Projekt 1: Marker Pose Estimation - Evaluation der Unterschiede zwischen cv::solvePnP und RANSAC

Der Ren David Chung, Marcel Vrdoljak, Halil Ibrahim Pamuk

***Abstract*—Die präzise Bestimmung der Pose spielt eine zentrale Rolle in Kamerasystemen für Roboter. Sie ermöglicht die zuverlässige Erkennung von Objekten und die damit verbundene Festlegung der Greifpunkte für eine Robotermanipulation. Für die Pose Estimation können verschiedene Algorithmen angewandt werden, jede mit ihren Vor- und Nachteilen. Dieses Paper beschäftigt sich mit dem Vergleich von zwei Methoden zur Pose Estimation von Markern, die auf einem Schachbrett platziert sind. Die eine Methode ist SolvePnP, die die Korrespondenz zwischen 2D- und 3D-Punkten nutzt, die andere ist RANSAC, die ein Schätzverfahren auf Basis von zufälligen Stichproben verwendet. Die Ergebnisse zeigen, dass die selbst implementierte RANSAC-Methode der SolvePnP-Methode nahe kommt, aber einige Vorzeichenfehler aufweist. Die Leistung des RANSAC-Algorithmus kann durch die Anpassung der Anzahl der Iterationen und des Reprojektionsfehlers verbessert werden.**

I. Einleitung

Die genaue Bestimmung der Position sowie der Rotation eines Objektes, auch bekannt als Pose Estimation, spielt eine entscheidende Rolle in verschiedenen Gebieten wie Computer Vision, Virtual Reality und Robotik. Die genaue Lokalisierung von Objekten ermöglicht es komplexe Aufgaben wie die Navigation von Robotern oder die Einbettung von virtuellen Objekten zu lösen. In den letzten Jahren haben sich mehrere unterschiedliche Ansätze und Algorithmen zur Lösung der Pose Estimation etabliert. Die Wahl eines geeigneten Algorithmus basiert auf mehreren Faktoren, wie die Genauigkeit, Robustheit gegenüber Ausreißern und die Geschwindigkeit der Berechnungen. Zwei Ansätze zur Lösung der Pose Estimation welche weit verbreitet sind, sind SolvePnP (Solve Perspective-n-Point) und RANSAC (Random Sample Consensus). SolvePnP nutzt die Korrespondenz zwischen 2D und 3D Punkten zur Berechnung der Pose. RANSAC verwendet hingegen ein Schätzverfahren, welches auf zufälligen Stichproben basiert und eine Handhabung von Ausreißern ermöglicht.

II. Stand der Technik

Die SolvePnP Methode ist bereits weit verbreitet und bewährt. Ein Beispiel ist die Verwendung von SolvePnP zur Erkennung der Kopfposition und Augenposition von Autofahrern um Unfälle und Sekundenschlaf zu vermeiden [9]. Eine weitere Anwendung ist die Pose Estimation einer Kamera auf einem unbemannten Luftfahrzeugen (UAV) [2]. Auch in Verbindung mit Augmented Reality findet SolvePnP durch seine Genauigkeit und Effizienz Anwendung [4]. Die solvePnP Funktion schätzt die Pose eines Objekts basierend auf einer Reihe von Objektpunten und ihrer Korrespondenzen im Bild. Dabei werden auch auf die intrinsischen Parameter, wie Kameramatrix und Verzerrungskoeffizient, Rücksicht genommen [6]. Die cv::solvePnP Funktion stellt eine Vielzahl an Methoden zur Verfügung wie Efficient Perspective-n-Point [3] oder auch P3P Methode basierend auf [1]. Die Methode kann über einfache cv::Flags ausgewählt werden. Standardmäßig benutzt cv::solvePnP die iterative Methode basierend auf Levenberg-Marquardt. RANSAC hingegen findet Anwendung in Situation, in denen Ausreißer in Daten auftreten bzw. Daten fehlerhaft oder gestört sind. So wird dieser Algorithmus in stark rauschenden Umgebungen sehr oft angewandt [7]. Sowohl SolvePnP als auch RANSAC haben Vor- und Nachteile. Während SolvePnP präsizer und empfindlicher gegenüber Ausreißern ist, ist RANSAC robuster und ungenauer.

III. Aufgabenstellung

Im Rahmen dieser Arbeit soll ein Programm zur Pose Estimation entwickelt werden. Es soll in der Lage sein einen erstellten Marker zu detektieren und die Pose zu bestimmen. Es soll dabei ein Vergleich zwischen dem solvePnP Algorithmus und einer eigen implementierten Methode (RANSAC) untersucht werden. Abbildung 1 zeigt eine mögliche Implementierung des Algorithmus.

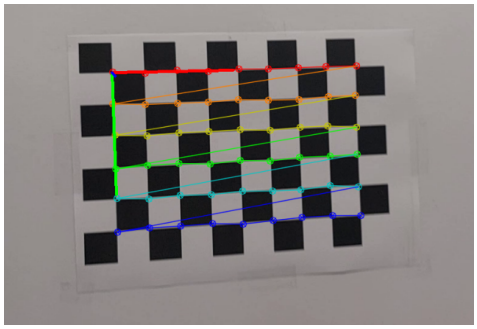

Fig. 1: Mögliche Implementierung eines Algorithmus [10]

IV. Methodik

In dieser Arbeit wurde die eigene Methode in Form eines RANSAC Algorithmus implementiert. Der mathematische Hintergrund wird in Kapitel IV-B genauer erläutert. Für den qualitativen Vergleich beider Methoden wurde ein 7x10 Felder großes Schachbrett verwendet. Hierbei dienen die Eckpunkte als 2D Bildpunkte für beide Herangehensweisen. Die Merkmale werden mit Hilfe der OpenCV Funktion cv::findChessboardCorner vom entzerrten Bild ermittelt [5]. Es wurde ein Video mit einer Auflösung von 1280x720 aufgenommen.

A. Methode mit solvePnP

Hierfür wurde die cv::solvePnP Funktion verwendet. Diese ermittelt aus den korrespondierenden 2D-Bildpunkte und 3D-Objektpunkte die dazugehörige Translation und Rotation. Falls alle notwendigen Eckpunkte detektiert wurden, wird das Resultat in Form eines eingezeichneten Koordinatensystems in jedem Frame des Videos dargestellt. Da keine cv Flag ausgewählt ist, wurde hier die Standard Methode basierend auf die Levenberg-Marquardt Optimierung ausgewählt.

B. Methode mit RANSAC

Die Implementierung des RANSAC Algorithmus erfolgt ähnlich zur bereits beschriebener Implementierung in IV-A. Jedoch im Gegensatz zu solvePnP wird die Pose zufällig durch eine Untermenge von vier Bildpunkten und ihren entsprechenden Objektpunkten ermittelt. Dieser Vorgang wird für eine bestimmte Anzahl an Iterationen wiederholt um die beste Pose zu finden. Dabei wird in jeder Iteration wieder zufällig vier Korrespondenzen ausgewählt. Die vorgestellte RANSAC Implementierung verwendet die Methode der kleinsten Quadrate, um eine Transformation zwischen 2D-Bildpunkt und 3D-Objektpunkte zu berechnen. Wir können aus dem Zusammenhang in Formel 1, wobei x die 2D-Bildpunkte, X die 3D-Objektpunkte und P (3x3) die Projektionsmatrix ist, eine lineare Gleichungssystem in Form von Formel 2 aufstellen und diese über die Singulärwertzerlegung eine Lösung ermitteln [8]. Das Symbol a_i entspricht einer einzigen Korrespondenz. Die Variable p ist hier die Ansammlung der Parameter von P in 9x1 Form (siehe Formel 3).

$$x = P \cdot X \tag{1}$$

$$a_i \cdot p = 0 => A \cdot p = 0 \tag{2}$$

$$P = \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a33 & a32 & a33 \end{bmatrix} = \begin{bmatrix} A \\ B \\ C \end{bmatrix} => p = \begin{bmatrix} A^T \\ B^T \\ C^T \end{bmatrix} \tag{3}$$

In unseren aufgestellten Gleichungssystem haben wir redundante Beobachtung, somit ist das System überbestimmt und wir haben mehr Gleichungen als Unbekannte. Es gibt daher keine exakte Lösung und suchen daher nach einer optimalen Lösung die den Fehler zwischen den Beobachtung und geschätzten Parameter minimiert. In unseren Fall wurde die letzte Spalte der V Matrix (siehe Formel 4) ausgewählt, da dieser mit dem kleinsten Singulärwert korresponidert [8].

$$M = U \cdot \Sigma \cdot V^T \tag{4}$$

Es ist zu beachten, dass der Algorithmus keine Lösung hat, wenn alle Z Koordinaten im Objekt auf einer Ebene liegen (Z = 0). Aus diesem Grund wurde dieser Implementierung ein Z Wert von 1 genommen. Nach der Schätzung der Pose werden die verbleibenden Punkte betrachtet. Die Bildpunkte werden unter Anwendung der geschätzten Pose auf die 3D-Objektpunkte projiziert. Der Abstand zwischen den projizierten Punkten und den tatsächlichen Bildpunkten wird berechnet. Durch einen Schwellenwert (reprojection-Error) wird der Status (Inlier oder Outlier) bestimmt. Punkte, deren Abstand kleiner als der Schwellenwert ist, werden hier als Inlier bezeichnet. Die geschätzte Pose mit der größten Anzahl an Inlier wird als ”beste” Pose ausgewählt und zur Ausgabe weiterverarbeitet.

V. Ergebnisse und Diskussion

Nun erfolgt der Vergleich zwischen der cv::solvePnP und der eigenen Methode in Form eines RANSAC Algorithmus. Das Ergebnis der cv::solvePnP kann Abbildung 2 entnommen werden. Hier wird die zeitliche Änderung von der Translation (tvec_x,tvec_y und tvec_z) und Rotation (rvec_x,rvec_y und rvec_z) dargestellt. Insgesamt stellt der Graph die Änderungen im aufgenommenen Video gut dar.

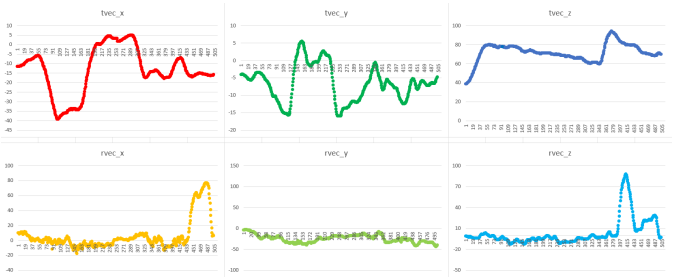

Fig. 2: Output cv::solvePnP

Um einen Vergleich zum obigen fertigen Algorithmus herzustellen, wurden zwei Versuche mit eigener RANSAC Implementierung durchgeführt. Die selbstgeschriebene Methode wurde mit unterschiedlicher Parametrisierung durchgeführt. Die Anzahl der Iterationen und der Schwellenwert für den Reprojektionsfehler sind zwei wichtige Parameter, die die Leistung des RANSAC-Algorithmus für die Schätzung der Kameraposition beeinflussen können. Die Anzahl der Iterationen bestimmt, wie oft der RANSAC-Algorithmus eine Teilmenge von Punkten zufällig auswählt und die Kameraposition anhand dieser Punkte schätzt. Eine Erhöhung der Anzahl der Iterationen kann die Chancen auf eine gute Schätzung der Kameraposition erhöhen, vor allem, wenn es viele Ausreißer in den Daten gibt. Die Erhöhung der Anzahl der Iterationen erhöht jedoch auch die Rechenkosten des Algorithmus. Der Schwellenwert für den Reprojektionsfehler legt fest, wie nahe ein projizierter 3D-Punkt an seinem entsprechenden 2D-Bildpunkt liegen muss, um als Ausreißer zu gelten. Liegt der Reprojektionsfehler unter diesem Schwellenwert, gilt der Punkt als Ausreißer und wird zur Schätzung der Kameraposition verwendet. Liegt der Reprojektionsfehler über diesem Schwellenwert, gilt der Punkt als Ausreißer und wird nicht zur Schätzung der Kameraposition verwendet. Die Einstellung eines niedrigen Schwellenwerts für den Reprojektionsfehler kann den Algorithmus robuster gegenüber Rauschen und Fehlern in den Daten machen, kann aber auch dazu führen, dass weniger Ausreißer gefunden werden, was eine genaue Schätzung der Kameraposition erschweren kann.

Erstens, wurde der Funktion 100 Iterationen und 8 Reprojektionsfehler mitgegeben. Das Ergebnis dieses Durchgangs ist in Abbildung 3 ersichtlich.

Fig. 3: Output Graph RANSAC (100,8)

Der RANSAC-Algorithmus ist ein randomisierter Algorithmus, der in jeder Iteration eine zufällige Teilmenge von Punkten auswählt, um die Kameraposition zu schätzen. Dies kann zu Schwankungen in der geschätzten Pose von einer Iteration zur nächsten führen. Infolgedessen kam es sehr häufig zu vertauschten Vorzeichen und dadurch ist der Anschein eines gespiegelten Graphen aufgekommen.

Wenn man die Vorzeichenfehler, die auf die Natur vom RANSAC-Algorithmus zurückzuführen sind, korrigiert, kommt die selbst erstellte Methode der cv::solvePnP Methode nah. Dies sieht man auch in Abbildung 4. Viele Abweichungen sind sehr gering und fast nicht zu erkennen.

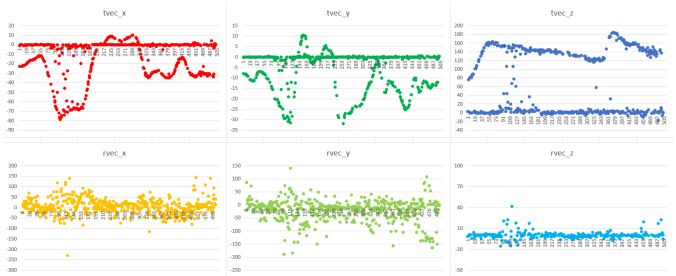

Fig. 4: Output Graph Error RANSAC (100,8)

Zweitens, wurde die Anzahl an Iterationen auf 1000 erhöht und die Menge an Reprojektionsfehlern auf 2 heruntergesetzt. Das Ergebnis ist in Abbildung 5 visualisiert. Es ist weniger ”noise” und ein deutlicherer Verlauf für die Translationen und Rotationen erkennbar. Die Vorzeichenfehler sind nach wie vor persistent.

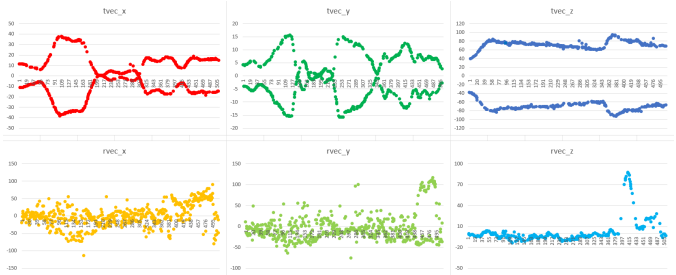

Fig. 5: Output Graph RANSAC (1000,2)

Der ”Fehler” zur cv::solvePnP wird in Abbildung 6 gezeigt. Auch hier sieht man, dass die eigene Funktion - ausgenommen Vorzeichenfehler - nicht stark abweicht.

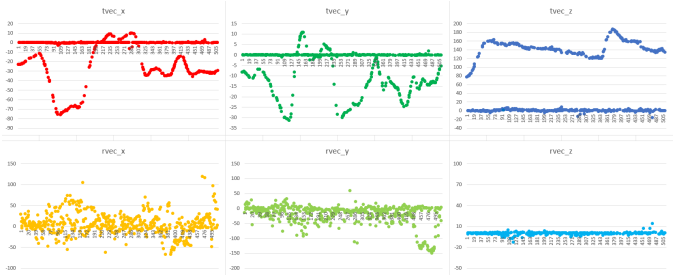

Fig. 6: Output Graph Error RANSAC (1000,2)

Um gegen das bestehende Rauschen der eigenen Implementation vorzugehen empfiehlt es sich, ein Filter über die Zeit einzusetzen. Der zeitliche Filter zielt darauf ab, die zeitliche Konsistenz der Pose-Schätzungen über aufeinanderfolgende Frames zu nutzen und so die Auswirkungen von Rauschen und Ausreißern zu reduzieren. Durch Berücksichtigung der Verlaufshistorie der Pose-Schätzungen bietet der zeitliche Filter eine Möglichkeit, die Schätzungen zu glätten und so ein optimirteres Ergebnis zu erzielen. Abschließen bleibt zu sagen, dass der selbst implementierte RANSAC in Vergleich zu cv::solvePnP ein ähnliches aber kein qualitatives besseres Ergebnis liefert. Jedoch sticht die eigene Implementierung, wie von einem RANSAC Algorithmus üblichen Eigenschaft erwartet, durch ihre Robustheit gegen Ausreißer heraus. Aus diesem Grund ist der RANSAC Algorithmus eine legitime Herangehensweise in Bezug auf Vorverarbeitung für die Pose Estimation.

VI. Zusammenfassung

Dieses Paper beschäftigt sich mit der Implementierung von zwei Methoden zur Pose Estimation, SolvePnP und RANSAC, und vergleicht deren Ergebnisse. SolvePnP nutzt die Korrespondenz zwischen 2D und 3D Punkten zur Berechnung der Pose, während RANSAC ein Schätzverfahren verwendet, welches auf zufälligen Stichproben basiert und eine Handhabung von Ausreißern ermöglicht. Die selbstgeschriebene RANSAC Methode wurde mit unterschiedlicher Parametrisierung durchgeführt. Die Ergebnisse zeigen, dass die selbst implementierte Methode der cv::solvePnP-Methode nahe kommt, wobei viele Abweichungen sehr gering und fast nicht zu erkennen sind. Es gibt jedoch immer noch einige Vorzeichenfehler aufgrund der Natur des RANSAC-Algorithmus. Eine Erhöhung der Anzahl der Iterationen und Senkung der Reprojektionsfehler kann die Leistung des RANSAC-Algorithmus verbessern, erhöht jedoch auch die Rechenkosten.

References

[1] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, “Complete solution classification for the perspective-three-point problem,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 930–943, 2003.

[2] D. H. Lee, S. S. Lee, H. H. Kang, and C. K. Ahn, "Camera position estimation for uavs using solvepnp with kalman filter," pp. 250–251, 2018.
[3] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, 02 2009.
[4] S. Lin, H. F. Cheng, W. Li, Z. Huang, P. Hui, and C. Peylo, "Ubii: Physical world interaction through augmented reality," *IEEE Transactions on Mobile Computing*, vol. 16, no. 3, pp. 872–885, 2017.
[5] OpenCV. cvfindchessboardcorners. [Online]. Available: https://docs.adaptive-vision.com/current/studio/filters/CameraCalibrationAnd3DReconstruction/cvFindChessboardCorners.html
[6] ——. Perspective-n-point (pnp) pose computation. [Online]. Available: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html
[7] C. Papazov and D. Burschka, "An efficient ransac for 3d object recognition in noisy and occluded scenes." 01 2010, pp. 135–148.
[8] C. Stachniss. Camera calibration: Direct linear transform. [Online]. Available: https://www.ipb.uni-bonn.de/html/teaching/photo12-2021/2021-pho1-21-DLT.pptx.pdf
[9] F. Vicente, Z. Huang, X. Xiong, F. De la Torre, W. Zhang, and D. Levi, "Driver gaze tracking and eyes off the road detection system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 1–14, 08 2015.
[10] W. Wöber and S. Simon. Mvsr slides termin1.