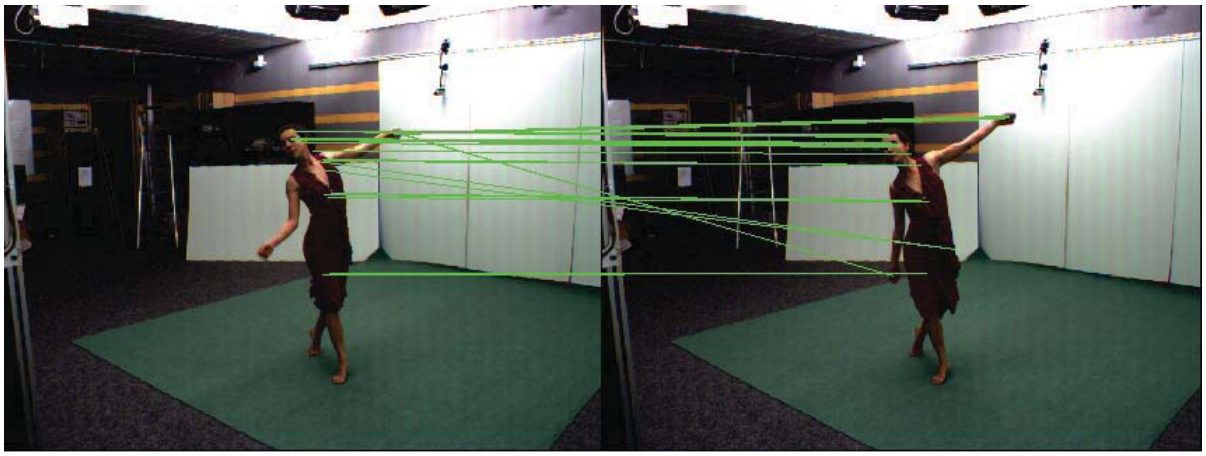

Abbildung 5.16 Hessian-Affine Detektor mit SIFT Deskriptor: 47 Matches werden gefunden, davon 42 korrekt.

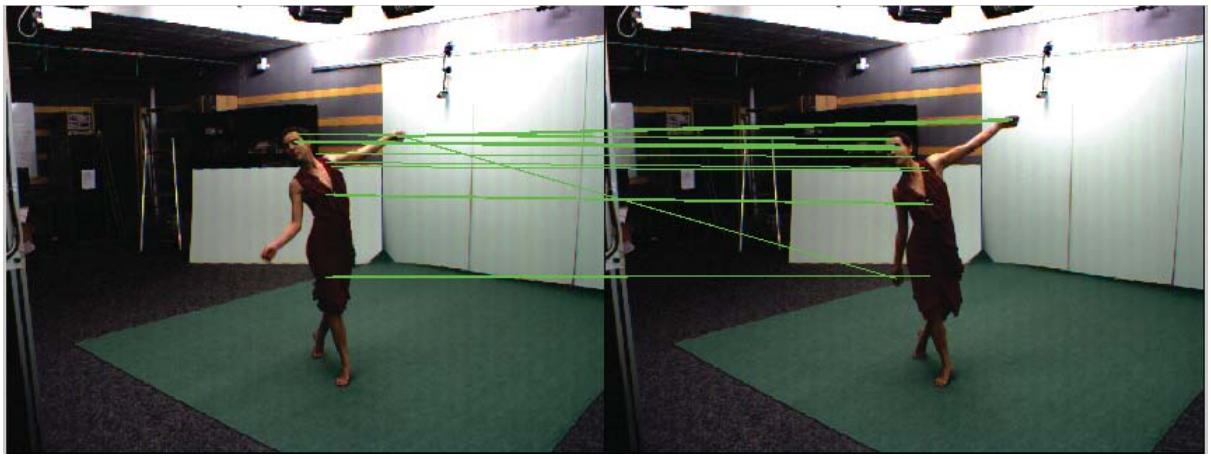

Abbildung 5.17 Hessian-Affine Detektor mit GLOH Deskriptor: 37 Matches werden gefunden, davon 34 korrekt.

5.2 Szenario 2

In diesem Abschnitt werden größere Abstände einer Kamera betrachtet. Abbildung 5.18 zeigt die Bilder 1, 10, 20 und 50 der Kamera 0.

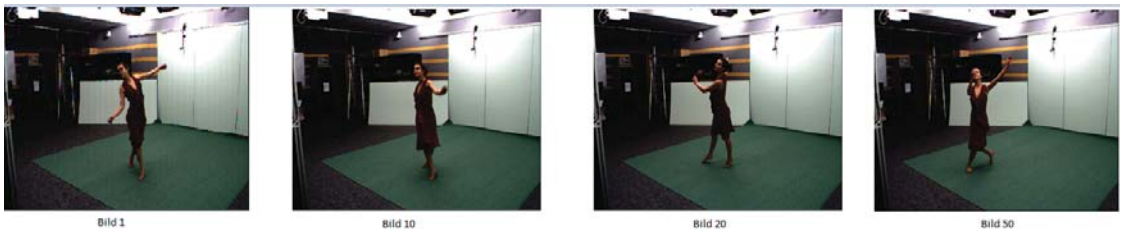

Abbildung 5.18 Die Bilder 1, 10, 20 und 50 der Kamera 0. Bild 1 soll gegen die anderen 3 gematcht werden.

Es fällt auf, dass die Bilder sehr unterschiedlich sind, wenn man z.B. die Arme der Tänzerin betrachtet. Je unterschiedlicher die Bilder sind, umso schwieriger ist es natürlich korrespondierende Punkte zu finden. Auch die Frage, wann ein Match korrekt ist, ist schwerer zu beantworten. Handelt es sich um den Match in folgender Abbildung um einen korrekten Match oder nicht?

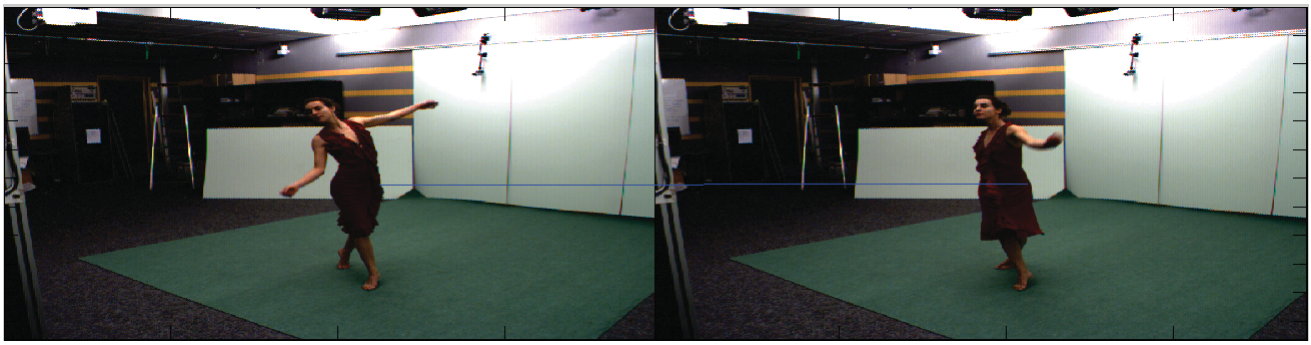

Abbildung 5.19 Ist dieser Beispielmatch korrekt?

In dieser Arbeit wurde die Frage mit ja beantwortet. Ob ein Match nun tatsächlich korrekt ist, liegt also oftmals im Auge des Betrachters. Zunächst wird in den folgenden Abbildungen jeweils Bild 1 gegen Bild 10 gematcht.

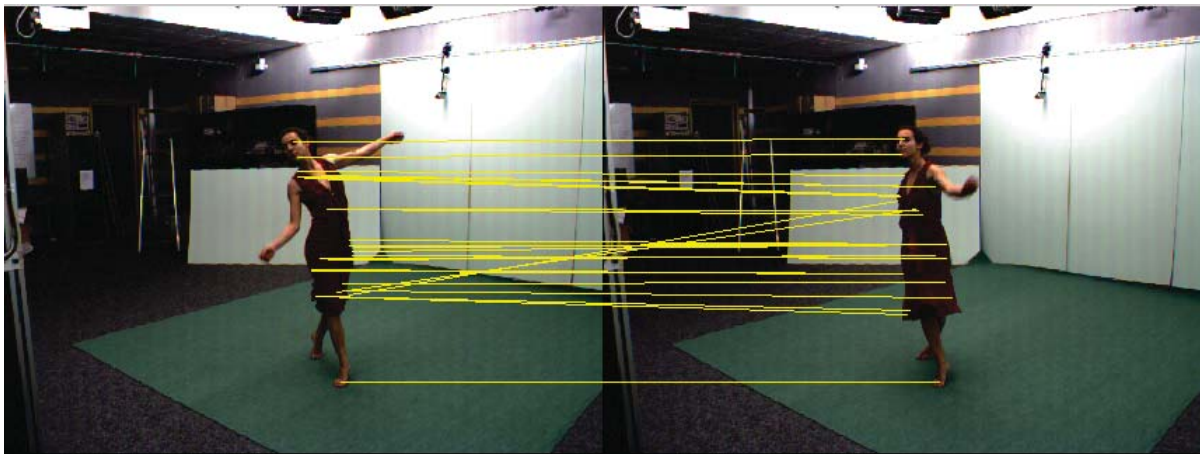

Abbildung 5.20 SIFT mit Levels = 10, magnif = 8 ,edge thresh = 100 und match ratio = 1.3: 28 Matches werden gefunden, wovon 15 Matches korrekt sind

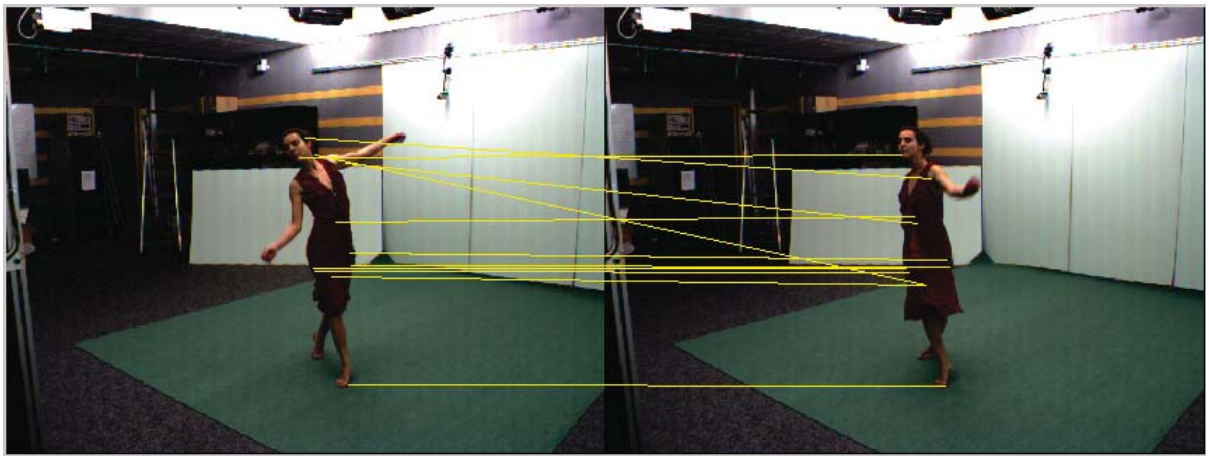

Abbildung 5.21 SURF mit doppelter Oktaven Anzahl und doppelter Anzahl von Ebenen pro Oktave,hessianThreshold = 5 und match ratio = 1.3: 13 Matches werden gefunden, wovon 8 Matches korrekt sind

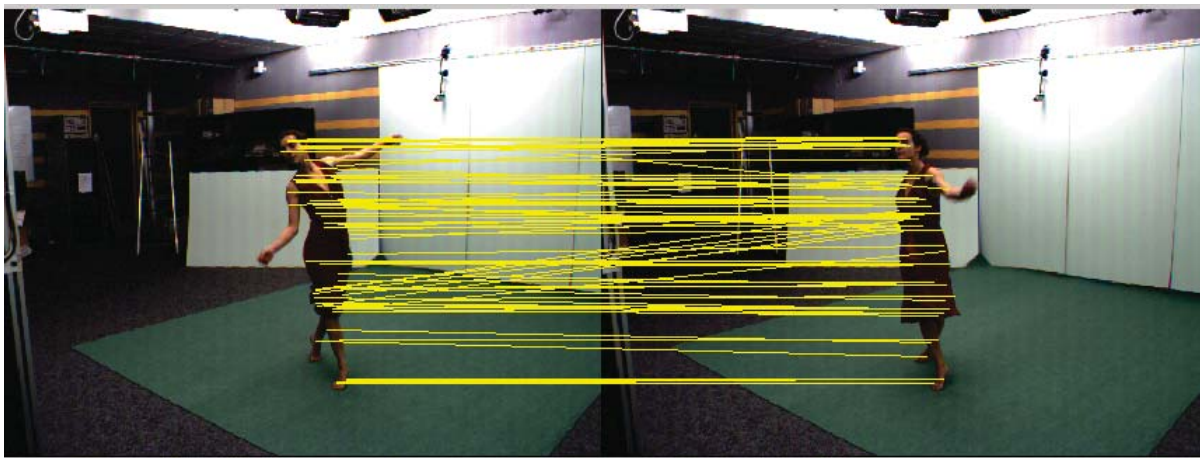

Abbildung 5.22 ASIFT in default Konfiguration: 76 Matches werden gefunden, wovon 20 der 50 besten Matches korrekt sind.

Erhöht man bei ASIFT die Anzahl der Level und den *magnification factor* wie oben bei SIFT, werden mehr Matches gefunden.

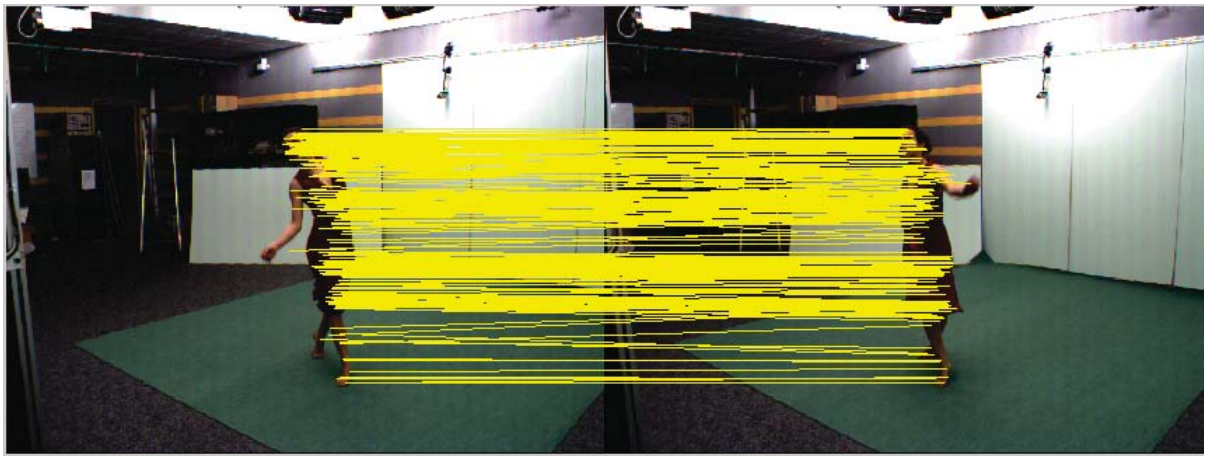

Abbildung 5.23 ASIFT mit magnif=8 ‚OriHistThresh = 1 und scales=10: 277 Matches werden gefunden, wovon 23 der 50 besten Matches korrekt sind.

Die beiden Bilder 1 und 20 sind schon verschieden, dass ein sinnvolles Matchen eigentlich nicht mehr durchgeführt werden kann bzw. keine Aussage über die Korrektheit der Matches getroffen werden kann.

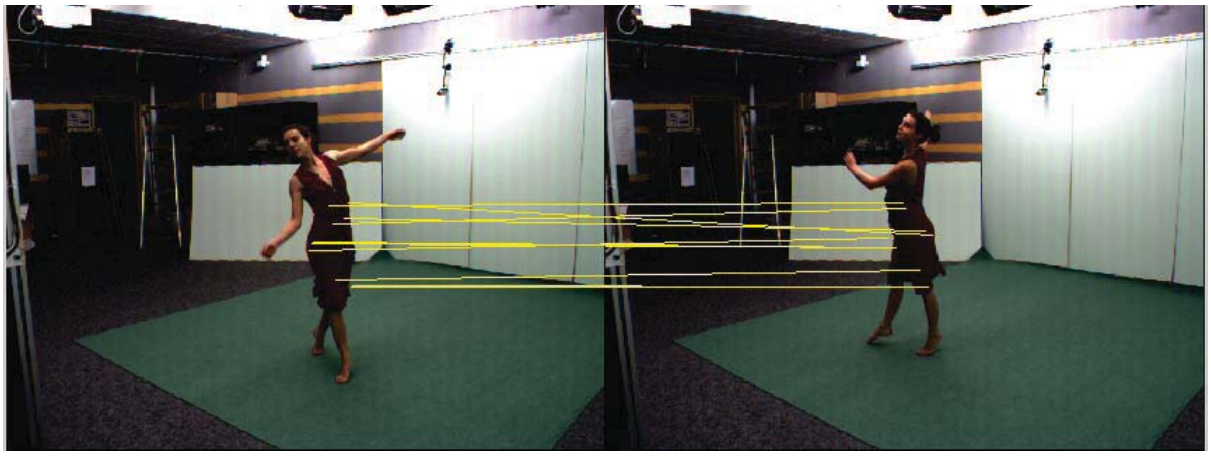

Abbildung 5.24 SIFT mit Levels = 10, magnif = 8 ,edge thresh = 100 und match ratio = 1.3: 11 Matches werden gefunden

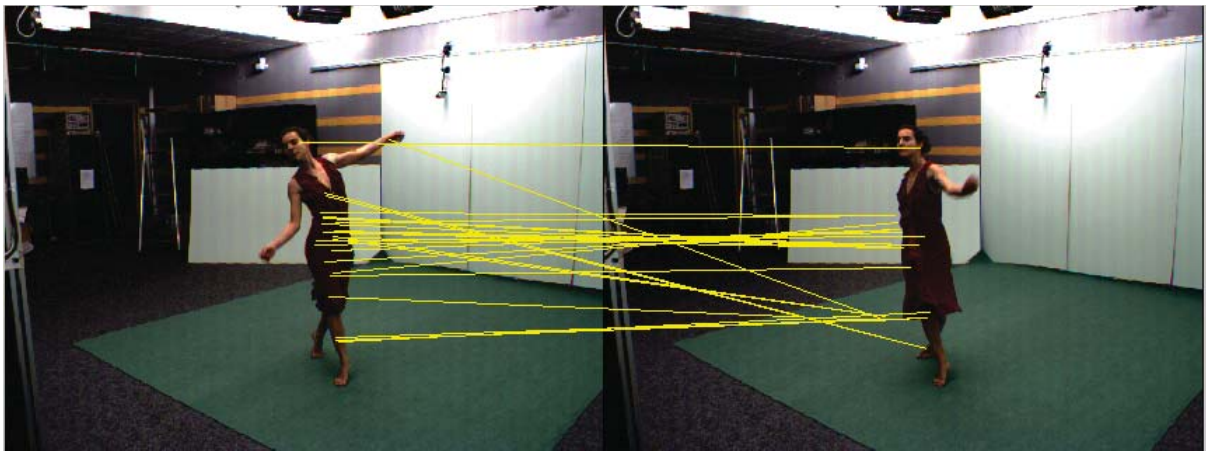

Abbildung 5.25 SURF mit doppelter Oktaven Anzahl und doppelter Anzahl von Ebenen pro Oktave,hessianThreshold = 5 und match ratio = 1.3: 27 Matches werden gefunden

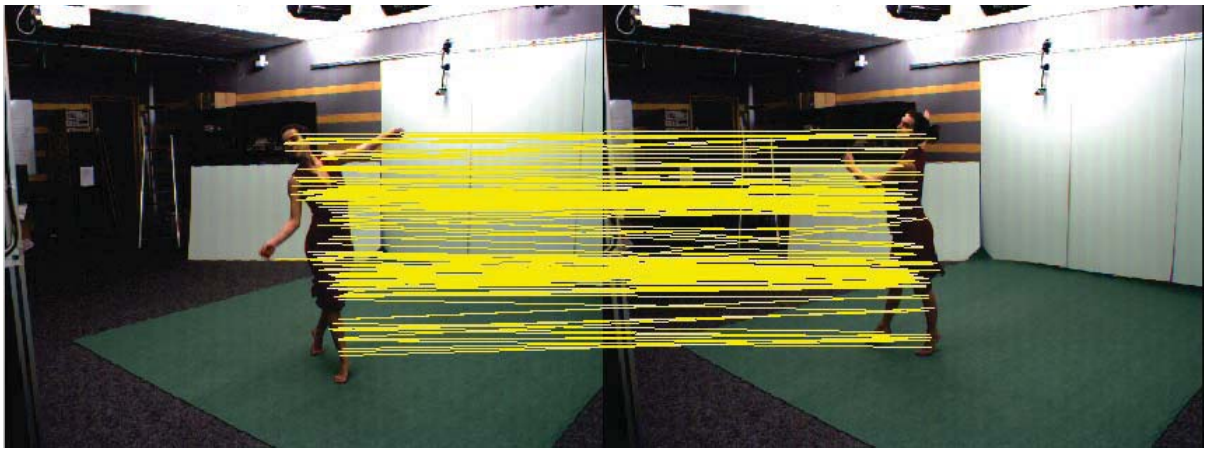

Abbildung 5.26 ASIFT mit magnif=8 ,OriHistThresh = 1 und scales=10: 147 Matches werden gefunden

5.3 Szenario 3

Nachdem bisher nur Bilder der selben Kamera gegeneinander gematcht wurden, werden jetzt gleichzeitige Bilder verschiedener Kameras gematcht. Dabei kommt es sehr darauf an, welche 2 Kameras betrachtet werden. Abbildung 3.1 zeigt Bild 1 aller 8 Kameras. Es macht zum Beispiel keinen Sinn Kamera 1 und Kamera 5 zu nehmen, da die Tänzerin einmal von vorne und einmal von hinten zu sehen ist. Am erfolgversprechendsten sehen die Kamera Paare 1 und 8 sowie 4 und 5 aus. Im folgenden werden daher Bilder der Kameras 1 und 8 gegeneinander gematcht.

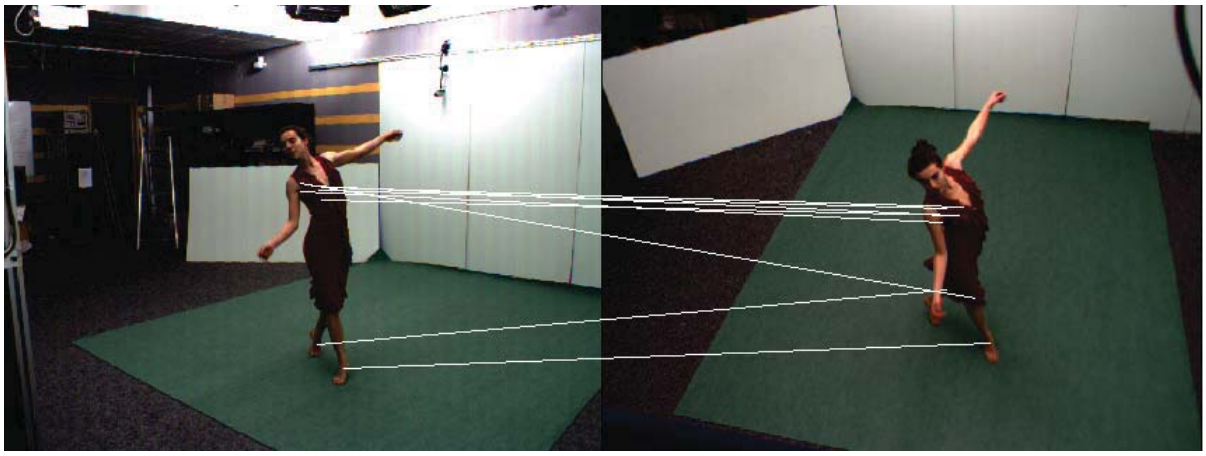

Abbildung 5.27 SIFT mit Levels = 10, magnif = 8 ,edge thresh = 100 und match ratio = 1.3: 8 Matches werden gefunden, wovon 6 korrekt sind

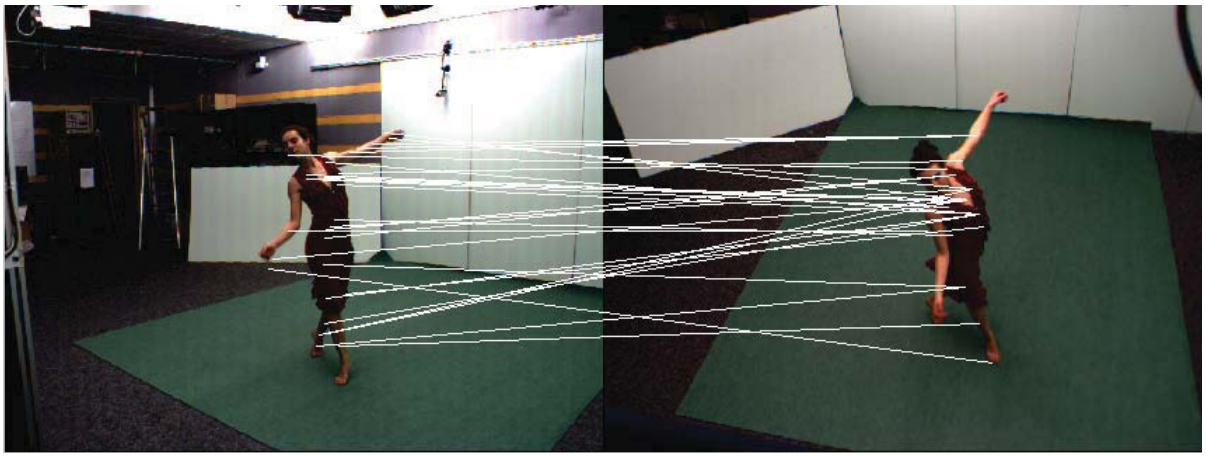

Abbildung 5.28 SURF mit doppelter Oktaven Anzahl und doppelter Anzahl von Ebenen pro Oktave,hessianThreshold = 5 und match ratio = 1.3: 29 Matches werden gefunden, wovon 10 korrekt sind

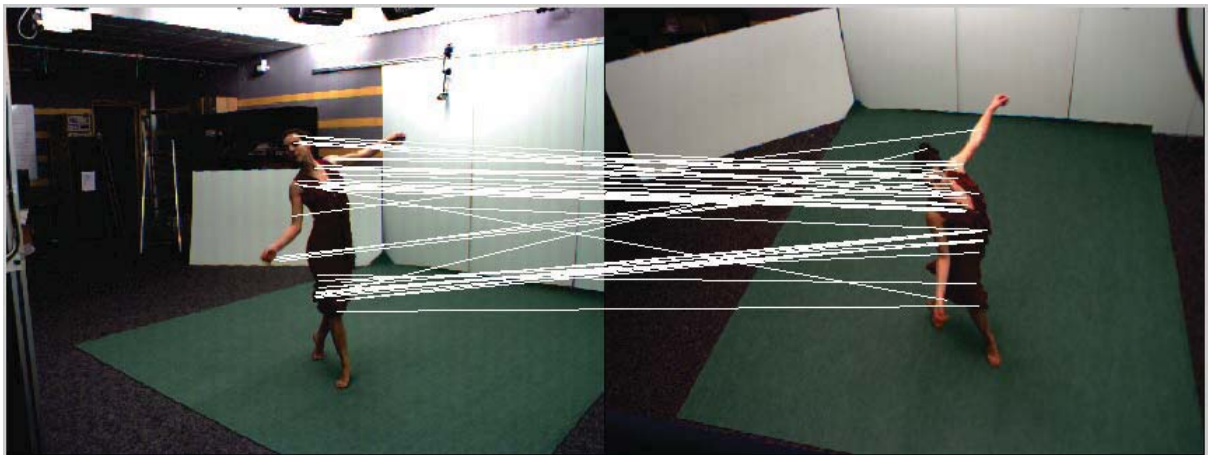

Abbildung 5.29 ASIFT mit magnif=3 ‚OriHistThresh = 1 und scales=3: 47 Matches werden gefunden, wovon 24 korrekt sind

Alle Verfahren finden also sehr viel weniger Matches. Verringert man den *match ratio* Parameter, werden zwar sehr viel mehr Matches gefunden, allerdings mit einer sehr großen Fehlerrate.
Im folgenden noch eine kurze Betrachtung der Ergebnisse in diesem Szenario der weiteren Verfahren.

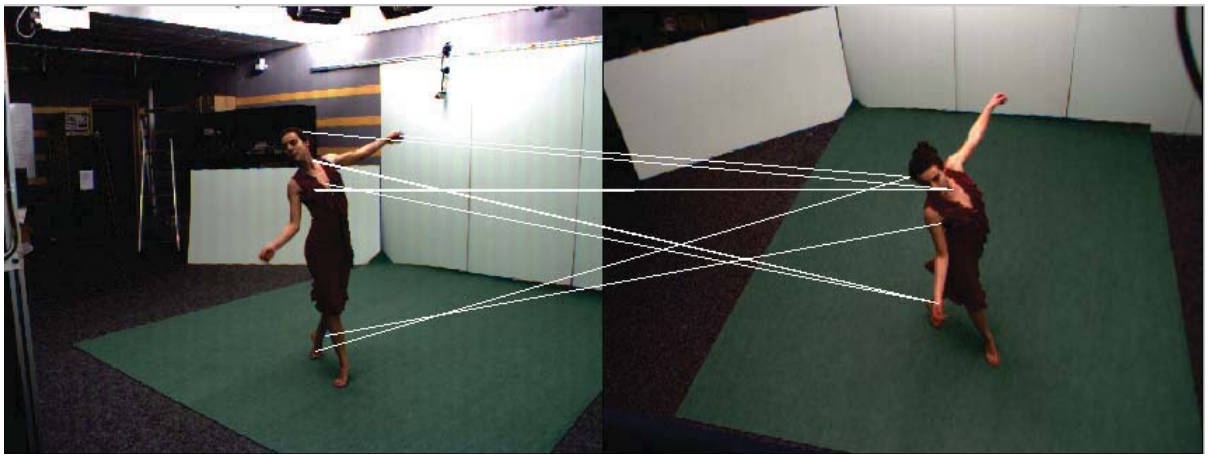

Abbildung 5.30 Harris-AFfine Detektor mit Sift Deskriptor: 12 Matches werden gefunden, davon alle 12 falsch.

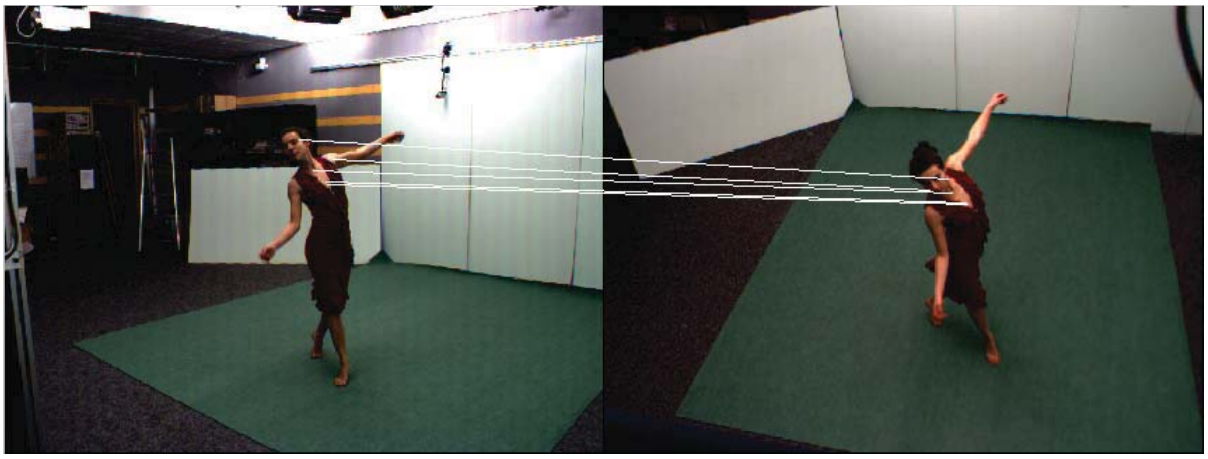

Abbildung 5.31 Harris-AFfine Detektor mit GLOH Deskriptor: 12 Matches werden gefunden, davon 9 korrekt.

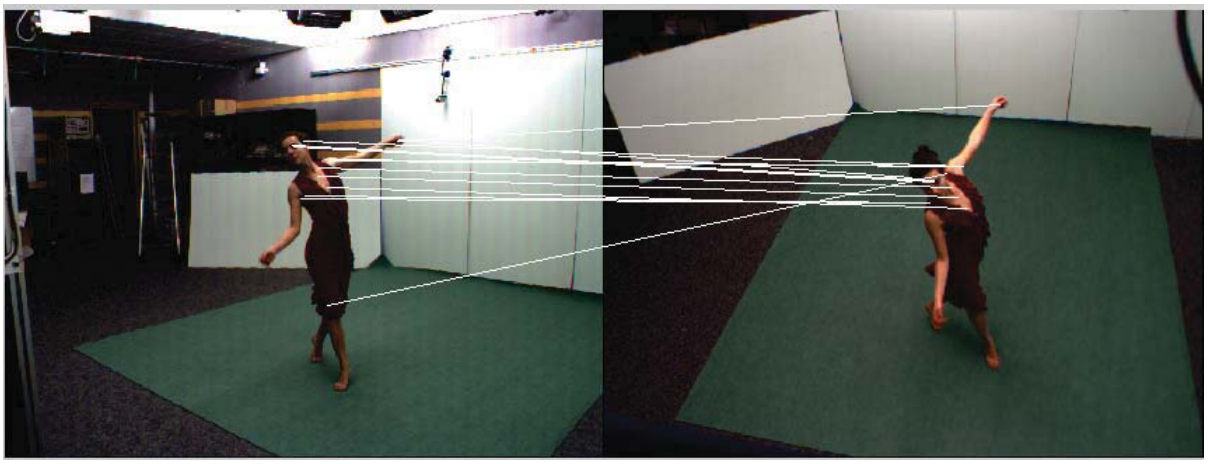

Abbildung 5.32 Hessian-AFfine Detektor mit SIFT Deskriptor: 16 Matches werden gefunden, davon 10 korrekt.

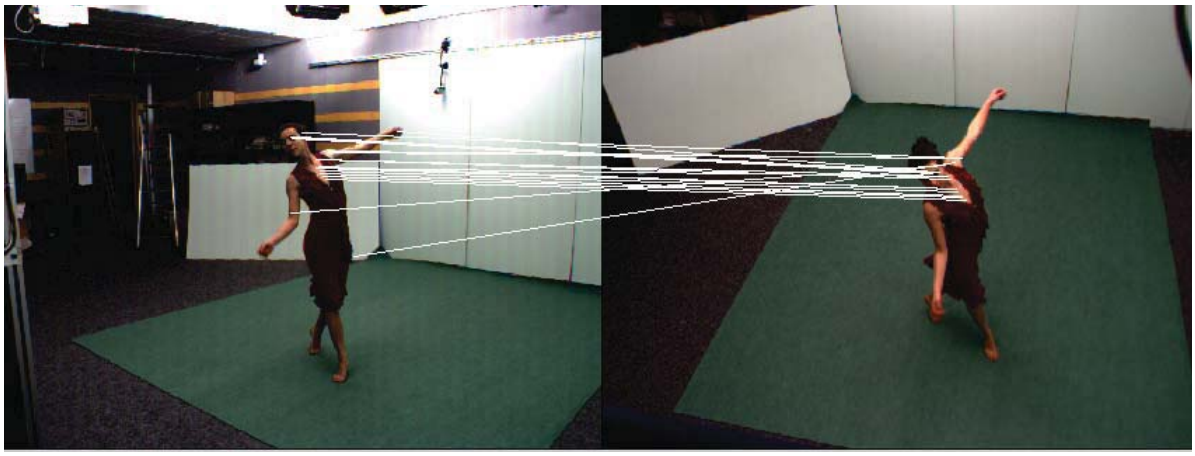

Abbildung 5.33 Hessian-AFfine Detektor mit GLOH Deskriptor: 22 Matches werden gefunden, davon 18 korrekt.

5.4 Szenario 4

In diesem Szenario werden Bilder verschiedener Kameras zu verschiedenen Zeitpunkten gematcht. Auch hier ist klar, je unterschiedlicher die Kameraperspektiven sind und je größer die Zeitsprünge sind, umso schlechter werden die Ergebnisse. Zunächst wird Bild 1 von Kamera 1 gegen das Bild 3 von Kamera 8 gematcht.

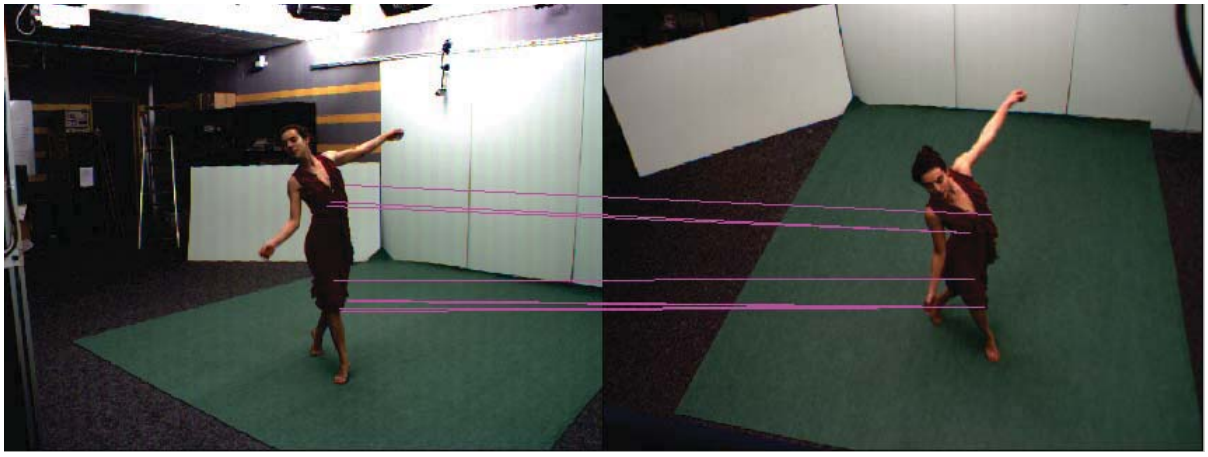

Abbildung 5.34 SIFT mit Levels = 10, magnif = 8 ,edge thresh = 100 und match ratio = 1.3: 9 Matches werden gefunden, wovon 7 korrekt sind

Wird der *magnification factor* auf 3 reduziert, werden 10 Matches gefunden, wovon aber 8 falsch sind.

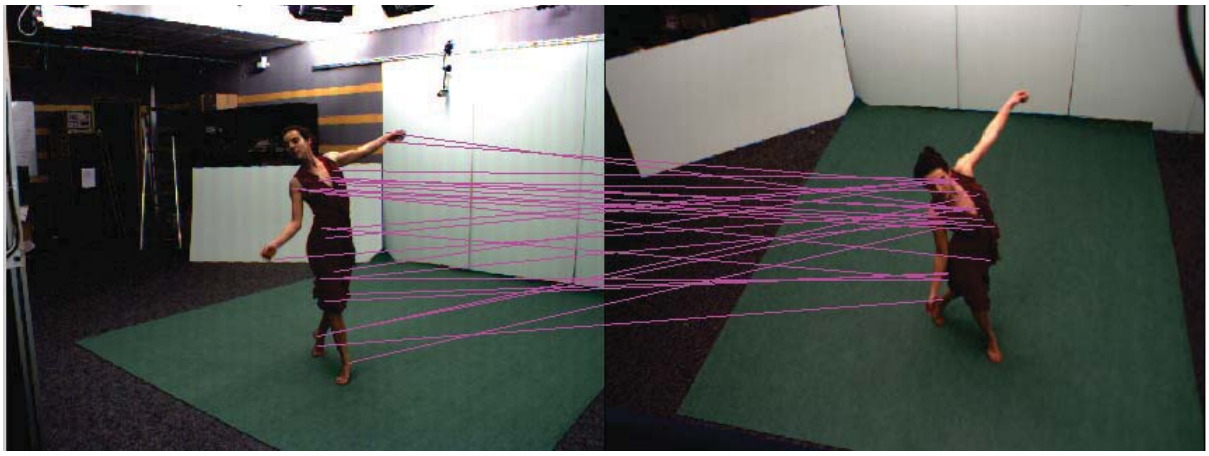

Abbildung 5.35 Surf mit doppelter Octaven Anzahl und doppelter Anzahl von Ebenen pro Octave,hessianThreshold = 5 und match ratio = 1.3: 24 Matches werden gefunden, wovon nur 2 korrekt sind

Nutzt man den *extended* Modus von SURF, also einen Deskriptor mit doppelter Größe (128 statt 64), werden die Ergebnisse leicht besser.

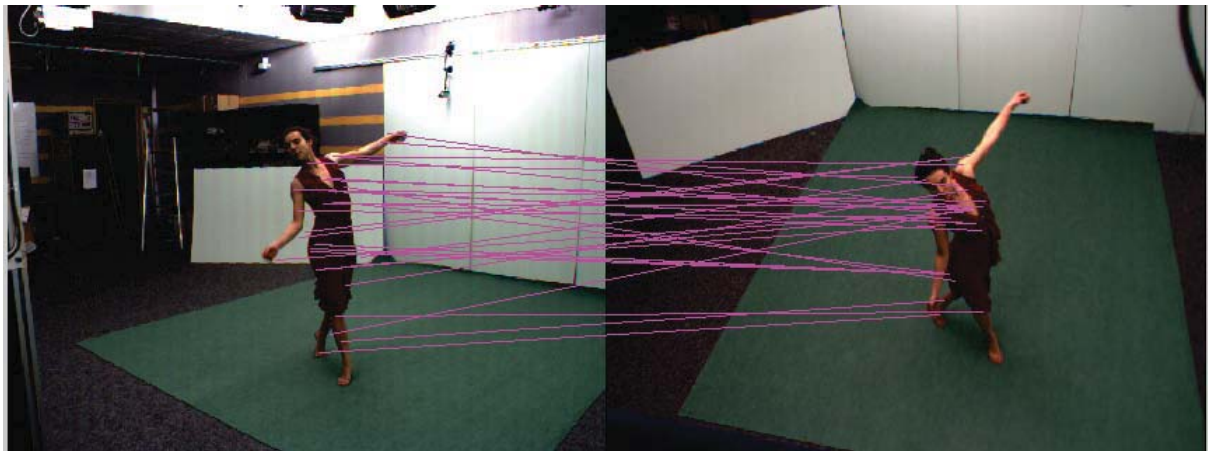

Abbildung 5.36 SURF mit doppelter Oktaven Anzahl und doppelter Anzahl von Ebenen pro Oktave und doppelter Deskriptor Größe, hessianThreshold = 50 und match ratio = 1.3: 31 Matches werden gefunden, wovon 7 korrekt sind

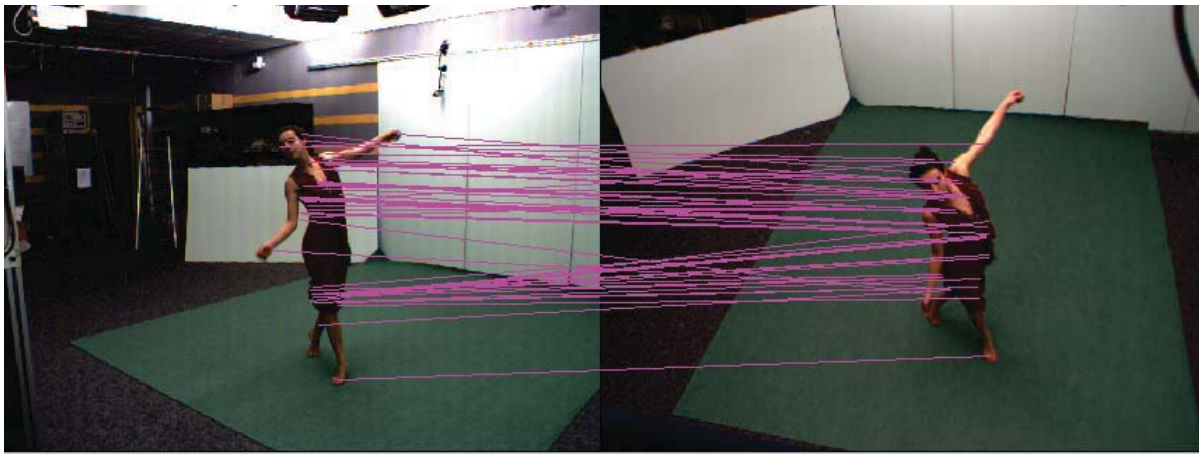

Abbildung 5.37 ASIFT mit magnif=3 ,OriHistThresh = 1 und scales=3: 53 Matches werden gefunden, wovon 22 der 50 besten korrekt sind

Kapitel 6

Auswertung

Werden aufeinanderfolgende Bildern einer Kamera betrachtet, liefern alle Verfahren gute Ergebnisse. SURF hat in diesem Fall am besten abgeschnitten. ASIFT findet zwar sehr viel mehr Matches, aber eine schlechtere Quote korrekter Matches. Was auffällt ist, dass je größer die Abstände zwischen 2 Bildern werden, desto weniger Matches gefunden werden und desto mehr falsche Matches auftreten. Bei Zeitabständen von 5 Frames liefern alle Verfahren noch sehr gute Ergebnisse. Am besten schneiden hier SURF , Harris-Affine und Hessian-Affine mit SIFT und GLOH Deskriptor ab, fast alle mit 100% korrekten Matches. ASIFT findet viele Matches mehr, aber eine schlechtere Quote korrekter Matches. Noch größere Zeitabstände von 10 Frames oder höher führen dazu, dass sinnvolles Matchen nicht mehr möglich ist, da die Bilder zu unterschiedlich sind. Werden gleichzeitige Bilder verschiedener Kameras betrachtet, liefern Hessian-Affine mit GLOH Deskriptor und SIFT die beste Quote korrekter Matches. ASIFT findet wie immer die meisten Matches, aber wieder auch viele falsche Matches und kommt auf eine Quote korrekter Matches von 51%. Hessian-Affine mit GLOH liegt bei 81% SIFT und Harris-Affine mit GLOH bei 75% , Hessian-Affine mit SIFT bei 62%, SURF bei 35% und Harris-Affine mit SIFT sogar bei 0%. Diese Ergebnisse wiederholen sich auch, wenn dazu noch die Zeitabstände erhöht werden (Szenario 4). Hessian-Affine mit GLOH liegt dann bei 91%, Hessian-Affine mit SIFT bei 90%, SIFT bei 78%, Harris-Affine mit GLOH bei 50%, ASIFT bei 44%, SURF bei 33% und Hessian-Affine mit SIFT wieder bei 0%.
Der Hessian-Affine Detektor ist besser als der Harris-Affine Detektor in den Szenarien 3 und 4, in den Szenarien 1 und 2 liefert Harris-Affine leicht bessere Ergebnisse. In allen 4 Szenarien liefert der GLOH Deskriptor bessere Ergebnisse als der SIFT Deskriptor. ASIFT findet in jedem Szenario die meisten Matches, aber auch mehr falsche Matches. SURF ist besser als SIFT, wenn die Kamera gleich bleibt und sich die Bilder nur in einzelnen Frames unterscheiden (Szenarien 1 und 2). Werden Bilder verschiedener Kameras betrachtet,

liefert SIFT bessere Ergebnisse als SURF (Szenarien 1 und 2).
Im folgenden wer den die Ergebnisse in Tabellenform dargestellt.

Verfahren	Szenario 1 - 1 Frame Abstand					
	Default Parameter			Optimierte Parameter		
	Anz.	Korrekt	Falsch	Anz.	Korrekt	Falsch
SIFT	45	45 / 100%	0 / 0%	143	50/50 / 100%	0/50 / 0%
SURF	54	54 / 100%	0 / 0%	93	50/50 / 100%	0/50 / 0%
ASIFT	983	50/50 / 100%	0/50 / 0%	1004	50/50 / 100%	0/50 / 0%

Verfahren	Szenario 1 - 3 Frames Abstand					
	Default Parameter			Optimierte Parameter		
	Anz.	Korrekt	Falsch	Anz.	Korrekt	Falsch
SIFT	29	29 / 100%	0 / 0%	119	50/50 / 100%	0/50 / 0%
SURF	34	34 / 100%	0 / 0%	69	50/50 / 100%	0/50 / 0%
ASIFT	594	50/50 / 100%	0/50 / 0%	678	50/50 / 100%	0/50 / 0%

Verfahren	Szenario 1 - 5 Frames Abstand					
	Default Parameter			Optimierte Parameter		
	Anz.	Korrekt	Falsch	Anz.	Korrekt	Falsch
SIFT	15	12 / 80%	3 / 20%	88	44/50 / 88%	6 / 12%
SURF	20	19 / 95%	1 / 5%	32	31 / 97%	1 / 3%
ASIFT	273	44/50 / 88%	6/50 / 12%	273	44/50 / 88%	6/50 / 12%
Harris-Aff-Sift	48	47 / 98%	1 / 2%	-	-	-
Harris-Aff-Gloh	24	24 / 100%	0 / 0%	-	-	-
Hessian-Aff-Sift	47	42 / 89%	5 / 11%	-	-	-
Hessian-Aff-Gloh	37	34 / 92%	3 / 8%	-	-	-

Verfahren	Szenario 2 - 10 Frames Abstand					
	Default Parameter			Optimierte Parameter		
	Anz.	Korrekt	Falsch	Anz.	Korrekt	Falsch
SIFT	15	12 / 80%	3 / 20%	88	44/50 / 88%	6 / 12%
SURF	20	19 / 95%	1 / 5%	32	31 / 97%	1 / 3%
ASIFT	76	20/50 / 40%	30/50 / 60%	277	23/50 / 46%	27/50 / 54%

Verfahren	Szenario 3					
	Default Parameter			Optimierte Parameter		
	Anz.	Korrekt	Falsch	Anz.	Korrekt	Falsch
SIFT	6	3 / 50%	3 / 50%	8	6 / 75%	2 / 25%
SURF	20	5 / 25%	15 / 75%	29	10 / 35%	19 / 65%
ASIFT	49	17 / 35%	32 / 65%	47	24 / 51%	23 / 49%
Harris-Aff-Sift	12	0 / 0%	12 / 100%	-	-	-
Harris-Aff-Gloh	12	9 / 75%	3 / 25%	-	-	-
Hessian-Aff-Sift	16	10 / 63%	6 / 37%	-	-	-
Hessian-Aff-Gloh	22	18 / 82%	4 / 18%	-	-	-

Verfahren	Szenario 4					
	Default Parameter			Optimierte Parameter		
	Anz.	Korrekt	Falsch	Anz.	Korrekt	Falsch
SIFT	8	5 / 63%	3 / 37%	9	7 / 78%	2 / 22%
SURF	20	2 / 10%	18 / 90%	31	7 / 23%	24 / 77%
ASIFT	55	14 / 28%	36 / 72%	53	23 / 43%	30 / 57%

6.1 Verteilung der Matches

Wenn mehr als 50 Matches gefunden wurden, wurden nur die 50 vermutlich besten Matches mit den kleinsten euklidischen Abständen berachtet. Dabei viel auf, dass diese 50 Matches nicht immer die besten waren. Es gab Fälle, in welchen die beiden Matches mit den kleinsten euklidischen Abständen, falsche Matches waren. Folgende Tabelle zeigt so einen Fall.

Match	Quadrierte Eukli. Distanz
1	8159
2	10788
3	11390
4	12812
5	15367
6	16982
7	19724
8	21312
9	22484
10	37748
11	42785
12	46571
13	51254
14	58486
15	77685

Die Tabelle zeigt die 15 Matches von SIFT zwischen den Bildern 1 und 5 der Kamera 0. Korrekte Matches sind in grün dargestellt, falsche Matches in rot. 12 der 15 Matches sind korrekt, die beiden vermutlich besten Matches sind aber ncht korrekt. In den meisten Fällen waren die besten Matches allerdings auch wirklich korrekt, Ausnahmen gab es jedoch auch.

6.2 Parametereinstellungen

Für die 3 Verfahren SIFT, SURF und ASIFT wird je ein Parameter beispielhaft variiert und die Auswirkungen auf die Matches untersucht. Für diese Versuche wurde in allen Fällen Bild 1 von Kamera 1 gegen Bild 5 der selben Kamera gematcht.

SIFT

Im folgenden werden die Auswirkungen der beiden Parameter *edgethres* und *peakthresh* auf die Matches betrachtet. In diesen Versuchen wurden die anderen Parameter nicht verändert.

Edgetresh	Anzahl Matches	Korrekte Matches
1	0	0 / 0%
5	8	6 / 75%
10	17	14 / 82%
20	25	17 / 68%
50	28	22 / 74%
100	27	21 / 78%
200	27	21 / 78%
300	27	21 / 78%
500	27	21 / 78%
750	27	21 / 78%
1000	27	21 / 78%

Ab einem edgethresh von 100, werden keine Feature Punkte auf Kanten mehr gefiltert, weshalb dieser Wert für alle Versuche mit SIFT verwendet wurde. Gibt man dem *peakthres* einen von 0 verschiedenen Wert, werden keine Matches gefunden. Der *peakthres* wurde daher in allen Versuchen auf 0 gesetzt.

SURF

Die folgende Tabelle zeigt, wie sich der Parameter *hessianThreshold* auf die Matches auswirkt. Wird ein großer Wert wie z.B. 500 gewählt, ist die Quote der korrekten Matches am besten. Wird dieser Wert klein gewählt, werden mehr Matches gefunden, wobei die Quote korrekter Matches nur minimal schlechter wird.

HessianThreshold	Anzahl Matches	Korrekte Matches
1	35	28 / 80%
5	36	30 / 83%
10	38	30 / 79%
20	36	30 / 83%
50	29	24 / 83%
100	31	24 / 77%
200	30	22 / 73%
300	21	18 / 86%
500	23	20 / 87%
750	23	19 / 83%
1000	19	16 / 84%

ASIFT

Je höher der Parameter *edgetresh* gesetzt wird, desto mehr Feature Punkte auf Kanten werden gefiltert. Auf Grund der großen Anzahl der gefundenden Matches, ist es nur schwer möglich, Aussagen über die Entwicklung der korrekten Matches zu machen. Es wurden daher immer nur die 50 besten Matches verglichen. Auf die 50 besten Matches hat der Parameter *edgetresh* so gut wie keine Auswirkungen. Die folgende Tabelle zeigt die Ergebnisse. Für die Versuche in diser Arbeit wurde der Wert 0.06 gewählt.

Edgetresh	Anzahl Matches
0.00	337
0.02	320
0.04	290
0.06	273
0.08	251
0.10	240
0.12	234
0.14	221
0.20	189

Literaturverzeichnis

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (surf).

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. *International Conference on Computer Vision & Pattern Recognition*.

INRIA (2007). Multiviewvideo datensatz dancer. *http://charibdis.inrialpes.fr/public/datasets*.

Ke, Y. and Sukthankar, R. (2003). Pca-sift: A more distinctive representation for local image descriptors.

Lowe, D. G. (2004). Distinctive image features from scale-invariant-keypoints. *International Journal of Computer Vision*.

Lowe, D. G. (2005). Demo software: Sift keypoint detector. *http : //www.cs.ubc.ca/ lowe/keypoints/*.

Mikolajczyk, K. and Schmid, C. (2004). Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27.

Morel, J.-M. and Yu, G. (2008). Asift: A new frmework for fully affine invariant image comparison. *SIAM*, 2.

Morel, J.-M. and Yu, G. (2011). Asift: An algorithm for fully affine invariant comparison. $http : //www.ipol.im/pub/algo/my_affine_sift/$.

Strandmark, P. (2010). Surfmex: A matlab surf interface. *http://www.maths.lth.se/matematiklth/personal/petter/surfmex.php*.

Vedaldi, A. and Fulkerson, B. (2011). Vlfeat 0.9.13. *http://www.vlfeat.org/*.

Zilly, F., Riechert, C., Eisert, P., and Kauff, P. (2011). Semantic kernels binarized: A feature descriptor for fast and robust matching.