Laporan Kalibrasi Kamera Menggunakan OpenCV dan Python

Agus Fuad Mudhofar
Muhammad Syawal Ridho
Hasan

1 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:

- **Pengukuran 3D:** Memperkirakan ukuran dan jarak objek nyata dari gambar.
- **Rekonstruksi 3D:** Membuat model 3D dari lingkungan atau objek menggunakan satu atau lebih gambar.
- **Navigasi Robot:** Memungkinkan robot untuk memahami lingkungannya dan bernavigasi secara akurat.
- **Augmented Reality (AR):** Menempatkan objek virtual secara realistis ke dalam tayangan dunia nyata.
- **Pelacakan Objek:** Melacak pergerakan objek dalam ruang 3D.
- **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  # Mendefinisikan ukuran papan catur (jumlah sudut internal per baris
       dan kolom)
6  CHECKERBOARD = (8, 5) # (jumlah sudut horizontal, jumlah sudut vertikal
       )
7  # Kriteria untuk penghentian algoritma iteratif (cornerSubPix)
8  # cv2.TERM_CRITERIA_EPS: berhenti jika error epsilon tercapai
9  # cv2.TERM_CRITERIA_MAX_ITER: berhenti setelah jumlah iterasi maksimum
10 # 30: jumlah iterasi maksimum
11 # 0.001: epsilon (akurasi yang diinginkan)
12 criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30,
       0.001)
13
14 # Membuat array untuk menyimpan koordinat 3D (titik objek) dari sudut
       papan catur
15 # di sistem koordinat dunia (objek).
16 # Asumsikan papan catur terletak pada bidang Z=0.
17 objp = np.zeros((CHECKERBOARD[0]*CHECKERBOARD[1], 3), np.float32)
18 # Mengisi koordinat X dan Y (misalnya, (0,0,0), (1,0,0), ..., (7,4,0))
19 # np.mgrid membuat grid koordinat.
20 # .T mentransposisi grid.
21 # .reshape(-1, 2) mengubahnya menjadi array N baris x 2 kolom.
22 objp[:, :2] = np.mgrid[0:CHECKERBOARD[0], 0:CHECKERBOARD[1]].T.reshape
       (-1, 2)
23
24 # Array untuk menyimpan titik objek (3D) dan titik gambar (2D) dari
       semua frame
25 objpoints = [] # Titik 3D dalam ruang dunia nyata
26 imgpoints = [] # Titik 2D yang sesuai dalam bidang gambar
27
28 # Menginisialisasi penangkapan video dari kamera default (indeks 0)
29 cap = cv2.VideoCapture(0)
30 img_count = 0 # Penghitung gambar yang berhasil ditambahkan (tidak
       digunakan di kode ini)
31 last_capture_time = 0  # Waktu terakhir gambar ditambahkan untuk
       kalibrasi
```

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 6:** ‘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 9-12:** ‘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 16:** ‘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 20-23:** `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 26-27:** ‘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 30:** ‘cv2.VideoCapture(0)‘ menginisialisasi objek penangkapan video dari kamera pertama yang terhubung ke sistem.
- **Baris 31:** ‘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 32:** ‘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:
    # Membaca frame dari kamera
    ret, frame = cap.read()
    if not ret: # Jika frame tidak berhasil dibaca, keluar dari loop
        print("Gagal membaca frame dari kamera.")
        break

    # Mengonversi frame ke skala abu-abu (grayscale)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Mencari sudut papan catur pada gambar grayscale
    # cv2.CALIB_CB_ADAPTIVE_THRESH: Menggunakan threshold adaptif
```

```
13      # cv2.CALIB_CB_FAST_CHECK: Memeriksa gambar dengan cepat untuk
            keberadaan papan catur
14      found, corners = cv2.findChessboardCorners(gray, CHECKERBOARD,
15                                                 cv2.
                                                       CALIB_CB_ADAPTIVE_THRESH
                                                        +
16                                                 cv2.CALIB_CB_FAST_CHECK)
17
18      now = time.time() # Waktu saat ini
19
20      # Jika papan catur ditemukan
21      if found:
22          # Menyempurnakan lokasi sudut yang terdeteksi ke tingkat sub-
                piksel
23          # (11,11): ukuran window pencarian (winSize)
24          # (-1,-1): zona mati (zeroZone), -1 berarti tidak ada
25          corners2 = cv2.cornerSubPix(gray, corners, (11,11), (-1,-1),
                criteria)
26
27          # Menambahkan titik objek dan titik gambar jika jeda waktu
                cukup (1 detik)
28          if (now - last_capture_time > 1):
29              objpoints.append(objp)
30              imgpoints.append(corners2)
31              print(f"Papan catur terdeteksi, {len(imgpoints)} set data
                    ditambahkan.")
32              last_capture_time = now  # Perbarui waktu terakhir
                    penangkapan
33
34          # Menggambar sudut papan catur pada frame asli untuk
                visualisasi
35          cv2.drawChessboardCorners(frame, CHECKERBOARD, corners2, found)
36
37      # Membalik frame secara horizontal (efek cermin) untuk tampilan
            yang lebih intuitif
38      frame_display = cv2.flip(frame, 1)
39
40      # Menampilkan frame
41      cv2.imshow('Kalibrasi Kamera', frame_display)
42
43      # Menunggu tombol ditekan (1 milidetik)
44      key = cv2.waitKey(1) & 0xFF
45      if key == ord('q'): # Jika tombol 'q' ditekan, keluar dari loop
46          break
47      elif key == ord('s') and found: # Opsi untuk menyimpan secara
            manual (modifikasi)
48          objpoints.append(objp)
49          imgpoints.append(corners2)
50          print(f"Papan catur terdeteksi (manual), {len(imgpoints)} set
                data ditambahkan.")
51          last_capture_time = time.time() # Perbarui waktu untuk jeda
                berikutnya
```

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 3:** ‘cap.read()‘ membaca frame berikutnya dari kamera. ‘ret‘ adalah boolean yang menunjukkan apakah frame berhasil dibaca, dan ‘frame‘ adalah data gambar itu sendiri.
- **Baris 4-6:** Pemeriksaan jika ‘ret‘ adalah ‘False‘, yang berarti ada masalah dalam membaca frame.
- **Baris 9:** Gambar dikonversi ke skala abu-abu (‘gray‘) karena deteksi sudut papan catur biasanya lebih efisien dan akurat pada gambar monokrom.
- **Baris 13-16:** ‘cv2.findChessboardCorners(gray, CHECKERBOARD, flags)‘ mencoba menemukan sudut-sudut internal papan catur pada gambar ‘gray‘.
  - `CHECKERBOARD`: Ukuran papan catur yang telah didefinisikan.
  - `flags`: Parameter tambahan. `cv2.CALIB_CB_ADAPTIVE_THRESH` menggunakan thresholding adaptif untuk mengubah gambar menjadi hitam putih, yang dapat meningkatkan ketahanan terhadap kondisi pencahayaan yang berbeda. `cv2.CALIB_CB_FAST_CHECK` menjalankan tes cepat untuk keberadaan papan catur dan dapat mempercepat proses jika papan catur tidak ada.
  - `found`: Boolean yang bernilai `True` jika semua sudut papan catur ditemukan.
  - `corners`: Array yang berisi koordinat piksel dari sudut-sudut yang terdeteksi.
- **Baris 18:** ‘now = time.time()‘ mencatat waktu saat ini.
- **Baris 21:** Blok ‘if found:‘ dieksekusi jika papan catur berhasil dideteksi.
- **Baris 24:** ‘cv2.cornerSubPix(gray, corners, winSize, zeroZone, criteria)‘ menyempurnakan koordinat sudut yang ditemukan oleh ‘findChessboardCorners‘ ke akurasi sub-piksel. Ini penting untuk kalibrasi yang akurat.
  - ‘winSize‘: Ukuran jendela pencarian di sekitar setiap sudut (misalnya, 11×11 piksel).
  - ‘zeroZone‘: Setengah ukuran dari area mati di tengah jendela pencarian di mana jumlah dari produk skalar tidak dihitung. Nilai ‘(-1, -1)‘ berarti tidak ada zona mati.
  - ‘criteria‘: Kriteria penghentian yang telah didefinisikan sebelumnya.
  - ‘corners2‘: Array yang berisi koordinat sudut yang telah disempurnakan.
- **Baris 27-31:** Bagian ini memastikan bahwa data titik hanya ditambahkan jika telah berlalu lebih dari 1 detik sejak penangkapan terakhir yang berhasil. Ini bertujuan untuk mendapatkan pose papan catur yang beragam. Jika kondisi terpenuhi, `objp` (titik 3D dunia) dan `corners2` (titik 2D gambar) ditambahkan ke daftar `objpoints` dan `imgpoints`. Waktu `last_capture_time` diperbarui.
- **Baris 34:** `cv2.drawChessboardCorners(frame, CHECKERBOARD, corners2, found)` menggambar sudut-sudut yang terdeteksi (dan disempurnakan) pada frame berwarna asli. Ini berguna untuk visualisasi dan memastikan bahwa deteksi berjalan dengan benar.

- **Baris 37:** `frame_display = cv2.flip(frame, 1)` membalik frame secara horizontal. Ini memberikan efek cermin, yang seringkali lebih intuitif bagi pengguna saat melihat diri mereka sendiri atau lingkungan melalui kamera.
- **Baris 40:** `cv2.imshow('Kalibrasi Kamera', frame_display)` menampilkan frame (yang mungkin telah digambari sudut) di jendela bernama 'Kalibrasi Kamera'.
- **Baris 43-49:** `cv2.waitKey(1)` menunggu input keyboard selama 1 milidetik. Jika tombol 'q' ditekan (`ord('q')`), loop akan berhenti. Modifikasi ditambahkan: jika tombol 's' ditekan **dan** papan catur ditemukan, data akan disimpan secara manual. Ini memberi pengguna kontrol lebih besar atas gambar mana yang digunakan.

3.3 Melepaskan Sumber Daya

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

```
# Melepaskan objek penangkapan video
cap.release()
# Menutup semua jendela OpenCV
cv2.destroyAllWindows()
```

Listing 3: Melepaskan Sumber Daya

Penjelasan Kode (3):

- **Baris 2:** 'cap.release()' melepaskan kamera, membuatnya tersedia untuk aplikasi lain.
- **Baris 4:** '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.

```
# Melakukan kalibrasi kamera jika ada cukup data
if len(objpoints) > 0 and len(imgpoints) > 0:
    # gray.shape[::-1] memberikan (lebar, tinggi) gambar
    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)
    # rvecs dan tvecs adalah parameter ekstrinsik untuk setiap tampilan
        papan catur
    # print("\nVektor Rotasi (rvecs):\n", rvecs)
    # print("\nVektor Translasi (tvecs):\n", tvecs)

```

```
15      # Menyimpan hasil kalibrasi ke file .npz
16      np.savez("hasil_kalibrasi_kamera.npz", mtx=mtx, dist=dist, rvecs=
            rvecs, tvecs=tvecs)
17      print("\nHasil kalibrasi disimpan ke 'hasil_kalibrasi_kamera.npz'")
18
19      # Contoh menghitung error reproyeksi (opsional, untuk evaluasi)
20      mean_error = 0
21      for i in range(len(objpoints)):
22          imgpoints2, _ = cv2.projectPoints(objpoints[i], rvecs[i], tvecs
                [i], mtx, dist)
23          error = cv2.norm(imgpoints[i], imgpoints2, cv2.NORM_L2)/len(
                imgpoints2)
24          mean_error += error
25      print(f"\nTotal Error Reproyeksi Rata-rata: {mean_error/len(
            objpoints)}")
26
27  else:
28      print("Tidak ada data titik yang cukup untuk melakukan kalibrasi.")
```

Listing 4: Kalibrasi Kamera

Penjelasan Kode (4):

- **Baris 2:** Memeriksa apakah ada data yang cukup (‘objpoints‘ dan ‘imgpoints‘ tidak kosong) sebelum mencoba kalibrasi.
- **Baris 4-5:** ‘cv2.calibrateCamera(objpoints, imgpoints, imageSize, cameraMatrix, distCoeffs, ...)‘ adalah fungsi inti untuk kalibrasi.
  - ‘objpoints‘: Daftar titik 3D dunia.
  - ‘imgpoints‘: Daftar titik 2D gambar yang sesuai.
  - ‘gray.shape[::-1]‘: Ukuran gambar (lebar, tinggi). ‘gray.shape‘ menghasilkan (tinggi, lebar), jadi ‘[::-1]‘ membaliknya.
  - ‘None‘, ‘None‘: Argumen ini adalah untuk ‘cameraMatrix‘ dan ‘distCoeffs‘ awal. Jika ‘None‘, fungsi akan mengestimasi nilai-nilai ini.
- **Output dari ‘cv2.calibrateCamera()‘:**
  - ‘ret‘: Nilai boolean yang menunjukkan keberhasilan. Dalam konteks ini, lebih sering merujuk pada RMS (Root Mean Square) error reproyeksi. Semakin kecil nilainya, semakin baik kalibrasinya.
  - ‘mtx‘: Matriks kamera intrinsik 3×3.
  - ‘dist‘: Vektor koefisien distorsi.
  - ‘rvecs‘: Vektor rotasi (parameter ekstrinsik) untuk setiap tampilan papan catur. Ini adalah representasi Rodrigues dari matriks rotasi.
  - ‘tvecs‘: Vektor translasi (parameter ekstrinsik) untuk setiap tampilan papan catur.
- **Baris 7-13:** Mencetak hasil kalibrasi: status, matriks kamera, dan koefisien distorsi. Vektor rotasi dan translasi juga bisa dicetak jika diperlukan.

- **Baris 16-17:** Hasil kalibrasi (`mtx`, `dist`, `rvecs`, `tvecs`) disimpan ke dalam file `hasil_kalibrasi_kamera.npz` menggunakan `np.savez`. Ini memungkinkan parameter untuk dimuat dan digunakan nanti tanpa perlu mengulang proses kalibrasi.
- **Baris 20-25 (Opsional - Evaluasi Error):** Bagian ini menghitung *error reproyeksi*. Untuk setiap gambar kalibrasi, titik-titik objek 3D (`objpoints[i]`) diproyeksikan kembali ke bidang gambar menggunakan parameter kamera yang dihitung (`mtx`, `dist`) dan parameter ekstrinsik untuk tampilan tersebut (`rvecs[i]`, `tvecs[i]`). Hasilnya adalah `imgpoints2`. Kemudian, jarak Euclidean antara titik gambar yang diproyeksikan ulang (`imgpoints2`) dan titik gambar asli yang terdeteksi (`imgpoints[i]`) dihitung. Rata-rata error ini memberikan ukuran seberapa baik parameter yang ditemukan memodelkan kamera. Nilai error yang rendah (biasanya di bawah 1.0 piksel) menunjukkan kalibrasi yang baik.
- **Baris 27-28:** Jika tidak ada data yang cukup, pesan kesalahan dicetak.

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:

- 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.
- 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.
- 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:

- **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 $k_1, k_2, k_3, \ldots$.
- **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.

- ‘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.
- ‘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 (Singkat)

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

1. **Deteksi Fitur:** Deteksi titik-titik fitur yang diketahui (sudut papan catur) pada beberapa gambar pola kalibrasi yang diambil dari orientasi berbeda.
2. **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.

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

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

5. **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:

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

```
# Muat parameter kalibrasi (jika belum ada di memori)
# data = np.load('hasil_kalibrasi_kamera.npz')
# mtx = data['mtx']
# dist = data['dist']

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

# Dapatkan matriks kamera baru yang optimal (untuk cropping atau scaling)
# alpha=1 berarti semua piksel sumber dipertahankan, mungkin dengan piksel hitam tambahan.
# alpha=0 berarti piksel hasil yang tidak valid dipotong.
newcameramtx, roi = cv2.getOptimalNewCameraMatrix(mtx, dist, (w,h), 1, (w,h))

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

# Potong gambar berdasarkan ROI (Region of Interest) jika alpha=0 atau untuk menghilangkan piksel hitam
# x, y, w_roi, h_roi = roi
# dst_img = dst_img[y:y+h_roi, x:x+w_roi]

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

Listing 5: Contoh Undistortion

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

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

7 Tips untuk Kalibrasi yang Baik

- **Jumlah Gambar:** Gunakan setidaknya 10-20 gambar papan catur. Lebih banyak gambar biasanya lebih baik, asalkan beragam.
- **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).
- **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.
- **Papan Catur yang Rata:** Gunakan papan catur yang kaku dan rata. Papan yang melengkung akan menghasilkan kalibrasi yang tidak akurat.
- **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.
- **Resolusi Penuh:** Lakukan kalibrasi pada resolusi kamera penuh yang akan Anda gunakan untuk aplikasi Anda.
- **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

- Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(11), 1330-1334.
- Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Reilly Media, Inc.

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