Laporan Kalibrasi Kamera Menggunakan OpenCV dan Python

Agus Fuad Mudhofar
Muhammad Syawal Ridho
Hasan Palito

23 Juni 2025

Daftar Isi

1 Abstrak

Kalibrasi kamera adalah proses fundamental dalam visi komputer yang bertujuan untuk menentukan parameter intrinsik dan ekstrinsik kamera. Parameter intrinsik menggambarkan karakteristik optik dan geometris internal kamera (misalnya, panjang fokus, titik utama, dan distorsi lensa), sedangkan parameter ekstrinsik mendefinisikan orientasi (rotasi) dan posisi (translasi) kamera relatif terhadap sistem koordinat dunia. Laporan ini menjelaskan secara rinci proses kalibrasi kamera menggunakan pustaka OpenCV di Python dengan pola papan catur (checkerboard) sebagai objek kalibrasi. Setiap baris kode dan konsep yang terlibat akan diuraikan untuk memberikan pemahaman yang komprehensif.

2 Pendahuluan

Kamera digital, meskipun merupakan alat canggih untuk menangkap informasi visual, tidak sempurna. Lensa kamera dapat menyebabkan distorsi geometris pada gambar yang ditangkap. Selain itu, untuk aplikasi visi komputer yang memerlukan pengukuran metrik atau rekonstruksi 3D, penting untuk mengetahui bagaimana piksel dalam gambar 2D berhubungan dengan koordinat 3D di dunia nyata. Proses kalibrasi kamera mengatasi masalah ini dengan memodelkan kamera secara matematis.

2.1 Mengapa Kalibrasi Kamera Penting?

Kalibrasi kamera sangat penting untuk berbagai aplikasi, termasuk:

a. **Pengukuran 3D:** Memperkirakan ukuran dan jarak objek nyata dari gambar.

b. **Rekonstruksi 3D:** Membuat model 3D dari lingkungan atau objek menggunakan satu atau lebih gambar.

c. **Navigasi Robot:** Memungkinkan robot untuk memahami lingkungannya dan bernavigasi secara akurat.

d. **Augmented Reality (AR):** Menempatkan objek virtual secara realistis ke dalam tayangan dunia nyata.

e. **Pelacakan Objek:** Melacak pergerakan objek dalam ruang 3D.

f. **Koreksi Distorsi Gambar:** Menghilangkan efek distorsi lensa untuk mendapatkan gambar yang lebih akurat secara geometris.

2.2 Metode Kalibrasi dengan Pola Papan Catur

Salah satu metode yang paling umum digunakan untuk kalibrasi kamera adalah dengan menggunakan pola kalibrasi dengan fitur yang mudah dideteksi, seperti papan catur. Keuntungan menggunakan papan catur adalah sudut-sudut internalnya (pertemuan antara kotak hitam dan putih) dapat dideteksi dengan presisi tinggi. Dengan mengambil beberapa gambar papan catur dari berbagai sudut pandang dan jarak, kita dapat mengumpulkan cukup data untuk menghitung parameter kamera.

3 Metodologi: Penjelasan Kode Python

Kode Python yang diberikan melakukan proses kalibrasi kamera langkah demi langkah. Berikut adalah penjelasan rinci untuk setiap bagian kode.

3.1 Inisialisasi dan Persiapan

Bagian awal kode mengimpor pustaka yang diperlukan dan mendefinisikan parameter awal.

```
1  import cv2
2  import numpy as np
3  import time
4
5  CHECKERBOARD = (8, 5)
6
7  criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30,
       0.001)
8
9  objp = np.zeros((CHECKERBOARD[0]*CHECKERBOARD[1], 3), np.float32)
10
11 objp[:, :2] = np.mgrid[0:CHECKERBOARD[0], 0:CHECKERBOARD[1]].T.reshape
       (-1, 2)
12
13 objpoints = []
14 imgpoints = []
15
16 cap = cv2.VideoCapture(0)
17 img_count = 0
18 last_capture_time = 0
```

Listing 1: Inisialisasi Pustaka dan Parameter

Penjelasan Kode (1):

Baris 1-3: Mengimpor pustaka OpenCV (‘cv2‘) untuk fungsi visi komputer, NumPy (‘np‘) untuk operasi numerik, dan ‘time‘ untuk mengontrol laju pengambilan gambar.

Baris 5: ‘CHECKERBOARD‘ adalah tuple yang mendefinisikan jumlah sudut internal pada papan catur yang akan dideteksi. Misalnya, ‘(8, 5)‘ berarti 8 sudut secara horizontal dan 5 sudut secara vertikal. Ini adalah jumlah persimpangan antara kotak hitam dan putih, bukan jumlah kotak itu sendiri.

Baris 7: ‘criteria‘ mendefinisikan kondisi penghentian untuk algoritma iteratif ‘cv2.cornerSubPix()‘. Algoritma ini akan berhenti jika akurasi yang diinginkan (‘0.001‘) tercapai atau setelah jumlah iterasi maksimum (‘30‘) dilakukan.

Baris 9: ‘objp‘ adalah array NumPy yang akan menyimpan koordinat 3D dari sudut-sudut papan catur. Kita mendefinisikan sistem koordinat dunia (objek) di mana papan catur terletak pada bidang XY (yaitu, $Z = 0$). Ukuran array ini adalah (jumlah total sudut $\times$ 3) karena setiap titik memiliki koordinat (X, Y, Z).

Baris 11: `objp[:, :2]` diisi dengan koordinat (X, Y) dari sudut-sudut. `np.mgrid[0:CHECKERBOARD[0], 0:CHECKERBOARD[1]]` menghasilkan dua array 2D untuk X dan Y. menghasilkan dua array 2D yang mewakili koordinat X dan Y.

Setelah dilakukan `.T` (transpose) dan `.reshape(-1, 2)`, hasilnya adalah daftar koordinat $(0,0),(1,0),\ldots,(\text{lebar}-1,\text{tinggi}-1)$. Koordinat Z tetap 0. Satuan dari koordinat ini bisa apa saja (misalnya, mm, cm, atau unit kotak), selama konsisten.

Baris 13: 'objpoints' akan menyimpan daftar array 'objp' untuk setiap gambar kalibrasi yang berhasil. 'imgpoints' akan menyimpan daftar koordinat 2D (piksel) dari sudut-sudut papan catur yang terdeteksi di setiap gambar.

Baris 16: 'cv2.VideoCapture(0)' menginisialisasi objek penangkapan video dari kamera pertama yang terhubung ke sistem.

Baris 17: 'img_count' adalah variabel yang bisa digunakan untuk melacak jumlah gambar yang telah digunakan, meskipun dalam kode ini tidak secara aktif digunakan untuk membatasi jumlah gambar.

Baris 18: 'last_capture_time' digunakan untuk memastikan ada jeda waktu antara pengambilan gambar papan catur, sehingga kita mendapatkan pose papan catur yang beragam.

3.2 Pengambilan Gambar dan Deteksi Fitur

Loop utama program menangkap frame dari kamera, mencoba mendeteksi papan catur, dan jika berhasil, menyimpan titik-titik yang relevan.

```
while True:
    ret, frame = cap.read()
    if not ret:
        print("Gagal membaca frame dari kamera.")
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    found, corners = cv2.findChessboardCorners(gray, CHECKERBOARD, cv2.
        CALIB_CB_ADAPTIVE_THRESH + cv2.CALIB_CB_FAST_CHECK)

    now = time.time()

    if found:
        corners2 = cv2.cornerSubPix(gray, corners, (11,11), (-1,-1),
            criteria)

        if (now - last_capture_time > 1):
            objpoints.append(objp)
            imgpoints.append(corners2)
            print(f"Papan catur terdeteksi, {len(imgpoints)} set data
                ditambahkan.")
            last_capture_time = now

        cv2.drawChessboardCorners(frame, CHECKERBOARD, corners2, found)

    frame_display = cv2.flip(frame, 1)

    cv2.imshow('Kalibrasi Kamera', frame_display)

    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        break
```

```
31      elif key == ord('s') and found:
32          objpoints.append(objp)
33          imgpoints.append(corners2)
34          print(f"Papan catur terdeteksi (manual), {len(imgpoints)} set
                data ditambahkan.")
35          last_capture_time = time.time()
```

Listing 2: Loop Utama: Pengambilan Gambar dan Deteksi Papan Catur

Penjelasan Kode (2):

Baris 1: Memulai loop tak terbatas untuk terus memproses frame dari kamera.

Baris 2: Membaca frame dari kamera menggunakan `cap.read()` dan menyimpannya ke dalam `ret` dan `frame`.

Baris 3-5: Jika frame gagal dibaca (`ret` bernilai `False`), maka keluar dari loop dan menampilkan pesan kesalahan.

Baris 7: Mengonversi frame ke dalam format grayscale untuk mempermudah deteksi pola.

Baris 9-12: Mendeteksi sudut-sudut papan catur menggunakan `cv2.findChessboardCorners()` dengan flag tambahan untuk threshold adaptif dan pengecekan cepat.

Baris 14: Menyimpan waktu saat ini ke dalam variabel `now`.

Baris 16: Mengecek apakah papan catur berhasil ditemukan.

Baris 17: Menyempurnakan posisi sudut dengan akurasi sub-piksel menggunakan `cv2.cornerSubPix()`.

Baris 19-23: Jika sudah lewat 1 detik dari pengambilan terakhir, maka titik-titik objek (`objp`) dan gambar (`corners2`) disimpan ke daftar.

Baris 25: Menampilkan visualisasi sudut papan catur pada frame asli menggunakan `cv2.drawChessboardCorners()`.

Baris 27: Membalik frame secara horizontal agar tampilan lebih natural seperti cermin.

Baris 29: Menampilkan frame ke layar menggunakan `cv2.imshow()`.

Baris 31-35: Menunggu input dari keyboard:

a. Jika tombol `q` ditekan, keluar dari loop.

b. Jika tombol `s` ditekan dan papan catur ditemukan, data disimpan secara manual.

3.3 Melepaskan Sumber Daya

Setelah loop selesai (pengguna menekan 'q'), sumber daya kamera dilepaskan dan semua jendela OpenCV ditutup.

```
cap.release()
cv2.destroyAllWindows()
```

Listing 3: Melepaskan Sumber Daya

Penjelasan Kode (3):

Baris 1: `cap.release()` melepaskan kamera, membuatnya tersedia untuk aplikasi lain.

Baris 2: `cv2.destroyAllWindows()` menutup semua jendela yang dibuat oleh OpenCV.

3.4 Proses Kalibrasi Kamera

Setelah mengumpulkan cukup banyak pasangan titik objek (3D) dan titik gambar (2D) dari berbagai pandangan, fungsi 'cv2.calibrateCamera()' dipanggil untuk menghitung parameter kamera.

```
if len(objpoints) > 0 and len(imgpoints) > 0:
    ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints,
        imgpoints, gray.shape[::-1], None, None)

    print("\n--- Hasil Kalibrasi Kamera ---")
    print("Status Keberhasilan (ret):", ret)
    print("\nMatriks Kamera (Intrinsik):\n", mtx)
    print("\nKoefisien Distorsi:\n", dist)

    np.savez("hasil_kalibrasi_kamera.npz", mtx=mtx, dist=dist, rvecs=
        rvecs, tvecs=tvecs)
    print("\nHasil kalibrasi disimpan ke 'hasil_kalibrasi_kamera.npz'")

    mean_error = 0
    for i in range(len(objpoints)):
        imgpoints2, _ = cv2.projectPoints(objpoints[i], rvecs[i], tvecs
            [i], mtx, dist)
        error = cv2.norm(imgpoints[i], imgpoints2, cv2.NORM_L2)/len(
            imgpoints2)
        mean_error += error
    print(f"\nTotal Error Reproyeksi Rata-rata: {mean_error/len(
        objpoints)}")

else:
    print("Tidak ada data titik yang cukup untuk melakukan kalibrasi.")
```

Listing 4: Kalibrasi Kamera

Penjelasan Kode (4):

Baris 1: Mengecek apakah terdapat cukup data dalam `objpoints` dan `imgpoints` untuk melakukan kalibrasi kamera.

Baris 2: Fungsi `cv2.calibrateCamera()` melakukan kalibrasi kamera berdasarkan data titik 3D dan 2D.

a. `objpoints`: Titik-titik 3D dari dunia nyata (koordinat papan catur).

b. `imgpoints`: Titik-titik 2D hasil proyeksi pada gambar.

c. `gray.shape[::-1]`: Ukuran gambar (lebar, tinggi).

d. Dua `None`: Untuk estimasi otomatis `cameraMatrix` dan `distCoeffs`.

Output dari `cv2.calibrateCamera()`:

a. `ret`: RMS (Root Mean Square) error reproyeksi.

b. `mtx`: Matriks kamera intrinsik berukuran 3×3.

c. `dist`: Koefisien distorsi lensa.

d. `rvecs` dan `tvecs`: Parameter rotasi dan translasi (ekstrinsik) dari kamera ke papan catur.

Baris 4-7: Menampilkan hasil kalibrasi ke terminal, termasuk matriks kamera dan distorsi.

Baris 9-10: Menyimpan hasil kalibrasi ke file `.npz` agar dapat digunakan di kemudian hari.

Baris 12-17: Menghitung **error reproyeksi** rata-rata sebagai evaluasi akurasi kalibrasi.

a. Titik objek 3D diproyeksikan ke gambar.

b. Dibandingkan dengan titik asli, lalu dihitung rata-rata error-nya.

Baris 19-20: Jika data tidak cukup, maka menampilkan pesan bahwa kalibrasi tidak dapat dilakukan.

4 Memahami Parameter Kamera

Hasil utama dari proses kalibrasi adalah matriks kamera intrinsik dan koefisien distorsi.

4.1 Parameter Intrinsik (Matriks Kamera)

Matriks kamera, sering dilambangkan sebagai K atau 'mtx', menghubungkan koordinat 3D dari sebuah titik dalam sistem koordinat kamera ke koordinat piksel 2D dalam gambar. Matriks ini biasanya berbentuk:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Di mana:

a. f_x dan f_y adalah panjang fokus kamera dalam satuan piksel. $f_x = F \cdot k_x$ dan $f_y = F \cdot k_y$, di mana F adalah panjang fokus dalam satuan dunia (misalnya, mm), dan k_x, k_y adalah jumlah piksel per satuan panjang pada sumbu x dan y sensor.

b. c_x dan c_y adalah koordinat titik utama (pusat optik) dalam satuan piksel. Ini adalah titik di mana sumbu optik kamera memotong bidang gambar.

c. s adalah faktor kemiringan (skew coefficient) antara sumbu x dan y. Untuk sebagian besar kamera modern, s sangat dekat dengan 0 dan sering diabaikan.

Dalam output OpenCV, jika tidak ada asumsi khusus yang dibuat (misalnya, piksel persegi atau tidak ada kemiringan), matriks kamera akan berbentuk:

$$\text{mtx} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Jika diasumsikan piksel persegi ($k_x = k_y$), maka $f_x = f_y$.

4.2 Koefisien Distorsi

Lensa kamera nyata tidak sempurna dan dapat menyebabkan distorsi geometris pada gambar. Dua jenis distorsi utama adalah:

a. **Distorsi Radial:** Disebabkan oleh bentuk lensa. Garis lurus di dunia nyata mungkin tampak melengkung di dekat tepi gambar. Distorsi ini bisa berupa *barrel distortion* (gambar tampak melotot keluar) atau *pincushion distortion* (gambar tampak terjepit ke dalam). Dimodelkan oleh koefisien $k_1, k_2, k_3, \ldots$.

b. **Distorsi Tangensial:** Disebabkan oleh perakitan lensa yang tidak sempurna, di mana lensa tidak sejajar secara sempurna dengan bidang gambar. Dimodelkan oleh koefisien p_1, p_2.

Vektor koefisien distorsi ‘dist‘ biasanya berisi $(k_1, k_2, p_1, p_2, k_3)$. Beberapa model dapat menyertakan lebih banyak koefisien (k_4, k_5, k_6, dll.). Misalkan (x, y) adalah koordinat titik terdistorsi yang ideal (tanpa distorsi) dan (x_d, y_d) adalah koordinat titik terdistorsi yang diamati. Hubungannya dapat dimodelkan sebagai:

$$x_d = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + [2p_1 xy + p_2(r^2 + 2x^2)]$$

$$y_d = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

di mana $r^2 = x^2 + y^2$. Koordinat (x, y) dinormalisasi relatif terhadap titik utama.

4.3 Parameter Ekstrinsik (Vektor Rotasi dan Translasi)

Parameter ekstrinsik (‘rvecs‘ dan ‘tvecs‘) mendefinisikan transformasi dari sistem koordinat dunia (di mana papan catur didefinisikan) ke sistem koordinat kamera untuk setiap tampilan papan catur.

a. ‘rvecs‘: Daftar vektor rotasi. Setiap vektor rotasi dapat dikonversi menjadi matriks rotasi 3×3 (misalnya, menggunakan ‘cv2.Rodrigues()‘). Matriks rotasi R menggambarkan orientasi papan catur relatif terhadap kamera.

b. ‘tvecs‘: Daftar vektor translasi. Setiap vektor translasi T menggambarkan posisi asal sistem koordinat papan catur relatif terhadap asal sistem koordinat kamera.

Transformasi dari titik dunia $P_w = (X_w, Y_w, Z_w)^T$ ke titik kamera $P_c = (X_c, Y_c, Z_c)^T$ diberikan oleh:

$$P_c = R \cdot P_w + T$$

5 Algoritma Kalibrasi

OpenCV menggunakan variasi dari metode yang dipopulerkan oleh Zhang (2000). Ide dasarnya adalah sebagai berikut:

a. **Deteksi Fitur:** Deteksi titik-titik fitur yang diketahui (sudut papan catur) pada beberapa gambar pola kalibrasi yang diambil dari orientasi berbeda.

b. **Estimasi Homografi:** Untuk setiap gambar, hitung homografi yang memetakan titik-titik 3D pada pola (di bidang $Z = 0$) ke titik-titik 2D yang terdeteksi pada gambar.

c. **Estimasi Parameter Intrinsik:** Dengan menggunakan batasan dari homografi, parameter intrinsik kamera dapat diestimasi.

d. **Estimasi Parameter Ekstrinsik:** Setelah parameter intrinsik diketahui, parameter ekstrinsik (rotasi dan translasi) untuk setiap tampilan dapat dihitung.

e. **Optimasi Non-Linear:** Semua parameter (intrinsik dan ekstrinsik) disempurnakan secara bersamaan dengan meminimalkan *error reproyeksi*. Error reproyeksi adalah jarak antara titik gambar yang diamati dan titik gambar yang diproyeksikan (menggunakan estimasi parameter saat ini). Algoritma Levenberg-Marquardt sering digunakan untuk optimasi ini.

6 Penggunaan Hasil Kalibrasi

Setelah parameter kamera (‘mtx‘ dan ‘dist‘) diketahui, mereka dapat digunakan untuk:

a. **Koreksi Distorsi (Undistortion):** Gambar yang terdistorsi dapat dikoreksi menggunakan ‘cv2.undistort()‘.

```
data = np.load('hasil_kalibrasi_kamera.npz')
mtx = data['mtx']
dist = data['dist']

img_distorted = cv2.imread('gambar_terdistorsi.jpg')
h, w = img_distorted.shape[:2]

newcameramtx, roi = cv2.getOptimalNewCameraMatrix(mtx, dist, (w,h),
    1, (w,h))

dst_img = cv2.undistort(img_distorted, mtx, dist, None,
   newcameramtx)


cv2.imshow('Gambar Asli Terdistorsi', img_distorted)
cv2.imshow('Gambar Setelah Undistortion', dst_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Listing 5: Contoh Undistortion

b. **Estimasi Pose Objek:** Menentukan posisi dan orientasi objek 3D yang diketahui relatif terhadap kamera menggunakan ‘cv2.solvePnP()‘.

c. **Rekonstruksi 3D:** Menggabungkan informasi dari beberapa tampilan (misalnya, dalam sistem stereo) untuk merekonstruksi struktur 3D adegan.

7 Tips untuk Kalibrasi yang Baik

a. **Jumlah Gambar:** Gunakan setidaknya 10-20 gambar papan catur. Lebih banyak gambar biasanya lebih baik, asalkan beragam.

b. **Variasi Pose:** Ambil gambar papan catur dari berbagai sudut, jarak, dan posisi. Pastikan papan catur mengisi sebagian besar bidang pandang kamera dalam beberapa gambar, dan juga muncul di berbagai bagian gambar (tengah, tepi, sudut). Variasikan kemiringan papan catur (pitch, yaw, roll).

c. **Pencahayaan yang Baik:** Pastikan papan catur diterangi dengan baik dan merata untuk memudahkan deteksi sudut. Hindari bayangan yang kuat atau pantulan silau pada papan catur.

d. **Papan Catur yang Rata:** Gunakan papan catur yang kaku dan rata. Papan yang melengkung akan menghasilkan kalibrasi yang tidak akurat.

e. **Fokus Stabil:** Atur fokus kamera secara manual dan jaga agar tetap konstan selama proses pengambilan gambar. Jika menggunakan autofokus, pastikan ia mengunci dengan benar pada papan catur.

f. **Resolusi Penuh:** Lakukan kalibrasi pada resolusi kamera penuh yang akan Anda gunakan untuk aplikasi Anda.

g. **Periksa Error Reproyeksi:** Setelah kalibrasi, periksa error reproyeksi rata-rata. Nilai di bawah 1.0 piksel umumnya dianggap baik, tetapi ini bisa bergantung pada aplikasi.

8 Kesimpulan

Kalibrasi kamera adalah langkah penting untuk banyak aplikasi visi komputer. Dengan menggunakan papan catur dan fungsi-fungsi yang disediakan oleh OpenCV, kita dapat secara efektif menghitung parameter intrinsik dan ekstrinsik kamera. Kode Python yang dianalisis dalam laporan ini menyediakan implementasi praktis dari proses tersebut, mulai dari pengambilan gambar hingga penyimpanan hasil kalibrasi. Pemahaman yang baik tentang parameter ini dan bagaimana mereka diperoleh memungkinkan pengembangan aplikasi visi komputer yang lebih akurat dan andal.

Referensi

1. Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(11), 1330-1334.

2. Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Reilly Media, Inc.

3. Dokumentasi OpenCV Camera Calibration: https://docs.opencv.org/master/d4/d94/tutorial_camera_calibration.html