LAPORAN TEORI

PENGOLAHAN CITRA DIGITAL

NAMA : Felix Ewaldo Panggabean.

NIM : 202331084.

KELAS : C.

DOSEN : Ir. Darma Rusjdi, M.Kom.

NO.PC : 11.

ASISTEN : 1. Abdur Rasyid Ridho.

2. Rizqy Amanda.

3. Kashrina Masyid Azka.

4. Izzat Islami Kagapi.

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2025

1. Apa perbedaan antara operator deteksi tepi Sobel, Prewitt, dan Canny ?. Jelaskan keunggulan dan kelemahan masing-masing !.

Dalam pengolahan citra digital, operator deteksi tepi Sobel, Prewitt, dan Canny adalah teknik yang digunakan untuk menemukan batas atau kontur objek dalam sebuah gambar. Setiap teknik memiliki cara kerja dan fitur yang berbeda. Metode berbasis gradien digunakan oleh Sobel dan Prewitt untuk menghitung perubahan intensitas piksel baik di arah horizontal maupun vertikal. Sementara **Prewitt** menggunakan bobot yang merata, **Sobel** memberikan bobot yang lebih besar pada piksel pusat mask-nya, yang menghasilkan tepi yang lebih tajam namun lebih sensitif terhadap gangguan atau suara. Keduanya cepat dan sederhana dan cocok untuk kebutuhan dasar atau sistem yang memiliki keterbatasan komputasi. Namun, mereka kurang akurat dalam menangkap detail halus atau tepi yang sangat tipis.

Operator Canny, di sisi lain, lebih canggih karena melibatkan beberapa langkah : perhitungan gradien, penipisan tepi (non-maximum suppression), perataan gambar dengan filter Gaussian, dan thresholding ganda. Hasil Canny biasanya lebih akurat, tepinya lebih bersih, dan mampu mendeteksi garis halus sekaligus tahan terhadap suara, menjadikannya sangat cocok untuk aplikasi yang membutuhkan pendeteksian tepi yang berkualitas tinggi. Namun, kelemahan utamanya adalah waktu proses yang lebih lama dan kompleksitas perhitungan yang lebih tinggi dibandingkan Sobel dan Prewitt.

2. Apa perbedaan antara transformasi translasi, rotasi, dan skala dalam transformasi geometrik citra ?. Berikan contohnya masing-masing !.

Transformasi translasi, rotasi, dan skala adalah tiga jenis transformasi geometrik dasar dalam pengolahan citra yang digunakan untuk mengubah posisi, orientasi, atau ukuran suatu objek dalam gambar. **Translasi** adalah proses memindahkan seluruh citra atau objek dari satu posisi ke posisi lain tanpa mengubah bentuk, arah, atau ukurannya. **Misalnya,** jika sebuah gambar kotak dipindahkan 50 piksel ke kanan dan 30 piksel ke bawah, maka itulah translasi—seluruh isi gambar bergeser sesuai arah dan jarak tertentu. **Rotasi** adalah proses memutar citra terhadap titik pusat tertentu (biasanya titik pusat gambar) dengan sudut tertentu. **Sebagai contoh,** memutar sebuah gambar wajah sebesar 45 derajat searah jarum jam adalah bentuk rotasi, di mana bentuk dan ukuran tetap, tapi arah dan posisi visualnya berubah. **Skala** adalah perubahan ukuran objek dalam gambar, bisa diperbesar atau diperkecil, baik secara horizontal, vertikal, maupun keduanya. **Contohnya,** jika sebuah gambar diperbesar dua kali lipat, maka semua pikselnya diperluas sehingga gambar tampak lebih besar. Ketiga transformasi ini penting dalam berbagai aplikasi, seperti pelacakan objek, pengenalan pola, atau penyusunan ulang tampilan citra, karena memungkinkan manipulasi posisi dan bentuk visual dengan tetap mempertahankan informasi struktural.

3. Apa yang dimaksud dengan transformasi afine dalam transformasi geometrik ?. Sebutkan minimal dua contohnya dalam aplikasi pemrosesan citra !.

Transformasi afine dalam transformasi geometrik adalah jenis transformasi linear yang mempertahankan hubungan garis lurus dan paralel antar titik, namun dapat mengubah posisi, ukuran, sudut, dan bentuk objek secara proporsional. Transformasi ini mencakup operasi

seperti : translasi, rotasi, skala, refleksi, dan shear dalam satu kerangka transformasi tunggal. Ciri khas dari transformasi afine adalah bahwa bentuk dasar seperti garis lurus tetap lurus, dan rasio jarak antar titik pada garis tetap terjaga, meskipun sudut dan ukuran objek bisa berubah. Dalam pemrosesan citra, transformasi afine sering digunakan pada aplikasi **koreksi perspektif,** misalnya saat memperbaiki tampilan foto yang diambil miring agar tampak tegak lurus, serta pada **registrasi citra,** yaitu menyelaraskan dua citra dari sudut pandang berbeda agar bisa dibandingkan atau digabung secara akurat. Transformasi afine sangat penting karena memberikan fleksibilitas dalam manipulasi citra tanpa merusak struktur geometris utamanya.

4. Bandingkan tujuan utama antara operasi deteksi tepi dengan operasi transformasi geometrik (resize dan rotasi). Meskipun keduanya sama-sama memanipulasi piksel, jelaskan mengapa hasil dan tujuan akhir dari kedua kelompok operasi tersebut sangat berbeda !.

Meskipun operasi deteksi tepi dan transformasi geometrik seperti resize dan rotasi sama-sama memanipulasi piksel dalam citra digital, keduanya memiliki tujuan utama dan hasil akhir yang sangat berbeda. **Deteksi tepi bertujuan untuk mengekstrak informasi struktural dari sebuah gambar, khususnya untuk menemukan batas-batas objek dengan menyoroti perubahan intensitas piksel yang tajam.** Proses ini penting dalam tahap analisis citra karena membantu mengidentifikasi bentuk, kontur, dan fitur penting dari objek, seperti dalam pengenalan wajah, pelacakan objek, atau segmentasi. **Sebaliknya, transformasi geometrik seperti resize dan rotasi berfokus pada perubahan tampilan visual dari gambar tanpa menambah atau menghapus informasi struktural.** Resize digunakan untuk memperbesar atau memperkecil ukuran gambar, sedangkan rotasi mengubah orientasi citra, misalnya untuk menyelaraskan sudut pandang atau tata letak. Jadi, walaupun sama-sama mengubah piksel, deteksi tepi lebih bersifat analitis dan berfungsi sebagai alat ekstraksi informasi, sedangkan transformasi geometrik bersifat manipulatif untuk kebutuhan presentasi, perataan, atau persiapan data sebelum dianalisis lebih lanjut.

5. Jelaskan bagaimana proses rotasi citra bekerja dari sudut pandang pemetaan piksel !. Mengapa setelah citra dirotasi dengan sudut selain 90° atau 180°, seringkali muncul area kosong (berwarna hitam) di bagian sudut-sudut citra ?.

Proses rotasi citra dari sudut pandang pemetaan piksel dilakukan dengan cara memetakan setiap piksel pada posisi baru ke posisi asalnya menggunakan rumus rotasi dua dimensi. Teknik ini disebut inverse mapping, di mana sistem menentukan dari mana asal tiap piksel dalam citra yang sudah diputar, lalu mengambil nilai intensitas piksel tersebut. **Ketika citra diputar dengan sudut selain 90° atau 180°, bentuk hasil rotasi tidak lagi sejajar dengan batas kotak gambar asli. Akibatnya, ada bagian-bagian di sudut citra yang tidak memiliki padanan piksel dari citra awal, sehingga tidak terisi nilai apa pun. Bagian inilah yang kemudian tampak sebagai area kosong atau berwarna hitam. Hal ini dikarenakan karena merupakan suatu konsekuensi dari rotasi geometrik yang memperluas area cakupan gambar tanpa mengisi data baru secara otomatis di luar batas gambar asli, sehingga seringkali timbul area kosong di sudut citra.**

LAPORAN PRAKTIKUM

PENGOLAHAN CITRA DIGITAL

NAMA : Felix Ewaldo Panggabean.

NIM : 202331084.

KELAS : C.

DOSEN : Ir. Darma Rusjdi, M.Kom.

NO.PC : 11.

ASISTEN : 1. Abdur Rasyid Ridho.

2. Rizqy Amanda.

3. Kashrina Masyid Azka.

4. Izzat Islami Kagapi.

INSTITUT TEKNOLOGI PLN

TEKNIK INFORMATIKA

2025

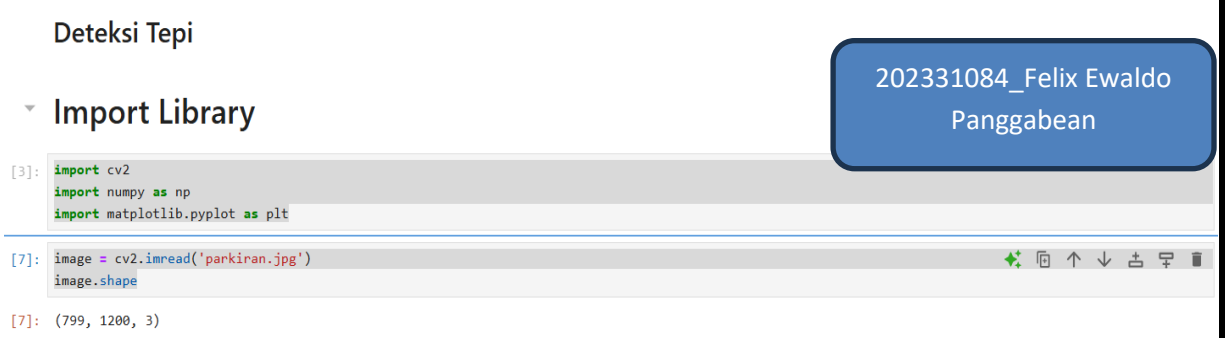
1. Olah gambar parkiran.jpg yang sudah diberikan dengan metode deteksi tepi yang sudah dipratikumkan.

Deteksi Tepi

Import Library

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread('parkiran.jpg')
image.shape
```

(799, 1200, 3)



Penjelasan :

Bagian "Import Library" :

- **import cv2 :** Baris ini mengimpor library OpenCV (Open Source Computer Vision Library). OpenCV adalah library yang sangat populer untuk pemrosesan gambar dan visi komputer.
- **import numpy as np :** Baris ini mengimpor library NumPy. NumPy adalah library fundamental untuk komputasi numerik di Python, terutama untuk bekerja dengan array (larik) dan matriks. Alias np untuk merujuk pada library ini. Dalam konteks pemrosesan gambar, gambar seringkali direpresentasikan sebagai array NumPy.
- **import matplotlib.pyplot as plt :** Baris ini mengimpor modul pyplot dari library Matplotlib. Matplotlib adalah library untuk membuat visualisasi statis, animasi, dan interaktif dalam Python. Pyplot menyediakan antarmuka seperti MATLAB untuk membuat plot. Alias plt digunakan untuk merujuk pada modul ini. Ini biasanya digunakan untuk menampilkan gambar atau hasil pemrosesan.

image = cv2.imread('parkiran.jpg') :

- **cv2.imread() :** Ini adalah fungsi dari OpenCV yang digunakan untuk membaca gambar dari file.
- **'parkiran.jpg' :** Ini adalah nama file gambar yang ingin dibaca.
- **image = ... :** Hasil dari fungsi cv2.imread() (yaitu, data gambar yang dibaca) disimpan ke dalam variabel bernama image. Variabel image ini sekarang akan berisi representasi numerik dari gambar tersebut (biasanya sebagai array NumPy).

image.shape :

- **image :** Ini adalah variabel yang menyimpan data gambar yang baru saja dibaca.
- **.shape :** Ini adalah atribut dari objek array NumPy. Atribut .shape mengembalikan sebuah tuple yang berisi dimensi-dimensi dari array tersebut. Untuk gambar berwarna, ini biasanya dalam format (tinggi, lebar, jumlah_saluran_warna).

(799, 1280, 3) : Ini adalah output dari baris image.shape :

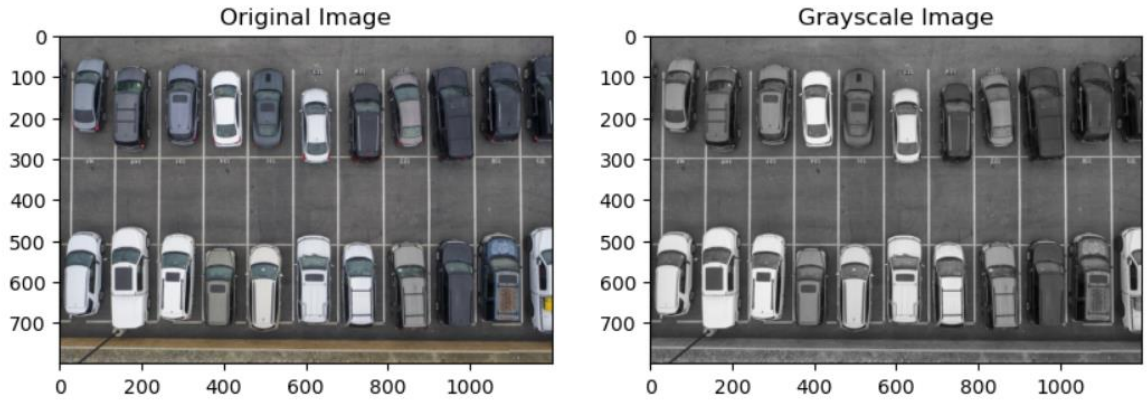
- **799 :** Tinggi gambar dalam piksel.
- **1280 :** Lebar gambar dalam piksel.
- **3 :** Jumlah saluran warna (channel). Angka 3 menunjukkan bahwa ini adalah gambar berwarna (biasanya BGR - Biru, Hijau, Merah - di OpenCV, bukan RGB). Jika merupakan gambar grayscale, jumlah saluran akan menjadi 1.

```
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

img_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

fig, axs = plt.subplots(1, 2, figsize=(10, 10))


ax = axs.ravel()

ax[0].imshow(image)

ax[0].set_title('Original Image')

ax[1].imshow(img_gray, cmap='gray')

ax[1].set_title('Grayscale Image')

Text(0.5, 1.0, 'Grayscale Image')
```

202331084_Felix Ewaldo Panggabean

```
[9]: image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
     img_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
     fig, axs = plt.subplots(1, 2, figsize=(10, 10))

     ax = axs.ravel()
     ax[0].imshow(image)
     ax[0].set_title('Original Image')
     ax[1].imshow(img_gray, cmap='gray')
     ax[1].set_title('Grayscale Image')
```

```
[9]: Text(0.5, 1.0, 'Grayscale Image')
```



Penjelasan :

img_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) :

- **cv2.cvtColor() :** Ini adalah fungsi dari OpenCV yang digunakan untuk mengubah ruang warna (color space) suatu gambar.
- **Image :** Ini adalah variabel yang berisi gambar asli yang sebelumnya telah dibaca (kemungkinan besar dalam format BGR, karena OpenCV secara default membaca gambar dalam BGR).
- **cv2.COLOR_BGR2RGB :** Ini adalah kode konversi yang memberitahu cvtColor untuk mengubah gambar dari format BGR (Biru, Hijau, Merah - urutan channel default di OpenCV) menjadi format RGB (Merah, Hijau, Biru - urutan channel yang umum digunakan oleh Matplotlib untuk menampilkan gambar).
- **img_rgb = ... :** Hasil konversi (gambar dalam format RGB) disimpan ke dalam variabel baru bernama img_rgb. Konversi ini penting agar Matplotlib dapat menampilkan warna gambar dengan benar, karena Matplotlib mengharapkan gambar dalam format RGB.

img_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) :

- **cv2.cvtColor() :** Fungsi konversi ruang warna yang sama.
- **Image :** Kembali menggunakan gambar asli yang dibaca (dalam format BGR).
- **cv2.COLOR_BGR2GRAY :** Ini adalah kode konversi yang memberitahu cvtColor untuk mengubah gambar dari format BGR menjadi gambar skala abu-abu (grayscale). Gambar grayscale hanya memiliki satu channel (intensitas piksel dari hitam ke putih).
- **img_gray = ... :** Hasil konversi (gambar grayscale) disimpan ke dalam variabel baru bernama img_gray.

fig, axs = plt.subplots(1, 2, figsize=(10, 8)) :

- **plt.subplots() :** Ini adalah fungsi dari Matplotlib yang digunakan untuk membuat figur (figure) dan sekumpulan sumbu (axes) dalam satu panggilan.
- **1 :** Argumen pertama menentukan jumlah baris subplot.
- **2 :** Argumen kedua menentukan jumlah kolom subplot.
- **figsize=(10, 8) :** Ini menentukan ukuran total figur dalam inci. 10 adalah lebar dan 8 adalah tinggi. Ini membantu mengontrol ukuran tampilan gambar yang dihasilkan.

fig, axs = ... : Fungsi ini mengembalikan dua objek :

- **fig :** Objek Figure, yang merupakan wadah tingkat atas untuk semua elemen plot.
- **Axs :** Objek Axes (atau array objek Axes jika ada beberapa subplot). Objek Axes adalah area plot individual di mana data sebenarnya di-plot. Karena terdapat 1 baris dan 2 kolom, axs akan menjadi array NumPy 1D yang berisi dua objek Axes.

ax = axs.ravel() :

- **axs.ravel() :** Metode .ravel() dari array NumPy mengubah array multidimensional menjadi array 1D. Memudahkan untuk mengiterasi atau mengakses setiap objek Axes secara berurutan.
- **ax = ... :** Variabel ax sekarang akan berisi array 1D dari objek Axes.

ax[0].imshow(img_rgb) :

- **ax[0] :** Mengakses subplot pertama (indeks 0) dari array ax.
- **.imshow(img_rgb) :** Ini adalah fungsi dari objek Axes yang digunakan untuk menampilkan gambar. img_rgb adalah gambar RGB yang akan ditampilkan di subplot pertama.

ax[0].set_title('Original Image') :

- **ax[0] :** Mengakses subplot pertama.
- **.set_title('Original Image') :** Mengatur judul untuk subplot pertama menjadi 'Original Image'.

ax[1].imshow(img_gray, cmap='gray') :

- **ax[1] :** Mengakses subplot kedua (indeks 1) dari array ax.
- **.imshow(img_gray, cmap='gray') :** Menampilkan img_gray (gambar grayscale) di subplot kedua.

cmap='gray' : Ini adalah argumen penting untuk gambar grayscale. cmap (colormap) menentukan bagaimana nilai-nilai intensitas piksel dipetakan ke warna. Dengan cmap='gray', Matplotlib akan menampilkan gambar dalam skala abu-abu yang benar. Jika ini tidak ditentukan, Matplotlib mungkin akan menggunakan colormap default yang akan menampilkan gambar grayscale dengan warna-warna yang tidak sesuai.

ax[1].set_title('Grayscale Image') :

- **ax[1] :** Mengakses subplot kedua.
- **.set_title('Grayscale Image') :** Mengatur judul untuk subplot kedua menjadi 'Grayscale Image'.

Penjelasan Output :

Gambar output menunjukkan dua subplot yang berdampingan :

"Original Image" : Adalah tampilan dari img_rgb. Terlihat gambar asli lahan parkir dengan semua warnanya utuh. Karena telah terkonversi gambar dari BGR ke RGB (yang merupakan format yang diharapkan Matplotlib), warna-warna di sini terlihat benar dan sesuai dengan gambar aslinya.

"Grayscale Image" : Merupakan tampilan dari img_gray. Gambar yang sama, tetapi semua informasi warnanya telah dihilangkan dan diganti dengan representasi intensitas (dari hitam ke putih). Objek dan detail dalam gambar masih dapat dikenali, tetapi tidak ada warna. Ini adalah hasil dari konversi cv2.COLOR_BGR2GRAY. Penggunaan cmap='gray' memastikan bahwa gambar ditampilkan dalam nuansa abu-abu yang benar.

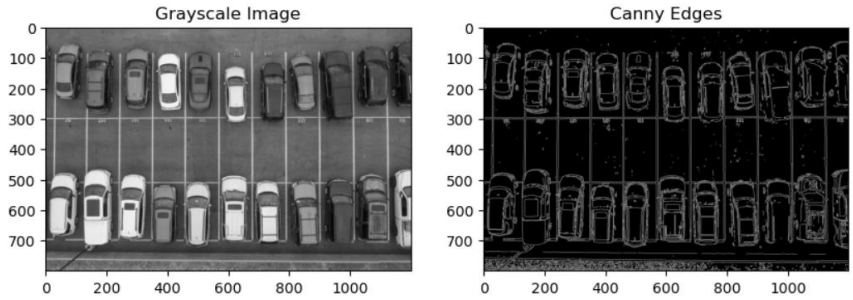
Mendeteksi ambang batas tepi menggunakan metode Canny

```
edges = cv2.Canny(img_gray, 100, 150)

fig, axs = plt.subplots(1, 2, figsize=(10, 10))

ax = axs.ravel()

ax[0].imshow(img_gray, cmap='gray')

ax[0].set_title('Grayscale Image')

ax[1].imshow(edges, cmap='gray')

ax[1].set_title('Canny Edges')

Text(0.5, 1.0, 'Canny Edges')
```

Mendeteksi ambang batas tepi menggunakan metode Canny

202331084_Felix Ewaldo Panggabean

```
[12]: edges = cv2.Canny(img_gray, 100, 150)
      fig, axs = plt.subplots(1, 2, figsize=(10, 10))
      ax = axs.ravel()
      ax[0].imshow(img_gray, cmap='gray')
      ax[0].set_title('Grayscale Image')
      ax[1].imshow(edges, cmap='gray')
      ax[1].set_title('Canny Edges')

[12]: Text(0.5, 1.0, 'Canny Edges')
```



Penjelasan :
edges = cv2.Canny(img_gray, 100, 150) :

- **cv2.Canny() :** Ini adalah fungsi dari OpenCV yang mengimplementasikan algoritma deteksi tepi Canny. Deteksi tepi Canny adalah algoritma multi-tahap yang populer untuk menemukan tepi yang luas dalam gambar.
- **img_gray :** Ini adalah gambar input yang akan diproses. Algoritma Canny biasanya bekerja paling baik pada gambar grayscale.
- **100 :** Ini adalah ambang batas (threshold) pertama atau ambang batas rendah (low threshold). Piksel dengan gradien intensitas di bawah nilai ini akan dianggap bukan tepi.
- **150 :** Ini adalah ambang batas kedua atau ambang batas tinggi (high threshold). Piksel dengan gradien intensitas di atas nilai ini pasti dianggap sebagai tepi. Piksel dengan gradien intensitas antara ambang batas rendah dan tinggi dianggap sebagai tepi hanya jika mereka terhubung ke piksel yang pasti merupakan tepi (yang berada di atas ambang batas tinggi).
- **edges = ... :** Hasil dari deteksi tepi Canny (gambar biner yang menunjukkan tepi) disimpan ke dalam variabel edges. Gambar ini akan memiliki piksel putih di lokasi tepi dan piksel hitam di tempat lain.

fig, axs = plt.subplots(1, 2, figsize=(10, 8)) : Membuat sebuah figur Matplotlib dengan 1 baris dan 2 kolom subplot, dengan ukuran figur 10x8 inci. Memungkinkan untuk menampilkan dua gambar secara berdampingan.

ax = axs.ravel() : Meratakan array axs (yang berisi objek Axes untuk setiap subplot) menjadi array 1D untuk kemudahan akses.

ax[0].imshow(img_gray, cmap='gray') :

- **ax[0] :** Mengakses subplot pertama.

- **.imshow(img_gray, cmap='gray') :** Menampilkan img_gray (gambar grayscale asli) di subplot pertama. cmap='gray' memastikan tampilan skala abu-abu yang benar.

ax[0].set_title('Grayscale Image') :

- **ax[0] :** Mengakses subplot pertama.
- **.set_title('Grayscale Image') :** Mengatur judul untuk subplot pertama menjadi 'Grayscale Image'.

ax[1].imshow(edges, cmap='gray') :

- **ax[1] :** Mengakses subplot kedua.
- **.imshow(edges, cmap='gray') :** Menampilkan edges (gambar hasil deteksi tepi Canny) di subplot kedua. cmap='gray' juga digunakan di sini karena output Canny adalah gambar biner yang pada dasarnya adalah grayscale (hitam dan putih).

ax[1].set_title('Canny Edges') :

- **ax[1] :** Mengakses subplot kedua.
- **.set_title('Canny Edges') :** Mengatur judul untuk subplot kedua menjadi 'Canny Edges'.

Penjelasan Output :

"Grayscale Image" : Merupakan tampilan dari img_gray. Yang dimana merupakan representasi gambar lahan parkir dalam nuansa abu-abu. Subplot ini berfungsi sebagai referensi untuk membandingkan hasil deteksi tepi.

"Canny Edges" : Merupakan tampilan dari edges, yaitu hasil dari algoritma deteksi tepi Canny. Terlihat bahwa sebagian besar objek di lahan parkir (mobil, garis parkir, pembatas) sekarang direpresentasikan hanya dengan tepiannya yang berwarna putih di atas latar belakang hitam.

Algoritma Canny telah berhasil mengidentifikasi batas-batas yang signifikan dalam gambar, seperti kontur mobil dan garis-garis pembatas parkir. Ambang batas (100, 150) yang diberikan ke fungsi Canny telah membantu menentukan seberapa "kuat" suatu gradien harus dianggap sebagai tepi. Tepi-tepi yang halus atau noise biasanya diabaikan oleh algoritma Canny, menghasilkan hasil yang bersih dan jelas.

```
lines = cv2.HoughLinesP(edges, 1, np.pi / 180, threshold=100, minLineLength=100, maxLineGap=10)

img_lines = image.copy()


for line in lines:
```

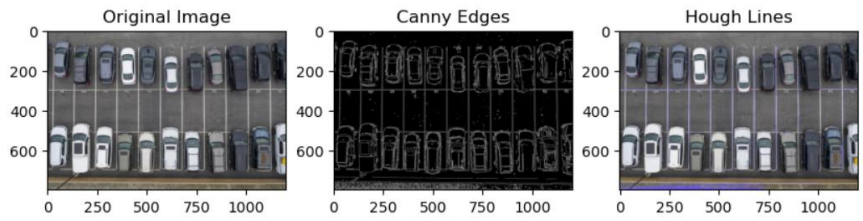
```
    x1, y1, x2, y2 = line[0]
    cv2.line(img_lines, (x1, y1), (x2, y2), (100, 80, 255), 1)


fig, axs = plt.subplots(1, 3, figsize=(10, 10))
ax = axs.ravel()
ax[0].imshow(image)
ax[0].set_title('Original Image')
ax[1].imshow(edges, cmap='gray')
ax[1].set_title('Canny Edges')
ax[2].imshow(img_lines)
ax[2].set_title('Hough Lines')
Text(0.5, 1.0, 'Hough Lines')
```

```
[14]: lines = cv2.HoughLinesP(edges, 1, np.pi / 180, threshold=100, minLineLength=100, maxLineGap=10)
      img_lines = image.copy()

      for line in lines:
          x1, y1, x2, y2 = line[0]
          cv2.line(img_lines, (x1, y1), (x2, y2), (100, 80, 255), 1)
```

202331084_Felix Ewaldo Panggabean

```
[16]: fig, axs = plt.subplots(1, 3, figsize=(10, 10))
      ax = axs.ravel()
      ax[0].imshow(image)
      ax[0].set_title('Original Image')
      ax[1].imshow(edges, cmap='gray')
      ax[1].set_title('Canny Edges')
      ax[2].imshow(img_lines)
      ax[2].set_title('Hough Lines')
```

```
[16]: Text(0.5, 1.0, 'Hough Lines')
```



lines = cv2.HoughLinesP(edges, 1, np.pi / 180, 100, minLineLength=100, maxLineGap=10) :

- **cv2.HoughLinesP() :** Ini adalah fungsi dari OpenCV yang mengimplementasikan Probabilistic Hough Line Transform. Ini adalah metode yang lebih efisien dan seringkali lebih praktis daripada Hough Line Transform standar karena tidak mengembalikan semua titik yang membentuk garis, tetapi hanya titik awal dan akhir garis yang terdeteksi. Ini bekerja dengan baik untuk mendeteksi garis lurus dalam gambar.
- **Edges :** Ini adalah input gambar yang akan dicari garisnya. Fungsi Hough Lines bekerja paling baik pada gambar biner (hitam dan putih) di mana fitur yang menarik

(dalam hal ini, tepi) ditandai dengan warna putih. Gambar edges yang kita dapatkan dari deteksi Canny sangat cocok untuk ini.

- **1:** rho resolution (resolusi jarak) dalam piksel. Ini adalah akurasi yang digunakan untuk parameter r dalam ruang Hough.
- **np.pi / 180 :** theta resolution (resolusi sudut) dalam radian. Ini adalah akurasi yang digunakan untuk parameter θ dalam ruang Hough. np.pi / 180 setara dengan 1 derajat.
- **100 :** threshold. Ini adalah ambang batas akumulator (votes). Hanya garis dengan jumlah "suara" yang lebih tinggi dari ambang batas ini yang akan dikembalikan. Semakin tinggi nilainya, semakin sedikit garis yang terdeteksi, tetapi garis yang terdeteksi akan lebih kuat.
- **minLineLength=100 :** Ini adalah argumen opsional yang menentukan panjang minimum garis yang dapat diterima. Garis yang lebih pendek dari 100 piksel akan diabaikan.
- **maxLineGap=10 :** Ini adalah argumen opsional yang menentukan jarak maksimum antara dua segmen garis yang berdekatan yang akan dianggap sebagai satu garis tunggal. Jika ada celah yang lebih besar dari 10 piksel, dua segmen akan dianggap sebagai dua garis terpisah.
- **lines = ... :** Fungsi ini mengembalikan array NumPy dari garis-garis yang terdeteksi. Setiap baris direpresentasikan sebagai [x1, y1, x2, y2], di mana (x1, y1) adalah titik awal garis dan (x2, y2) adalah titik akhir garis.

img_lines = image.copy() :

- **image.copy() :** Membuat salinan dari gambar asli (image). Penting untuk membuat salinan agar gambar asli tidak dimodifikasi secara langsung ketika menggambar garis.
- **img_lines = ... :** Salinan gambar disimpan ke dalam variabel img_lines.

for line in lines: : Ini memulai sebuah loop yang akan mengiterasi melalui setiap garis yang terdeteksi oleh cv2.HoughLinesP().

x1, y1, x2, y2 = line[0] :

- **line[0] :** Setiap elemen dalam lines adalah array tunggal yang berisi koordinat garis (misalnya [[x1, y1, x2, y2]]). Oleh karena itu, perlu mengakses elemen pertama ([0]) untuk mendapatkan array koordinat itu sendiri.
- **x1, y1, x2, y2 = ... :** Melakukan unpacking dari array koordinat menjadi empat variabel terpisah : x1, y1 (koordinat titik awal), dan x2, y2 (koordinat titik akhir).

cv2.line(img_lines, (x1, y1), (x2, y2), (0, 0, 255), 1) :

- **cv2.line() :** Ini adalah fungsi dari OpenCV yang digunakan untuk menggambar garis pada gambar.
- **img_lines :** Gambar di mana garis akan digambar.
- **(x1, y1) :** Koordinat titik awal garis.
- **(x2, y2) :** Koordinat titik akhir garis.

- **(0, 0, 255) :** Warna garis dalam format BGR (Biru, Hijau, Merah). Di sini, (0, 0, 255) berarti warna merah murni.
- **1 :** Ketebalan garis dalam piksel.

fig, axs = plt.subplots(1, 3, figsize=(10, 10)) : Baris ini sama seperti sebelumnya, tetapi kali ini membuat 3 subplot (1 baris, 3 kolom) karena ingin menampilkan tiga gambar.

ax = axs.ravel() : Sama seperti sebelumnya, meratakan array Axes.

ax[0].imshow(image) : Menampilkan gambar asli (image) di subplot pertama. Perhatikan bahwa di sini menggunakan image asli, bukan img_rgb. Ini bisa menyebabkan sedikit perbedaan warna jika gambar asli adalah BGR dan Matplotlib menampilkan sebagai RGB tanpa konversi. Namun, untuk gambar ini, tampaknya bekerja cukup baik.

ax[0].set_title('Original Image') : Mengatur judul untuk subplot pertama.

ax[1].imshow(edges, cmap='gray') : Menampilkan gambar hasil deteksi Canny (edges) di subplot kedua, dengan colormap grayscale.

ax[1].set_title('Canny Edges') : Mengatur judul untuk subplot kedua.

ax[2].imshow(img_lines) : Menampilkan gambar img_lines (gambar asli dengan garis-garis yang terdeteksi digambar di atasnya) di subplot ketiga.

ax[2].set_title('Hough Lines') : Mengatur judul untuk subplot ketiga.

Penjelasan Output :

Gambar output menunjukkan tiga subplot yang berdampingan :

"Original Image" : Merupakan tampilan dari gambar asli lahan parkir. Ini berfungsi sebagai referensi visual.

"Canny Edges" : Menampilkan tampilan dari hasil deteksi tepi Canny. Melihat garis-garis putih yang menunjukkan tepi-tepi objek seperti mobil dan garis parkir. Ini adalah input untuk Probabilistic Hough Line Transform.

"Hough Lines" : Tampilan dari img_lines, yaitu gambar asli dengan garis-garis merah yang digambar di atasnya. Garis-garis merah ini adalah hasil deteksi garis lurus oleh Probabilistic Hough Line Transform. Banyak garis lurus telah berhasil dideteksi, terutama garis-garis parkir yang panjang, bagian atas dan samping mobil, serta garis-garis batas lainnya. Parameter minLineLength=100 telah memastikan bahwa hanya garis yang cukup panjang yang dideteksi, dan maxLineGap=10 membantu menggabungkan segmen-segmen garis yang berdekatan menjadi satu garis utuh. Hasilnya cukup efektif dalam mengidentifikasi struktur garis lurus di gambar.

2. Olah gambar yang sama dengan gambar laporan 1 dengan metode geometrik yang sudah dipratikumkan gunakanlah metode resize (2 cara) dan rotated (2 cara) !.

Operasi Geomatrik

baca gambar

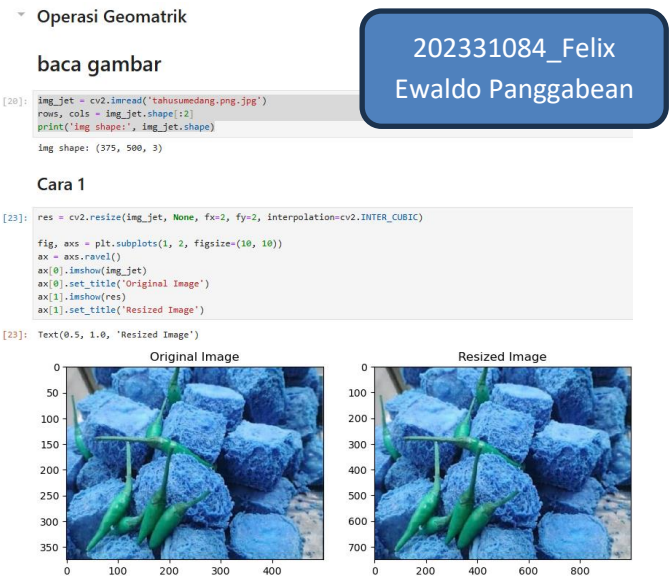
```
img_jet = cv2.imread('tahusumedang.png.jpg')
rows, cols = img_jet.shape[:2]
print('img shape:', img_jet.shape)
```

```
img shape: (375, 500, 3)
```

Cara 1

```
res = cv2.resize(img_jet, None, fx=2, fy=2, interpolation=cv2.INTER_CUBIC)


fig, axs = plt.subplots(1, 2, figsize=(10, 10))
ax = axs.ravel()
ax[0].imshow(img_jet)
ax[0].set_title('Original Image')
ax[1].imshow(res)
ax[1].set_title('Resized Image')
```

```
Text(0.5, 1.0, 'Resized Image')
```



Penjelasan :

img_jet = cv2.imread('tahusumedang.png') :

- **cv2.imread() :** Fungsi dari OpenCV untuk membaca gambar dari file.
- **'tahusumedang.png' :** Nama file gambar yang akan dibaca. Pastikan file ini ada di lokasi yang dapat diakses oleh script.
- **img_jet = ... :** Gambar yang dibaca akan disimpan dalam variabel img_jet.

rows, cols = img_jet.shape[:2] :

- **img_jet.shape :** Mengembalikan tuple yang berisi dimensi gambar (tinggi, lebar, jumlah_channel). Contoh : (375, 500, 3).
- **[:2] :** Melakukan slicing untuk mengambil hanya dua elemen pertama dari tuple shape, yaitu tinggi (rows) dan lebar (cols).
- **rows, cols = ... :** Nilai tinggi disimpan di variabel rows dan nilai lebar disimpan di variabel cols.

print('img shape:', img_jet.shape) :

- **print() :** Mencetak teks ke konsol.
- **'img shape:' :** String literal yang akan dicetak.
- **img_jet.shape :** Mencetak nilai shape dari gambar img_jet.

Output dari "baca gambar" :

img shape: (375, 500, 3) : Output ini mengkonfirmasi dimensi gambar tahusumedang.png.

375 : Tinggi gambar dalam piksel.

500 : Lebar gambar dalam piksel.

3 : Jumlah channel warna (biasanya BGR untuk gambar berwarna di OpenCV).

Bagian "Cara 1" (Resizing Gambar) :

res = cv2.resize(img_jet, None, fx=2, fy=2, interpolation=cv2.INTER_CUBIC) :

- **cv2.resize() :** Fungsi dari OpenCV yang digunakan untuk mengubah ukuran (resize) gambar.
- **img_jet :** Gambar input yang ingin diubah ukurannya.
- **None :** Argumen dsize (ukuran output) diatur ke None karena akan menggunakan faktor skala (fx dan fy) sebagai gantinya.
- **fx=2 :** Faktor skala untuk lebar. Gambar akan diperbesar 2 kali lebarnya.
- **fy=2 :** Faktor skala untuk tinggi. Gambar akan diperbesar 2 kali tingginya.
- **interpolation=cv2.INTER_CUBIC :** Ini adalah metode interpolasi yang digunakan saat mengubah ukuran gambar.
- **cv2.INTER_CUBIC adalah** interpolasi bicubic, yang menggunakan 4x4 piksel tetangga untuk perhitungan interpolasi. Ini umumnya memberikan hasil yang lebih

baik (lebih halus) untuk pembesaran dibandingkan metode lain seperti cv2.INTER_LINEAR, meskipun sedikit lebih lambat.

- **res = ... :** Gambar hasil resizing disimpan dalam variabel res.

fig, axs = plt.subplots(1, 2, figsize=(10, 10)) : Baris ini membuat figur Matplotlib dengan 1 baris dan 2 kolom subplot, serta ukuran figur 10x10 inci. Ini akan memungkinkan untuk menampilkan dua gambar secara berdampingan.

ax = axs.ravel() : Meratakan array axs (yang berisi objek Axes untuk setiap subplot) menjadi array 1D untuk kemudahan akses.

ax[0].imshow(img_jet) :

- **ax[0] :** Mengakses subplot pertama.
- **.imshow(img_jet) :** Menampilkan gambar asli (img_jet) di subplot pertama.

ax[0].set_title('Original Image') : Mengatur judul untuk subplot pertama menjadi 'Original Image'.

ax[1].imshow(res) :

- **ax[1] :** Mengakses subplot kedua.
- **.imshow(res) :** Menampilkan gambar hasil resizing (res) di subplot kedua.

ax[1].set_title('Resized Image') : Mengatur judul untuk subplot kedua menjadi 'Resized Image'.

Penjelasan Output :

Gambar output menunjukkan dua subplot yang berdampingan :

"Original Image" : Merupakan tampilan dari img_jet, yaitu gambar asli tahu sumedang dan cabai. Terlihat ukuran aslinya dengan dimensi sekitar 375 piksel tinggi dan 500 piksel lebar.

"Resized Image" : Merupakan tampilan dari res, yaitu gambar yang telah diubah ukurannya. Karena fx=2 dan fy=2, gambar ini telah diperbesar dua kali lipat baik secara horizontal maupun vertikal. Lebar gambar sekarang menjadi 500×2=1000 piksel, dan tingginya menjadi 375×2=750 piksel. Meskipun gambar diperbesar, interpolasi cv2.INTER_CUBIC telah membantu menjaga kualitas visual. Terlihat sedikit kehalusan dibandingkan jika menggunakan metode interpolasi yang lebih sederhana, yang bisa menghasilkan pikselasi yang lebih jelas saat pembesaran.

Cara 2

```
tinggi, lebar = img_jet.shape[:2]

print('tinggi:', tinggi, 'lebar:', lebar)
```

res2 = cv2.resize(img_jet, (4*tinggi, 4*lebar), interpolation=cv2.INTER_CUBIC)

fig, axs = plt.subplots(1, 2, figsize=(10, 10))

ax = axs.ravel()

ax[0].imshow(img_jet)

ax[0].set_title('Original Image')

ax[1].imshow(res2)

ax[1].set_title('Resized Image (4x)')

tinggi: 375 lebar: 500

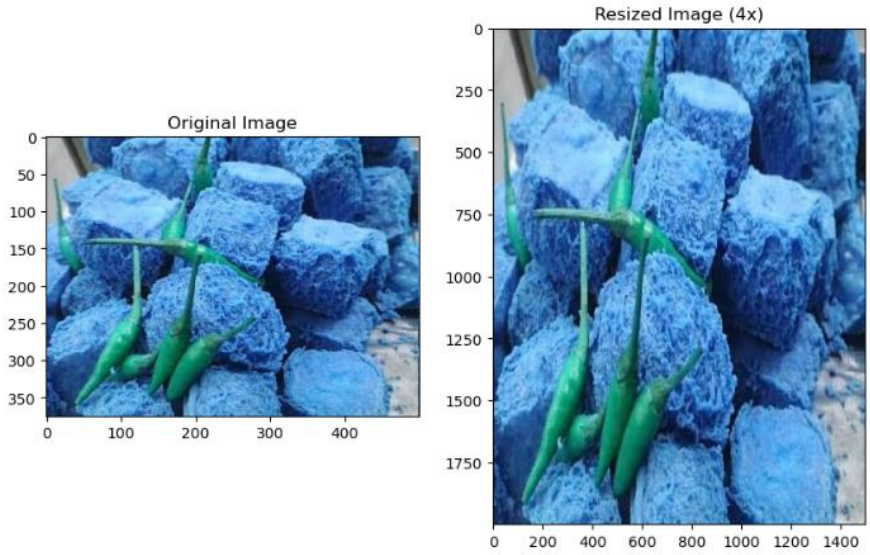
Text(0.5, 1.0, 'Resized Image (4x)')

Cara 2

```
[26]: tinggi, lebar = img_jet.shape[:2]
      print('tinggi:', tinggi, 'lebar:', lebar)
      res2 = cv2.resize(img_jet, (4*tinggi, 4*lebar), interpolation=cv2.INTER_CUBIC)
      fig, axs = plt.subplots(1, 2, figsize=(10, 10))
      ax = axs.ravel()
      ax[0].imshow(img_jet)
      ax[0].set_title('Original Image')
      ax[1].imshow(res2)
      ax[1].set_title('Resized Image (4x)')
```

202331084_Felix Ewaldo Panggabean

```
tinggi: 375 lebar: 500
[26]: Text(0.5, 1.0, 'Resized Image (4x)')
```



Penjelasan :

tinggi, lebar = img_jet.shape[:2] : Mendapatkan tinggi dan lebar gambar img_jet (gambar tahu sumedang asli) dan menyimpannya ke variabel tinggi dan lebar.

print('tinggi:', tinggi, 'lebar:', lebar) : Mencetak nilai tinggi dan lebar gambar asli ke konsol.

Output dari print() : tinggi: 375 lebar: 500 : Ini mengkonfirmasi dimensi gambar asli.

res2 = cv2.resize(img_jet, (4*lebar, 4*tinggi), interpolation=cv2.INTER_CUBIC) :

- **cv2.resize() :** Fungsi untuk mengubah ukuran gambar.
- **img_jet :** Gambar input (gambar tahu sumedang asli).
- **(4*lebar, 4*tinggi) :** Ini adalah argumen dsize (ukuran output) yang ditentukan secara eksplisit dalam format (lebar_baru, tinggi_baru).
- **4*lebar :** Lebar baru akan menjadi 4 kali lebar asli.
- **4*tinggi :** Tinggi baru akan menjadi 4 kali tinggi asli.
- **interpolation=cv2.INTER_CUBIC :** Menggunakan metode interpolasi bicubic, sama seperti sebelumnya, untuk menghasilkan gambar yang diperbesar dengan kualitas yang baik.
- **res2 = ... :** Gambar hasil resizing disimpan dalam variabel res2.

fig, axs = plt.subplots(1, 2, figsize=(10, 10)) : Baris ini membuat figur Matplotlib dengan 1 baris dan 2 kolom subplot, serta ukuran figur 10x10 inci.

ax = axs.ravel() : Meratakan array axs menjadi array 1D untuk akses mudah ke subplot.

ax[0].imshow(img_jet) :

- **ax[0] :** Mengakses subplot pertama.
- **.imshow(img_jet) :** Menampilkan gambar asli (img_jet) di subplot pertama.

ax[0].set_title('Original Image') : Mengatur judul untuk subplot pertama menjadi 'Original Image'.

ax[1].imshow(res2) :

- **ax[1] :** Mengakses subplot kedua.
- **.imshow(res2) :** Menampilkan gambar hasil resizing (res2) di subplot kedua.

ax[1].set_title('Resized Image (4x)') : Mengatur judul untuk subplot kedua menjadi 'Resized Image (4x)'. Penambahan "(4x)" pada judul ini secara eksplisit menunjukkan bahwa gambar telah diperbesar 4 kali lipat.

Penjelasan Output :

Gambar output menunjukkan dua subplot yang berdampingan :

"Original Image" : Merupakan tampilan dari img_jet, yaitu gambar asli tahu sumedang dan cabai. Dimensinya seperti yang dicetak sebelumnya : Tinggi 375 piksel, Lebar 500 piksel.

"Resized Image (4x)" :

Ini adalah tampilan dari res2, yaitu gambar yang telah diubah ukurannya dengan faktor 4x.

Lebar gambar sekarang menjadi 500×4=2000 piksel.

Tinggi gambar sekarang menjadi 375×4=1500 piksel.

Meskipun resolusinya menjadi jauh lebih tinggi, mungkin masih bisa melihat sedikit kelembutan atau kurangnya ketajaman pada detail yang sangat halus dibandingkan gambar aslinya jika diperbesar lebih jauh. Namun, untuk pembesaran 4x ini, metode interpolasi cv2.INTER_CUBIC berhasil mempertahankan kualitas gambar dengan cukup baik, menghasilkan gambar yang terlihat mulus meskipun ukurannya jauh lebih besar. Secara keseluruhan, "Cara 2" ini mendemonstrasikan metode lain untuk mengubah ukuran gambar di OpenCV, yaitu dengan secara eksplisit menentukan dimensi output yang diinginkan, yang dalam kasus ini adalah 4 kali lipat dari dimensi asli.\

Perputaran citra / rotasi

```
img_tahusumedang = cv2.imread('tahusumedang.png.jpg')

rows, cols = img_tahusumedang.shape[:2]

print('img shape:', img_tahusumedang.shape)
```

img shape: (375, 500, 3)

Cara 1

```
M = cv2.getRotationMatrix2D(((cols-1)/2, (rows-1)/2), 90, 1)

img_putar = cv2.warpAffine(img_tahusumedang, M, (cols, rows))


fig, axs = plt.subplots(1, 2, figsize=(10, 10))

ax = axs.ravel()

ax[0].imshow(img_tahusumedang)

ax[0].set_title('Original Image')

ax[1].imshow(img_putar)

ax[1].set_title('Rotated Image')


for a in axs:

    a.axis('off')

plt.tight_layout()

plt.show()
```

Perputaran citra / rotasi

202331084_Felix Ewaldo Panggabean

```
[29]: img_tahusumedang = cv2.imread('tahusumedang.png.jpg')
      rows, cols = img_tahusumedang.shape[:2]
      print('img shape:', img_tahusumedang.shape)
```

```
img shape: (375, 500, 3)
```

Cara 1

```
[32]: M = cv2.getRotationMatrix2D(((cols-1)/2, (rows-1)/2), 90, 1)
      img_putar = cv2.warpAffine(img_tahusumedang, M, (cols, rows))

      fig, axs = plt.subplots(1, 2, figsize=(10, 10))
      ax = axs.ravel()
      ax[0].imshow(img_tahusumedang)
      ax[0].set_title('Original Image')
      ax[1].imshow(img_putar)
      ax[1].set_title('Rotated Image')

      for a in axs:
          a.axis('off')
      plt.tight_layout()
      plt.show()
```

Penjelasan :
Bagian "Perputaran citra / rotasi" :

img_tahusumedang = cv2.imread('tahusumedang.png') :

- **cv2.imread() :** Fungsi OpenCV untuk membaca gambar dari file.
- **'tahusumedang.png' :** Nama file gambar yang akan dibaca. Ini adalah gambar yang sama dengan yang digunakan sebelumnya.
- **img_tahusumedang = ... :** Gambar yang dibaca akan disimpan dalam variabel img_tahusumedang.

rows, cols = img_tahusumedang.shape[:2] : Mendapatkan tinggi (rows) dan lebar (cols) dari gambar img_tahusumedang.

print('img shape:', img_tahusumedang.shape) : Mencetak dimensi gambar yang baru saja dibaca.

Output dari print() :

img shape: (375, 500, 3) : Konfirmasi dimensi gambar asli : tinggi 375 piksel, lebar 500 piksel, dan 3 saluran warna.

Bagian "Cara 1" (Rotasi Gambar) :

M = cv2.getRotationMatrix2D((cols-1)/2, (rows-1)/2, 90, 1) :

- **cv2.getRotationMatrix2D() :** Fungsi dari OpenCV yang digunakan untuk mendapatkan matriks transformasi rotasi 2D. Matriks ini kemudian akan digunakan oleh fungsi cv2.warpAffine untuk melakukan rotasi.
- **((cols-1)/2, (rows-1)/2) :** Ini adalah titik pusat rotasi. Mengurangi 1 karena indeks piksel dimulai dari 0.
- **90 :** Sudut rotasi dalam derajat. Di sini, gambar akan diputar 90 derajat searah jarum jam (positif berarti berlawanan arah jarum jam di beberapa sistem koordinat, tetapi di OpenCV biasanya searah jarum jam).
- **1 :** Faktor skala. 1 berarti tidak ada perubahan skala (gambar tidak diperbesar atau diperkecil selama rotasi).
- **M = ... :** Matriks transformasi 2x3 yang dihasilkan disimpan dalam variabel M.

img_putar = cv2.warpAffine(img_tahusumedang, M, (cols, rows)) :

- **cv2.warpAffine() :** Fungsi dari OpenCV yang melakukan transformasi affine (termasuk rotasi, translasi, scaling, shearing) pada gambar menggunakan matriks transformasi yang diberikan.
- **img_tahusumedang :** Gambar input yang akan dirotasi.
- **M :** Matriks transformasi rotasi yang didapatkan dari cv2.getRotationMatrix2D().
- **(cols, rows) :** Ukuran output gambar (lebar, tinggi). Penting untuk dicatat bahwa jika memutar gambar 90 derajat, dimensi output mungkin perlu disesuaikan (misalnya, lebar asli menjadi tinggi baru, dan tinggi asli menjadi lebar baru) agar seluruh gambar muat dan tidak terpotong. Namun, dalam kasus ini, digunakan dimensi asli (cols, rows) sehingga bagian gambar mungkin terpotong jika gambar tidak berbentuk persegi.
- **img_putar = ... :** Gambar hasil rotasi disimpan dalam variabel img_putar.

fig, axs = plt.subplots(1, 2, figsize=(10, 10)) : Membuat figur Matplotlib dengan 1 baris dan 2 kolom subplot, ukuran figur 10x10 inci.

ax = axs.ravel() : Meratakan array axs menjadi array 1D.

ax[0].imshow(img_tahusumedang) : Menampilkan gambar asli (img_tahusumedang) di subplot pertama.

ax[0].set_title('Original Image') : Mengatur judul untuk subplot pertama.

ax[1].imshow(img_putar) : Menampilkan gambar hasil rotasi (img_putar) di subplot kedua.

ax[1].set_title('Rotated Image') : Mengatur judul untuk subplot kedua.

for a in axs: : Memulai loop untuk mengiterasi setiap objek Axes (subplot) dalam array axs.

a.axis('off') : Untuk setiap subplot, ini mematikan label sumbu x dan y, serta tanda centang (ticks), membuat tampilan gambar lebih bersih.

plt.tight_layout() : Secara otomatis menyesuaikan parameter subplot untuk memberikan tata letak yang rapi dan padat, menghindari tumpang tindih label atau judul.

plt.show(): : Menampilkan figur Matplotlib.

Penjelasan Output :

Gambar output menunjukkan dua subplot yang berdampingan :

"Original Image" : Merupakan tampilan dari img_tahusumedang, yaitu gambar asli tahu sumedang dan cabai.

"Rotated Image" : Merupakan tampilan dari img_putar, yaitu gambar yang telah diputar 90 derajat searah jarum jam. Terlihat bahwa gambar telah berputar, dengan cabai sekarang mengarah ke kanan bawah. Karena menggunakan dimensi output (cols, rows) yang sama dengan dimensi gambar asli ((500, 375)), dan gambar asli tidak berbentuk persegi, sebagian dari gambar yang diputar telah terpotong. Misalnya, bagian atas dan bawah dari gambar yang diputar (yang sebenarnya adalah sisi kiri dan kanan dari gambar asli) tidak terlihat karena berada di luar batas (500, 375) yang ditentukan untuk output. Jika ingin melihat seluruh gambar yang diputar tanpa terpotong, ukuran output cv2.warpAffine harus disesuaikan menjadi (rows, cols) atau dimensi yang lebih besar yang dapat menampung gambar yang diputar.

Cara 2

```
from skimage import io, transform

img_tahusumedang2 = io.imread('tahusumedang.png.jpg')

rotated = transform.rotate(img_tahusumedang2, 45, resize=False)

rotated2 = transform.rotate(img_tahusumedang2, 45, resize=True)


fig, axs = plt.subplots(1, 3, figsize=(10, 10))

ax = axs.ravel()

ax[0].imshow(img_tahusumedang)

ax[0].set_title('Original Image')

ax[1].imshow(rotated)

ax[1].set_title('Rotated Image (No Resize)')

ax[2].imshow(rotated2)
```

```
ax[2].set_title('Rotated Image (With Resize)')

for a in axs:

    a.axis('off')

plt.tight_layout()

plt.show()
```

Cara 2

```
[35]: from skimage import io, transform
img_tahusumedang2 = io.imread('tahusumedang.png.jpg')
rotated = transform.rotate(img_tahusumedang2, 45, resize=False)
rotated2 = transform.rotate(img_tahusumedang2, 45, resize=True)

fig, axs = plt.subplots(1, 3, figsize=(10, 10))
ax = axs.ravel()
ax[0].imshow(img_tahusumedang)
ax[0].set_title('Original Image')
ax[1].imshow(rotated)
ax[1].set_title('Rotated Image (No Resize)')
ax[2].imshow(rotated2)
ax[2].set_title('Rotated Image (With Resize)')
for a in axs:
    a.axis('off')
plt.tight_layout()
plt.show()
```

202331084_Felix Ewaldo Panggabean

Penjelasan :

"Cara 2" (Rotasi Gambar Menggunakan scikit-image) :

from skimage import io, transform :

- **from skimage import ... :** Mengimpor modul atau fungsi tertentu dari library scikit-image (skimage).
- **Io :** Modul untuk operasi input/output gambar (membaca, menulis).
- **Transform :** Modul yang berisi berbagai fungsi transformasi geometris, termasuk rotasi.

img_tahusumedang2 = io.imread('tahusumedang.png') :

- **io.imread() :** Fungsi dari skimage.io untuk membaca gambar dari file. Ini mirip dengan cv2.imread().
- **'tahusumedang.png' :** Nama file gambar yang akan dibaca.

- **img_tahusumedang2 = ... :** Gambar yang dibaca disimpan dalam variabel img_tahusumedang2. Perhatikan bahwa Matplotlib dan scikit-image biasanya bekerja dengan gambar dalam format RGB, sedangkan OpenCV menggunakan BGR. io.imread biasanya membaca dalam RGB.

rotated = transform.rotate(img_tahusumedang2, 45, resize=False) :

- **transform.rotate() :** Fungsi dari skimage.transform untuk melakukan rotasi pada gambar.
- **img_tahusumedang2 :** Gambar input yang akan dirotasi.
- **45 :** Sudut rotasi dalam derajat. Dalam scikit-image, sudut positif berarti rotasi berlawanan arah jarum jam.
- **resize=False :** Argumen ini menentukan apakah ukuran output gambar harus disesuaikan agar seluruh gambar yang dirotasi dapat dimuat tanpa terpotong. Dengan False, ukuran gambar output akan tetap sama dengan gambar input, sehingga bagian gambar yang diputar dan keluar dari batas akan terpotong.
- **rotated = ... :** Gambar hasil rotasi disimpan dalam variabel rotated.

rotated2 = transform.rotate(img_tahusumedang2, 45, resize=True) :

- **transform.rotate() :** Fungsi rotasi yang sama.
- **img_tahusumedang2 :** Gambar input.
- **45 :** Sudut rotasi 45 derajat berlawanan arah jarum jam.
- **resize=True :** Argumen ini menentukan bahwa ukuran output gambar harus disesuaikan secara otomatis agar seluruh gambar yang dirotasi dapat dimuat tanpa terpotong. Ini akan membuat kanvas output lebih besar jika diperlukan.
- **rotated2 = ... :** Gambar hasil rotasi dengan penyesuaian ukuran disimpan dalam variabel rotated2.

fig, axs = plt.subplots(1, 3, figsize=(10, 10)) : Membuat figur Matplotlib dengan 1 baris dan 3 kolom subplot, serta ukuran figur 10x10 inci. Ini untuk menampilkan tiga gambar.

ax = axs.ravel() : Meratakan array axs menjadi array 1D untuk akses mudah ke subplot.

ax[0].imshow(img_tahusumedang) : Menampilkan gambar asli (img_tahusumedang) di subplot pertama. Catatan: Ada kemungkinan img_tahusumedang ini adalah gambar yang dibaca sebelumnya dengan OpenCV (BGR), sedangkan rotated dan rotated2 dibaca dengan skimage.io (RGB). Matplotlib umumnya menampilkan RGB dengan benar, jadi mungkin tidak ada masalah visual yang signifikan di sini.

ax[0].set_title('Original Image') : Mengatur judul untuk subplot pertama.

ax[1].imshow(rotated) : Menampilkan gambar hasil rotasi tanpa penyesuaian ukuran (rotated) di subplot kedua.

ax[1].set_title('Rotated Image (No Resize)') : Mengatur judul untuk subplot kedua.

ax[2].imshow(rotated2) : Menampilkan gambar hasil rotasi dengan penyesuaian ukuran (rotated2) di subplot ketiga.

ax[2].set_title('Rotated Image (With Resize)') : Mengatur judul untuk subplot ketiga.

for a in axs: : Memulai loop untuk setiap subplot.

a.axis('off') : Mematikan label sumbu x dan y serta tanda centang pada setiap subplot untuk tampilan yang bersih.

plt.tight_layout() : Menyesuaikan tata letak subplot agar rapi.

plt.show() : Menampilkan figur.

Penjelasan Output :

Gambar output menunjukkan tiga subplot yang berdampingan :

"Original Image" : Merupakan tampilan dari gambar asli tahu sumedang dan cabai. Ini adalah referensi untuk perbandingan.

"Rotated Image (No Resize)" : Merupakan tampilan dari rotated, yaitu gambar asli yang telah diputar 45 derajat berlawanan arah jarum jam. Terlihat bahwa gambar telah berputar, tetapi bagian-bagian gambar yang meluas keluar dari batas persegi panjang asli telah **terpotong**. Kanvas outputnya memiliki dimensi yang sama persis dengan gambar asli.

"Rotated Image (With Resize)" : Merupakan tampilan dari rotated2, yaitu gambar asli yang telah diputar 45 derajat berlawanan arah jarum jam, tetapi dengan resize=True. Di sini, gambar telah diputar, dan ukuran kanvas output telah secara otomatis diperbesar (ditambah area hitam di sekitarnya) agar seluruh gambar yang diputar dapat dimuat tanpa terpotong. Ini memastikan tidak ada kehilangan informasi visual akibat rotasi. Terlihat seluruh objek dalam gambar, termasuk bagian-bagian yang akan terpotong jika resize=False.

3. Olah gambar yang sudah diberikan dan terapkan pengaplikasiannya untuk mengambil plat DD 4419 KF dari gambar plat.jpg.

Pengaplikasian

```
t = cv2.imread('plat.jpg')

t_gray = cv2.cvtColor(t, cv2.COLOR_BGR2GRAY)

fig, axs = plt.subplots(1, 2, figsize=(10, 10))

ax = axs.ravel()

ax[0].imshow(t)

ax[0].set_title('Original Image')
```

ax[1].imshow(t_gray, cmap='gray')

ax[1].set_title('Grayscale Image')

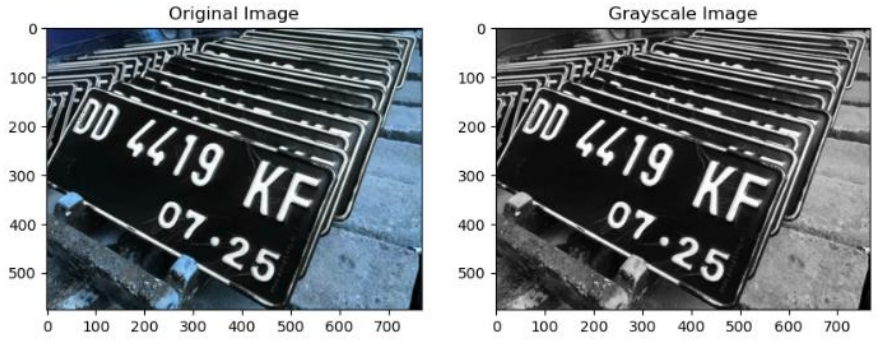
Text(0.5, 1.0, 'Grayscale Image')

Pengaplikasian ¶

```
[38]: t = cv2.imread('plat.jpg')
      t_gray = cv2.cvtColor(t, cv2.COLOR_BGR2GRAY)
      fig, axs = plt.subplots(1, 2, figsize=(10, 10))
      ax = axs.ravel()
      ax[0].imshow(t)
      ax[0].set_title('Original Image')
      ax[1].imshow(t_gray, cmap='gray')
      ax[1].set_title('Grayscale Image')
```

202331084_Felix Ewaldo Panggabean

```
[38]: Text(0.5, 1.0, 'Grayscale Image')
```



Penjelasan :

Bagian "Pengaplikasian 1" :

t = cv2.imread('plat.jpg') :

- **cv2.imread() :** Fungsi dari OpenCV untuk membaca gambar dari file.
- **'plat.jpg' :** Nama file gambar yang akan dibaca. Berdasarkan output, ini adalah gambar plat nomor kendaraan.
- **t = ... :** Gambar yang dibaca akan disimpan dalam variabel t.

t_gray = cv2.cvtColor(t, cv2.COLOR_BGR2GRAY) :

- **cv2.cvtColor() :** Fungsi dari OpenCV untuk mengubah ruang warna suatu gambar.
- **t :** Gambar input (gambar plat nomor) yang kemungkinan besar dalam format BGR (default OpenCV).
- **cv2.COLOR_BGR2GRAY :** Kode konversi yang memberitahu cvtColor untuk mengubah gambar dari BGR menjadi gambar skala abu-abu (grayscale).
- **t_gray = ... :** Gambar hasil konversi ke grayscale disimpan ke dalam variabel t_gray.

fig, axs = plt.subplots(1, 2, figsize=(10, 10)) : Membuat figur Matplotlib dengan 1 baris dan 2 kolom subplot, serta ukuran figur 10x10 inci. Ini akan digunakan untuk menampilkan dua gambar secara berdampingan.

ax = axs.ravel() : Meratakan array axs (yang berisi objek Axes untuk setiap subplot) menjadi array 1D untuk kemudahan akses.

ax[0].imshow(t) :

- **ax[0] :** Mengakses subplot pertama.
- **.imshow(t) :** Menampilkan gambar asli (t) di subplot pertama. Karena gambar asli dibaca oleh OpenCV dalam BGR dan Matplotlib cenderung menampilkan RGB, kadang-kadang warna bisa sedikit menyimpang (misalnya, biru jadi merah, merah jadi biru), tetapi untuk gambar ini efeknya mungkin tidak terlalu signifikan jika warnanya didominasi hitam putih.

ax[0].set_title('Original Image') : Mengatur judul untuk subplot pertama menjadi 'Original Image'.

ax[1].imshow(t_gray, cmap='gray') :

- **ax[1] :** Mengakses subplot kedua.
- **.imshow(t_gray, cmap='gray') :** Menampilkan gambar grayscale (t_gray) di subplot kedua.
- **cmap='gray' :** Argumen ini penting untuk memastikan bahwa gambar grayscale ditampilkan dengan benar dalam nuansa abu-abu, bukan dengan colormap default Matplotlib yang mungkin berwarna-warni.

ax[1].set_title('Grayscale Image') : Mengatur judul untuk subplot kedua menjadi 'Grayscale Image'.

Penjelasan Output :

Gambar output menunjukkan dua subplot yang berdampingan :

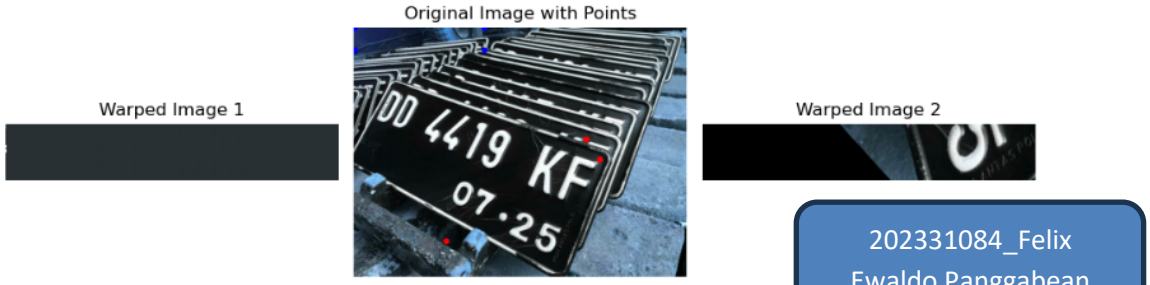
"Original Image" : Merupakan tampilan dari gambar asli plat.jpg. Terlihat plat nomor kendaraan dengan latar belakang hitam dan tulisan putih. Warnanya terlihat seperti gambar aslinya (meskipun mungkin ada sedikit pergeseran warna jika asli BGR ditampilkan sebagai RGB).

"Grayscale Image" : Merupakan tampilan dari t_gray, yaitu gambar plat nomor yang telah dikonversi ke skala abu-abu. Semua informasi warna telah dihilangkan, dan gambar hanya terdiri dari nuansa hitam, abu-abu, dan putih. Detail pada plat nomor (huruf, angka, garis) masih terlihat jelas, tetapi dalam format monokrom. Ini adalah langkah pre-pemrosesan yang umum dalam banyak aplikasi visi komputer, seperti pengenalan karakter optik (OCR) pada plat nomor, karena algoritma seringkali bekerja lebih efisien pada gambar grayscale.

```
src = np.array([[0, 0], [0, 50], [300, 50], [300, 0]])

crp = np.array([[212, 493], [567, 303], [535, 256], [535, 256]])

crp2 = np.array([[165, 682], [199, 722], [529, 469], [496, 432]])
```

```
tform = transform.ProjectiveTransform()
tform.estimate(src, crp)
tform2 = transform.ProjectiveTransform()
tform2.estimate(src, crp2)

warped = transform.warp(t, tform, output_shape=(50, 300))
warped2 = transform.warp(t, tform2, output_shape=(50, 300))

fig, axs = plt.subplots(1, 3, figsize=(10, 10))
ax = axs.ravel()
ax[0].imshow(warped)
ax[0].set_title('Warped Image 1')
ax[1].imshow(t)
ax[1].plot(crp[:, 0], crp[:, 1], '.r')
ax[1].plot(src[:, 0], src[:, 1], '.b')
ax[1].set_title('Original Image with Points')

ax[2].imshow(warped2)
ax[2].set_title('Warped Image 2')
for a in axs:
    a.axis('off')

plt.tight_layout()
plt.show()
```

```
[44]: src = np.array([[0, 0], [0, 50], [300, 50], [300, 0]])
      crp = np.array([[212, 493], [567, 303], [535, 256], [535, 256]])
      crp2 = np.array([[165, 682], [199, 722], [529, 469], [496, 432]])

      tform = transform.ProjectiveTransform()
      tform.estimate(src, crp)
      tform2 = transform.ProjectiveTransform()
      tform2.estimate(src, crp2)

      warped = transform.warp(t, tform, output_shape=(50, 300))
      warped2 = transform.warp(t, tform2, output_shape=(50, 300))

      fig, axs = plt.subplots(1, 3, figsize=(10, 10))
      ax = axs.ravel()
      ax[0].imshow(warped)
      ax[0].set_title('Warped Image 1')
      ax[1].imshow(t)
      ax[1].plot(crp[:, 0], crp[:, 1], '.r')
      ax[1].plot(src[:, 0], src[:, 1], '.b')
      ax[1].set_title('Original Image with Points')

      ax[2].imshow(warped2)
      ax[2].set_title('Warped Image 2')
      for a in axs:
          a.axis('off')

      plt.tight_layout()
      plt.show()
```



202331084_Felix Ewaldo Panggabean

Penjelasan :

Bagian Inisialisasi Titik dan Transformasi Proyektif :

src = np.array([[0, 0], [0, 50], [300, 50], [300, 0]]) :

- **np.array() :** Membuat array NumPy.
- **Src :** Mendefinisikan empat titik sumber (source points) dalam koordinat piksel. Titik-titik ini biasanya dipilih dari gambar asli yang akan menjadi referensi untuk transformasi.
- **[0, 0] :** Sudut kiri atas.
- **[0, 50] :** Sudut kiri bawah.
- **[300, 50] :** Sudut kanan atas.
- **[300, 0] :** Sudut kanan bawah.
- **Catatan :** Koordinat ini terlihat seperti sudut-sudut persegi panjang standar, yang mungkin tidak sesuai dengan perspektif plat nomor yang sebenarnya.

crp = np.array([[212, 493], [567, 303], [535, 256], [535, 256]]) :

- **crp :** Mendefinisikan empat titik tujuan (destination points) yang sesuai dengan titik-titik src. Titik-titik ini adalah lokasi di mana src points akan dipetakan setelah transformasi.
- **[212, 493] :** Titik tujuan pertama.
- **[567, 303] :** Titik tujuan kedua.

- **[535, 256] :** Titik tujuan ketiga.
- **[535, 256] :** Titik tujuan keempat.
- **Catatan :** Ada dua titik tujuan yang sama ([535, 256]). Ini akan menyebabkan masalah dalam perhitungan transformasi proyektif karena membutuhkan minimal 4 pasang titik yang unik. Ini kemungkinan besar adalah kesalahan penulisan. Untuk transformasi proyektif, diperlukan 4 pasang titik unik yang tidak kolinear (tidak berada pada satu garis lurus) untuk mendefinisikan transformasi dengan benar.

crp2 = np.array([[165, 682], [199, 722], [529, 469], [496, 432]]) :

- **crp2 :** Mendefinisikan empat set titik tujuan kedua. Ini juga adalah titik-titik yang akan digunakan untuk transformasi proyektif kedua. Titik-titik ini terlihat lebih realistis untuk menangkap sudut-sudut plat nomor yang miring.

tform = transform.ProjectiveTransform() :

- **transform.ProjectiveTransform() :** Menginisialisasi objek transformasi proyektif dari skimage.transform. Transformasi proyektif (juga dikenal sebagai homografi) adalah transformasi non-linear yang dapat memetakan persegi ke sembarang quadilateral. Ini sangat berguna untuk mengoreksi perspektif.

tform.estimate(src, crp) :

- **tform.estimate() :** Metode ini menghitung matriks transformasi proyektif yang terbaik yang memetakan titik-titik src ke titik-titik crp. Ini akan menemukan matriks H sedemikian rupa sehingga H·src≈crp.

tform2 = transform.ProjectiveTransform() : Menginisialisasi objek transformasi proyektif kedua.

tform2.estimate(src, crp2) : Menghitung matriks transformasi proyektif kedua yang memetakan titik-titik src ke titik-titik crp2.

warped = transform.warp(t, tform, output_shape=(50, 300)) :

- **transform.warp() :** Fungsi dari skimage.transform yang menerapkan transformasi geometris (dalam hal ini, proyektif) pada gambar.
- **t :** Gambar input (gambar plat nomor asli, plat.jpg).
- **tform :** Matriks transformasi proyektif yang dihitung sebelumnya (menggunakan src dan crp).
- **output_shape=(50, 300) :** Ukuran output gambar yang diinginkan dalam format (tinggi, lebar). Gambar yang di-warp akan memiliki dimensi ini.

warped2 = transform.warp(t, tform2, output_shape=(50, 300)) :

- **transform.warp() :** Menerapkan transformasi proyektif kedua pada gambar asli.
- **t :** Gambar input.

- **tform2 :** Matriks transformasi proyektif kedua (menggunakan src dan crp2).
- **output_shape=(50, 300) :** Ukuran output gambar yang diinginkan (tinggi 50 piksel, lebar 300 piksel).

Bagian Plotting (Tampilan Hasil) :

fig, axs = plt.subplots(1, 3, figsize=(10, 10)) : Membuat figur Matplotlib dengan 1 baris dan 3 kolom subplot, ukuran 10x10 inci.

ax = axs.ravel() : Meratakan array axs menjadi 1D.

ax[0].imshow(warped) : Menampilkan gambar hasil warp pertama (warped) di subplot pertama.

ax[0].set_title('Warped Image 1') : Mengatur judul subplot pertama.

ax[1].imshow(t) : Menampilkan gambar asli (t) di subplot kedua.

ax[1].plot(crp[:, 1], crp[:, 0], '.r') :

- **crp[:, 1] :** Mengambil semua nilai kolom indeks 1 (y-koordinat) dari array crp.
- **crp[:, 0]:** Mengambil semua nilai kolom indeks 0 (x-koordinat) dari array crp.
- **'.r' :** Menentukan bahwa plot harus berupa titik (.) berwarna merah (r).
  Baris ini menggambar titik-titik tujuan (crp) pada gambar asli di subplot kedua. Titik-titik ini seharusnya menandai area yang ingin di "luruskan" atau diubah perspektifnya.

ax[1].plot(src[:, 1], src[:, 0], '.b') : Sama seperti di atas, tetapi menggambar titik-titik sumber (src) pada gambar asli di subplot kedua dengan warna biru (.b). Ini untuk menunjukkan korespondensi antara titik sumber dan titik tujuan.

ax[1].set_title('Original Image with Points') : Mengatur judul subplot kedua.

ax[2].imshow(warped2) : Menampilkan gambar hasil warp kedua (warped2) di subplot ketiga.

ax[2].set_title('Warped Image 2') : Mengatur judul subplot ketiga.

for a in axs: : Loop untuk setiap subplot.

a.axis('off') : Mematikan sumbu untuk tampilan yang bersih.

plt.tight_layout() : Menyesuaikan tata letak subplot.

plt.show() : Menampilkan figur.

Penjelasan Output :

Gambar output menunjukkan tiga subplot :

"Warped Image 1" : Merupakan hasil dari warped, menggunakan tform (dihitung dari src dan crp). Gambar ini terlihat hampir sepenuhnya hitam, kecuali sedikit garis horizontal putih di tengah. Ini adalah hasil dari masalah sebelumnya : Titik crp memiliki dua titik yang identik ([535, 256]), yang membuat perhitungan transformasi proyektif menjadi tidak stabil atau tidak valid. Ketika ada dua titik tujuan yang sama, transformasi tidak dapat ditentukan secara unik, dan hasilnya seringkali rusak, seperti yang terlihat di sini. Gambar ini pada dasarnya menunjukkan kegagalan transformasi karena input yang tidak benar.

"Original Image with Points" : Ini adalah gambar asli plat nomor.

- **Titik Biru :** Ini adalah titik-titik src ([[0, 0], [0, 500], [300, 0], [300, 500]]). Tampak seperti sudut-sudut persegi panjang di kiri atas dan kanan bawah gambar, tetapi posisinya mungkin tidak benar jika maksudnya adalah titik sudut dari plat nomor itu sendiri.
- **Titik Merah :** Ini adalah titik-titik crp ([[212, 493], [367, 383], [535, 256], [535, 256]]). Titik-titik ini seharusnya menandai area plat nomor yang ingin diluruskan. Namun, karena dua titik terakhir sama, ini secara visual hanya menampilkan 3 titik yang berbeda. Titik-titik ini memang berada di area plat nomor, tetapi karena ada duplikasi, transformasi pertama tidak bekerja.

"Warped Image 2" : Merupakan hasil dari warped2, menggunakan tform2 (dihitung dari src dan crp2).

- Gambar ini menunjukkan potongan plat nomor yang telah diluruskan perspektifnya. Terlihat bahwa bagian plat nomor (terutama angka "0") telah "diratakan" seolah-olah dilihat dari depan, meskipun masih ada bagian hitam yang mengisi area kosong di sekitarnya (karena output_shape kecil yaitu 50x300 piksel). Transformasi ini berhasil karena titik-titik crp2 kemungkinan besar adalah titik-titik unik yang dipilih dengan benar dari empat sudut plat nomor.
- **Kesimpulan :** Kode ini mendemonstrasikan aplikasi transformasi proyektif (homografi) menggunakan skimage.transform untuk mengoreksi perspektif gambar, khususnya plat nomor. Pentingnya pemilihan titik sumber dan tujuan yang benar (unik dan tidak kolinear) sangat ditekankan oleh perbandingan antara "Warped Image 1" (yang gagal karena titik tujuan duplikat) dan "Warped Image 2" (yang berhasil meluruskan perspektif plat nomor karena titik tujuan yang valid). Transformasi proyektif ini adalah langkah kunci dalam banyak aplikasi visi komputer, seperti sistem pengenalan plat nomor otomatis (ANPR).