

# Markerloses Posentracking für Augmented Reality Anwendungen

Matthias Schneider  
Universität Ulm  
matthias.schneider@uni-ulm.de

## Zusammenfassung

Diese Ausarbeitung behandelt ein Verfahren für markerloses Posentracking in Augmented Reality Anwendungen. Dabei werden statt klassischen *Markern* natürliche Objekte der realen Umgebung als Referenzpunkte genutzt um die Kameraposition zu bestimmen. Das Verfahren basiert nicht auf einem vorab bekannten 3D Modell der Umgebung, sondern kann direkt und ohne vorbereitende Phase eingesetzt werden. Die damit erkannten Merkmale sind größen- und rotationsinvariant, sowie teilweise invariant gegenüber Helligkeit- und Blickwinkeländerungen. Dadurch läßt es sich flexibel und dynamisch einsetzen. Ist die Kameraposition bekannt und somit das Weltkoordinatensystem festgelegt, können die Umwelt bereichernde virtuelle Objekte in die Szene eingefügt werden.

## 1 Einleitung

In der *Augmented Reality* werden virtuelle Objekte wie Bilder, Geräusche, Animationen etc. in die wirkliche Welt eingeblendet um dem Benutzer beispielsweise ein besseres Verständnis der Umgebung oder Informationen über diese zu geben. Die Projektion der Bilder wird hierbei in der Regel über ein *Head-Mounted Display* realisiert, das technisch auf verschiedene Arten aufgebaut sein kann. Eine Übersicht liefert [Azuma 2004]. Der Benutzer sieht mit dem Blick durch solch ein Gerät die virtuellen Objekte in das eigene Blickfeld eingeblendet. Es sind noch weitere Techniken zur Visualisierung der Objekte bekannt, auf die hier nicht weiter eingegangen werden soll.

Die Hauptaufgabe ist die korrekte Lokalisation der Blickrichtung des Benutzers bzw. des HMDs, denn nur so können die virtuellen Objekte perspektivisch korrekt in die Umgebung eingefügt werden. Die Blickrichtung bzw. die sich daraus ergebende Kameraposition kann mit 6 Freiheitsgrade dargestellt werden (3 Translations- und 3 Rotationsfreiheitsgrade). Die Bestimmung dieser Koordinaten ist eine komplexe Aufgabe. Es müssen schnelle und dynamische Bewegungen der Kamera berücksichtigt werden, der zugrunde liegende Algorithmus sollte möglichst robust und genau sein und die Berechenbarkeit sollte in einem für Echtzeitanwendungen vertretbaren Rahmen liegen. Weiterhin sollte eine Erkennung auch dann noch möglich sein, wenn Störungen wie zum Beispiel teilweise oder komplette Verdeckungen der Umgebungsobjekte auftreten.

## 2 Bisherige Forschung

In der Computer Vision wurde viel Forschung zu den Themen *Tracking* und Posenbestimmung betrieben. Daraus entstanden sind diverse Ansätze die interessant für die Umsetzung von Augmented Reality in Echtzeit sind.

Ein Ansatz versucht mit vorab aufgenommenen Bildsequenzen die Strukturen und Bewegungen der Szene zu bestimmen [Szeliski und Kang 1994]. Da wir aber an *Tracking* und Posenbestimmung in Echtzeit interessiert sind, ist dieser Ansatz mit seiner offline-Phase nicht empfehlenswert.

Weitere Verfahren, die auf optischem Fluss, *template matching* oder probabilistischen Methoden basieren, z.B. [Comaniciu et al. 2003], sind durchaus robust genug für Merkmalsverfolgung in Echtzeit,

sind aber nur für Situationen zu empfehlen in denen der Unterschied zwischen den einzelnen Frames nicht zu groß wird. Diese Unterschiede müssen gut vorhersehbar sein. Zum Beispiel abrupte Kopfbewegungen, wie sie in AR Anwendungen auftreten können, sind mit diesen Methoden nicht ausreichend gut behandelbar.

Eine Vielzahl der aktuellen Methoden verwendet den Ansatz, *Tracking* durch Merkmalerkennung möglich zu machen. Dabei werden markante Referenzpunkte, so genannte *Marker*, in der Umgebung platziert, die dem Zweck der einfachen Erkennung und Registrierung dienen. Für das Design dieser Marker sind viele Ansätze vorgeschlagen worden, vor allem kreisförmige und planare Marker finden in der Praxis häufig Verwendung [Claus and Fitzgibbon 2004].

Ein Problem von marker-basierten Verfahren ist, dass selbst kleinste Verdeckungen der Marker das *Tracking* unterbrechen. Dies ist häufig in sehr dynamischen und natürlichen Umgebungen der Fall. Dem kann durch die Platzierung von mehreren Marker entgegen gewirkt werden, was aber auch in Hinsicht auf die einfache und direkte Nutzung von Augmented Reality eine nicht optimale Lösung darstellt.

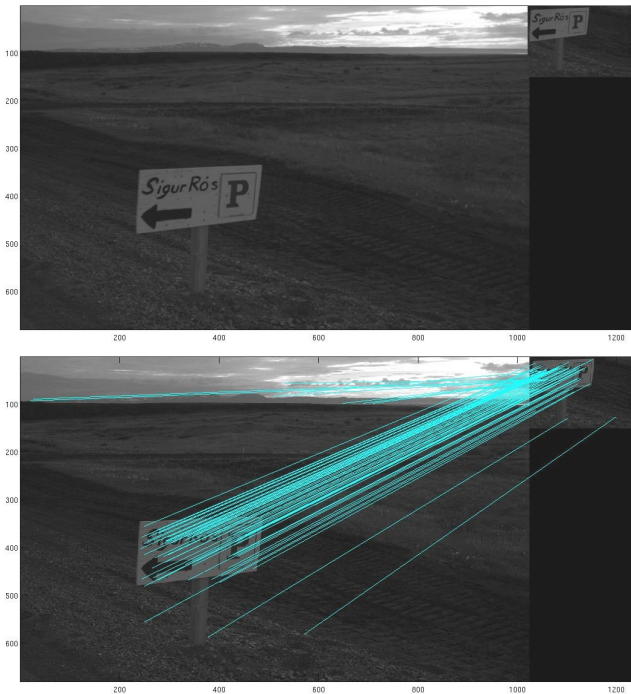
Eine weitere Methode, die vor allem in der letzten Zeit viel versprechende Resultate lieferte, ist die Verwendung von lokalen Merkmalen in der Umgebung für die Posenbestimmung der Kamera. Dabei hat sich laut [Mikolajczyk and Schmid 2003] das Verfahren SIFT (*scale invariant feature transform*) als einer der besseren Merkmalsdetektoren herausgestellt.

### 2.1 SIFT

Im Folgenden soll kurz auf das SIFT-Verfahren [Lowe 2004] eingegangen werden, da es die Grundlage für den noch vorzustellenden markerlosen Trackingalgorithmus liefert. SIFT ist ein Algorithmus der aus diskreten Pixelbildern Merkmale extrahiert. Diese Merkmale sind invariant gegenüber Größenänderung sowie Rotation und teilweise invariant bei Änderungen in der Beleuchtung und dem Standpunkt der Aufnahme. Die daraus resultierenden Merkmale (*keypoints*) sind sehr markant, was dazu führt, dass mit einer hohen Wahrscheinlichkeit die Merkmale in weiteren Bildern wiedererkannt werden, siehe auch Abbildung 1.

In einem ersten Schritt werden verschiedene Skalierungen des Bildes mit einem Gaußfilter geglättet und mit je zwei benachbarten Bildern das *Difference of Gaussian*-Verfahren (DoG) durchgeführt. Die weiteren Schritte sind:

- *Scale-space extrema detection*: Die DoG-Bilder werden nach interessanten Merkmalen, die skalierungs- und orientierungs-invariant sind durchsucht.
- *Keypoint localization*: Aus allen Merkmalskandidaten werden die Besten ausgewählt (Extrema). Als Maß gilt die Stabilität des Merkmals.
- *Orientation assignment*: Für jeden Keypoint wird dessen Ausrichtung in einer 16x16 Nachbarschaft bestimmt, basierend auf dem lokalen Bildgradienten.
- *Keypoint descriptor*: Ein 128-dimensionaler Vektor wird als Beschreibung der Keypoints erstellt. Dieser berechnet sich



**Abbildung 1:** SIFT Beispiel. Szene, in der Merkmale im linken Bild im rechten Bild, das ein verkleinerten Ausschnitt des linken Bildes darstellt, wiedergefunden wurden.

aus dem lokalen Gradienten und der ausgewählten Skalierung.

### 3 Markerloses Posentracking

Im Folgenden soll nun ein Verfahren zur markerlosen Posenschätzung von Chunrong Yuan vorgestellt werden [Yuan 2006]. Markerlose Verfahren haben in der besseren Benutzbarkeit und dem spontaneren Einsatz ohne vorherige Preparierung der Umgebung Vorteile gegenüber markerbasierten Verfahren.

#### 3.1 Ansatz

Das vorgestellte Verfahren schlägt diverse Verbesserungen gegenüber älteren Verfahren vor. So wird immer nur der aktuelle Frame zur Bestimmung der Kameraposition benötigt. Damit tritt das stetige "abdriften" der Bezugspunkte im Bild nicht auf, weil dort jeweils auf einen vorherigen Frame bezug genommen wird. Auch muss keine a priori Schätzung der wahrscheinlichen Kamerabewegung angenommen werden, so dass sich die sechs Freiheitsgrade der Kameraposition vollständig aus einem Frame berechnen lassen. Weiterhin muss keine 3D-Beschreibung der Umgebung vorliegen, wie das in anderen Verfahren, zum Beispiel durch ein CAD-Modell, der Fall ist. Der Algorithmus arbeitet komplett auf einer zweidimensionalen Beschreibung der Umgebung, was die Berechnungskomplexität verringert.

Zur Beschreibung der markanten Merkmalen wird SIFT verwendet, jedoch wurde zur Beschreibung der Features (*Keypoint descriptor*) ein leicht verändertes, kompakteres Verfahren entwickelt.

Um ein Koordinatensystem in der echten Umgebung zu modellieren, über das die Kameraposition berechnet wird und in späteren Schritten die virtuellen Objekte gerendert werden, wird ein virtu-

elles Quadrat in die reale Umgebung gesetzt. Die Eckpunkte des Quadrats werden dabei über per SIFT erkannte Features bestimmt.

Während einer vorbereitenden Phase werden Referenzframes aufgenommen (typischerweise nicht mehr als zwei bis drei Frames), in denen die vier Eckpunkte des Quadrates bestimmt werden. Während der Live-Phase werden diese Punkte bzw. die Merkmale dahinter den entsprechenden Merkmalen im aktuellen Bild zugeordnet (*matching*) und damit für jeden Frame das virtuelle Quadrat bestimmt.

### 3.2 Merkmalerkennung und Beschreibung

Um nun Merkmale zu finden, werden mehrere Arbeitsschritte sequentiell ausgeführt. Zuerst werden, mit Hilfe des *Difference of Gaussian*-Verfahren (DoG) die auffälligen Strukturen im Bild bestimmt. DoG ist eine *Mexican Hat* Funktion, im mehrdimensionalen auch Laplace-Operator genannt, durch Differenz einer schmalen mit einer breiteren Gaußnormalfunktion (siehe (1)). Damit können beispielsweise auffällige Kanten bzw. Kontraste im Bild ermittelt werden. Diese Operation wird nun auf jedem Frame mehrfach mit verschiedenen Skalen  $\sigma$  durchgeführt, um eine Menge  $\{D(x, y, \sigma)\}$  von DoG Bildern zu erhalten.

Nun werden in jedem Frame dieser Menge die Minima und Maxima eben dieses und der jeweils benachbarten Frames gesucht und als markante Merkmale interpretiert. Durch weiteren Methoden (Berechnung der Hauptkrümmung von  $D(x, y, \sigma)$  über die Eigenwerte der zugehörigen Hesse-Matrix) werden instabile Extrema mit niedrigem Kontrast heraus gefiltert, sowie die Löschung aller Punkte die auf einer Kante liegen veranlasst, damit man nur auf stabilen Merkmalen weiterarbeitet.

Sind nun alle Features eindeutig bestimmt, muss eine Beschreibung des Abstandes der entsprechenden Features zueinander bestimmt werden. Dabei muss auch jeweils die Skalierung  $\sigma$  notiert werden, da wir mit skalierungsinvarianten Merkmalspunkten arbeiten wollen und somit die Breite des Gaußkerns in jedem weiteren Berechnungsschritt bekannt sein muss.

Die Berechnung der Orientierung der Merkmalspunkte basiert auf den Änderungen der Intensität in einem durch den Laplace-Operator generierten Bildes. Angenommen  $F(x, y)$  ist das Eingabebild das in den obigen Schritten erzeugt wurde. Dieses Bild wird nun mit der Gauß'schen Dichtefunktion

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{(x^2+y^2)}{(2\sigma^2)}} \quad (1)$$

gefaltet und ergibt das angesprochene Laplacebild

$$L(x, y, \sigma) = G(x, y, \sigma) * F(x, y) \quad (2)$$

Aus diesem Bild wird nun ein Betragsbild  $M(x, y, \sigma)$  und ein Orientierungsbild  $\Theta(x, y, \sigma)$  berechnet:

$$M(x, y, \sigma) = |L(x + 1, y, \sigma) - L(x - 1, y, \sigma)| + |L(x, y + 1, \sigma) - L(x, y - 1, \sigma)| \quad (3)$$

$$\Theta(x, y, \sigma) = \arctan \frac{L(x, y + 1, \sigma) - L(x, y - 1, \sigma)}{L(x + 1, y, \sigma) - L(x - 1, y, \sigma)} \quad (4)$$

Nun wird für jeden erkannten Merkmalspunkt in  $L(x, y, \sigma)$  ein Histogramm mit 16 Bins aus den umliegenden Pixeln erstellt. Der Modalwert dieses Histogramms gibt die dominante Richtung des Merkmals an.

Der Modalwert, die Position  $(x, y)$  und die Skalierung  $\sigma$  ergeben nun zusammen mit den Bildern  $M(x, y, \sigma)$  und  $\Theta(x, y, \sigma)$  eine robuste, rotationsinvariante Beschreibung der Merkmalspunkte. Dabei wird um jeden Punkt eine  $8 \times 8$  Region mit je vier  $4 \times 4$  Teilregionen definiert. In jeder Teilregion wird nun wieder ein Histogramm mit 8 Bins aufgebaut, das sich als achtdimensionaler Vektor beschreiben lässt. Damit ergibt sich zusammen für jeden Merkmalspunkt ein  $4 * 8 = 32$ -dimensionaler Vektor der diesen Punkt beschreibt.

### 3.3 Zuordnung der Merkmale und Posenerkennung

Nun müssen die erkannten Merkmale in den Referenzframes  $f_i$  den korrespondierenden Merkmalen im aktuellen Frame zugeordnet werden (*matching*). Das Maß dafür ist der Euklidische Abstand der Punkte zueinander:

$$euklid(a, b) = \|a - b\|_2 = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2} \quad (5)$$

mit  $a = (x_1, y_1)$  und  $b = (x_2, y_2)$  Pixel bzw. Punkte im Bild.

Dabei wird als Bewertung des Abstands ein globaler Schwellwert (*threshold*) sowie das Verhältnis zwischen dem kleinsten und dem zweit kleinsten Abstand angenommen.

Weiter muss ein Maß für das Zuordnen von Frames zueinander gefunden werden. Hier wird für jeden Referenzframe zum Einen die komplette Anzahl  $n_i$  aller zugeordneten Punkte zwischen ihm und dem aktuellen Frame berechnet, zum Anderen wird der durchschnittliche Abstand  $d_i$  zwischen allen zugeordneten Punkten dieser beiden Frames gebildet.

Nun kann der am besten geeignete Referenzframe bestimmt werden. Dies ist der Frame, der die höchste Anzahl an zugeordneten Punkten zum aktuellen Frame und den kleinsten durchschnittlichen Abstand zwischen diesen Punkten aufweisen kann. Wenn das Maximum bzw. Minimum beider Werte über alle Frames in genau einem Frame gefunden wird, handelt es sich um den gleichen Referenzframe. Formell ausgedrückt:

$$f_i^1 = \max\{n_i\}, f_i^2 = \min\{d_i\} \Rightarrow f_i = f_i^1 = f_i^2$$

Falls  $f_i^1 \neq f_i^2$  gilt, wird  $f_i^1$  als Referenzframe eingesetzt, aber  $f_i^2$  wird weiterhin behalten, da  $f_i^1$  durch Ausreißer nicht immer die optimale Lösung darstellt und im Spezialfall durch  $f_i^2$  ersetzt werden kann.

Im nächsten Schritt wird eine weitere Optimierung vorgenommen. Aus dem aktuellen Frame und dem Referenzframe werden 3 Merkmalspaare per Zufall bestimmt und darauf eine affine Transformation angewandt. Damit lässt sich die Bewegung, die zwischen dem Referenzframe und dem aktuellen Frame gemacht wurde, bestimmen. Die sechs Unbekannten dieser Transformation können mit den drei Datenpaaren gelöst werden. Yuan verwendet hier RANSAC [Fischler and Bolles 1981], um die initiale Transformationsmatrix zu bestimmen. Nun wird mit der Methode der kleinsten Quadrate (*linear least squares*) versucht, die beste affine Transformation für die Merkmale zu finden, d.h. die Transformationsmatrix wird Schritt für Schritt angepasst und dabei optimiert. Damit können Ausreißer entdeckt und entfernt werden, falls diese nicht in die affine Transformation passen.

Ist nun der passende Referenzframe und die zugehörige affine Transformation gefunden, kann das virtuelle Quadrat im Raum berechnet werden. Dieses virtuelle Quadrat bestimmt das Weltkoordinatensystem für in die Szene einzufügende Objekte.

Die 2D Eckpunkte des virtuellen Quadrats können nun bestimmt werden indem die Transformationsmatrix  $A_k$  des k-ten Referenzframes mit dem j-ten Eckpunkt  $x_j^k$  mit  $(j = 1, \dots, 4)$  multipliziert wird. Nun muss anhand von diesem Informationen nur noch die 3D Kameraposition bestimmt werden.

Yuan bedient sich hier in einem ersten Schritt beim POSIT (*pose with iteration*) Algorithmus [Oberkamp et al. 1996] um die initiale Kameraposition zu bestimmen. Alle weiteren Kamerapositionen werden nun anhand des Algorithmus in [Schweighofer and Pinz 2006] verfeinert, da eine erste Schätzung durch POSIT nicht sehr genau sein kann.

### 3.4 Evaluation des Verfahrens



**Abbildung 2:** Einfaches Augmented Reality Beispiel mit Tracking. Die Wähltastatur wird vom virtuellen Quadrat eingerahmt, auf das zur Demonstration ein virtueller Kubus gesetzt wurde. Quelle: [Yuan 2006]

Yuan hat zur Evaluation des von ihm vorgeschlagenen Verfahrens eine Reihe von Tests in einer typischen Büroumgebung durchgeführt. In Abbildung 2 ist ein Telefon mit einem gelb eingezeichneten virtuellen Quadrat zu sehen. Dieses spannt das Weltkoordinatensystem auf. Das Quadrat kann dabei nicht exakt der Oberfläche des Telefons angepasst werden, da diese eine leichte Krümmung aufweist. Es lassen sich nun die vier Eckpunkte des Quadrates bestimmen.

Zur Demonstration der Genauigkeit des Verfahrens wird ein Kubus auf das virtuelle Quadrat gezeichnet, welches gleichzeitig die Grundfläche des Kubuses darstellt. Es können hier natürlich beliebige 3D-Modelle in die Umgebung gerendert werden, da die Kameraposition bekannt ist und somit alle Informationen für den Renderer vorliegen.

Wie man auf den Bildern in Abbildung 2 sieht, ist der Algorithmus auch bei starken Blickwinkeländerungen sowie Verdeckungen weiterhin in der Lage, den Kubus zu zeichnen. Dies ist ein Vorteil gegenüber Marker-basierten Verfahren.

Yuan gibt an, dass das Verhältnis zwischen aufgenommenen und erkannten Frames je nach Szene zwischen 80% und 95% liegt. Als erkannte Frames werden nur die gezählt, in denen sowohl die Merkmale erfolgreich erkannt sowie auch die 3D-Kameraposition erfolgreich berechnet wurde. Da es sich bei dem eingesetzten SIFT Verfahren um ein Verfahren handelt, das auf verschiedenen Skalierungen des Bildes nach Merkmalen sucht, leidet durch die daraus resultierende mehrfache Berechnung die Performanz des Algorithmus. Dies konnte aber durch den kompakteren *keypoint descriptor* verbessert werden. Auf einem Pentium M Notebook mit 1.73 Ghz und einer Logitech Webcam lies sich der Algorithmus mit 5 Bildern pro Sekunde betreiben.

### 3.5 Zusammenfassung

Yuan zeigt, dass mit seinem Verfahren Posentracking mit einer annehmbaren Performanz möglich ist, auch wenn die Umgebung nicht mit Markern ausgestattet wurde. Die Erkennung der Merkmale in den Referenzbildern sowie im aktuellen Kamerabild wird über SIFT realisiert, wobei eine eigene, kompaktere Beschreibung der Merkmale eingesetzt wird. Anhand dieser erkannten Merkmale wird ein virtuelles Quadrat in die Szene gesetzt, über welches die Kameraposition berechnet sowie als Weltkoordinatensystem für die Anzeige der virtuellen Objekte der *Augmented Reality* verwendet wird.

## 4 Weitere Verfahren

In [Pang et al. 2006] wird ein weiteres, ähnliches Verfahren vorgeschlagen. Es setzt statt SIFT den Kanada-Lucas-Tomasi (KLT) Algorithmus ein um markante Merkmale in der Umgebung zu finden. Wie bei Yuan werden Referenzbilder vorab aufgenommen, um je vier Punkte zu bestimmen, die sich zu einem Weltkoordinatensystem zusammenfügen lassen. Über affine Transformationen wird auch hier die eigentliche Kameraposition bestimmt.

Ein Nachteil dieses Verfahrens ist jedoch, dass die vier Punkte manuell von Hand in einer Initialisierungsphase bestimmt werden müssen. Ausgenommen, der Algorithmus findet eine flache, quadratische Fläche im Bild um die Punkte automatisch zu bestimmen. Dies kommt jedoch einem Marker-basierten Verfahren ziemlich nahe.

Das Verfahren zeigt bei ähnlicher Hardware eine bessere Performanz (ca. 20 fps bei einer Auflösung von 640x480 Pixeln), da es keine mehrfachen Berechnungen wie, bei Yuan verursacht durch die verschiedenen Skalierungen in SIFT, durchführt. Dadurch ist der vorgeschlagene Algorithmus aber anfälliger gegenüber Größenänderungen, was sich in einer niedrigeren Rate an korrekt erkannten Frames äußert.

## Literatur

AZUMA, R. 2004. Overview of augmented reality. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*, ACM, New York, NY, USA, 26.

CLAUS, D., AND FITZGIBBON, A. W. 2004. Reliable fiducial detection in natural scenes. In *European Conference on Computer Vision*, Vol IV: 469–480.

COMANICIU, D., RAMESH, V., AND MEER, P. 2003. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* 25, 5, 564–575.

FISCHLER, M. A., AND BOLLES, R. C. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6, 381–395.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60, 2, 91–110.

MIKOLAJCZYK, K., AND SCHMID, C. 2003. A performance evaluation of local descriptors. In *CVPR*, IEEE Computer Society, 257–263.

OBERKAMPF, D., DEMENTHON, D., AND DAVIS, L. S. 1996. Iterative pose estimation using coplanar feature points. *Computer Vision and Image Understanding* 63, 3, 495–511.

PANG, Y., YUAN, M. L., NEE, A. Y. C., ONG, S.-K., AND YUCEF-TOUMI, K. 2006. A markerless registration method for augmented reality based on affine properties. In *AUIC*, Australian Computer Society, W. Piekarski, Ed., vol. 50 of *CRPIT*, 25–32.

SCHWEIGHOFER, G., AND PINZ, A. 2006. Robust pose estimation from a planar target. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 12, 2024–2030.

SZELISKI, R., AND KANG, S. B. 1994. Recovering 3D shape and motion from image streams using non-linear least squares. *Journal of Visual Communication and Image Representation* 5, 1, 10–28.

YUAN, C. 2006. Markerless pose tracking for augmented reality. In *Advances in Visual Computing*, I: 721–730.