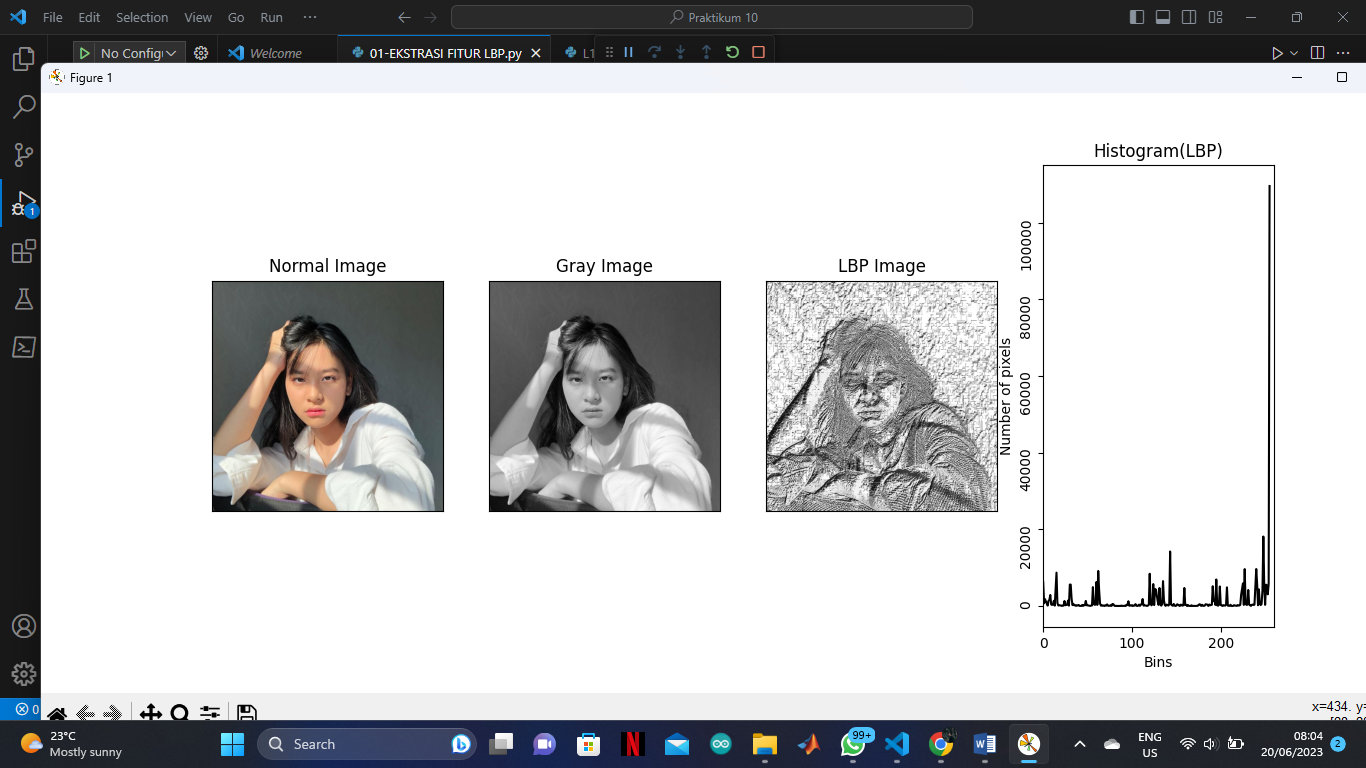
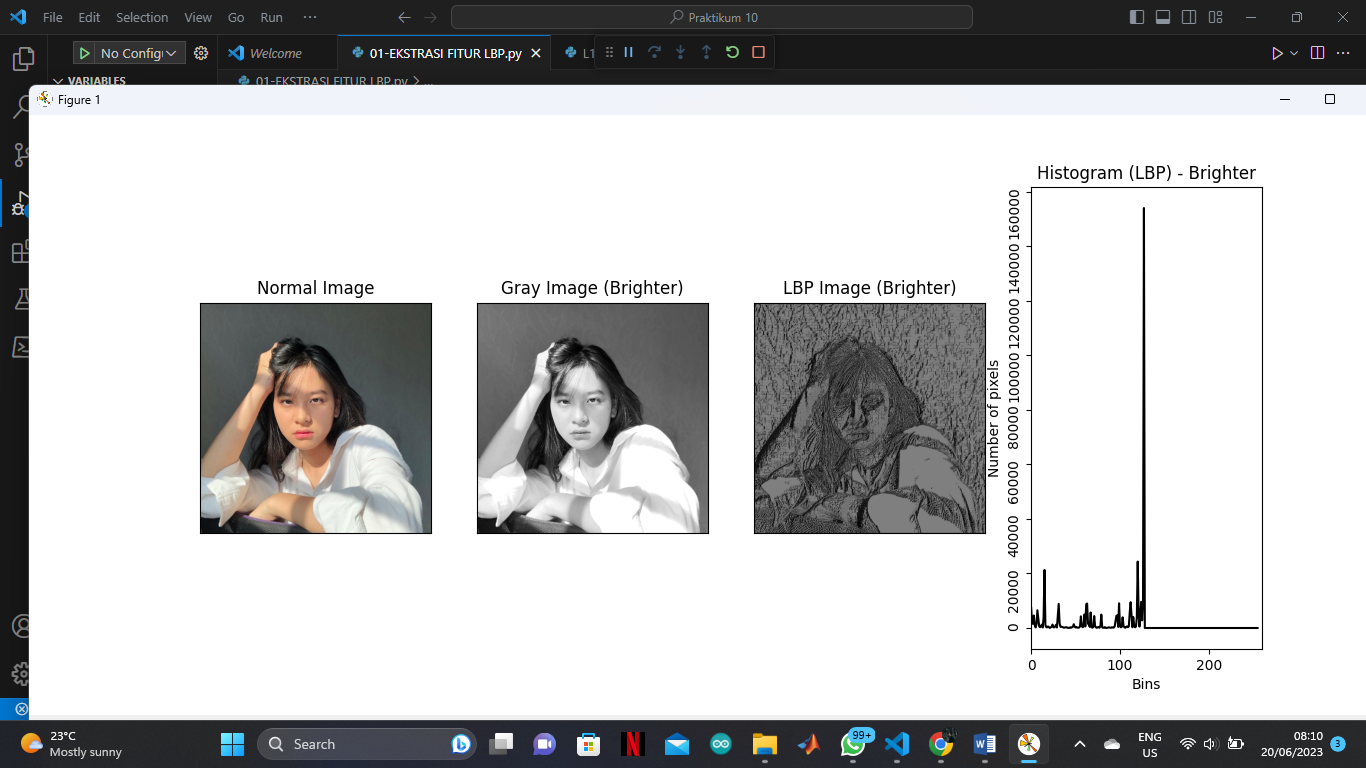
Nama : Reza Fajriatul Haq

NIM : 1207070103

1. EKSTRASI FITUR LBP





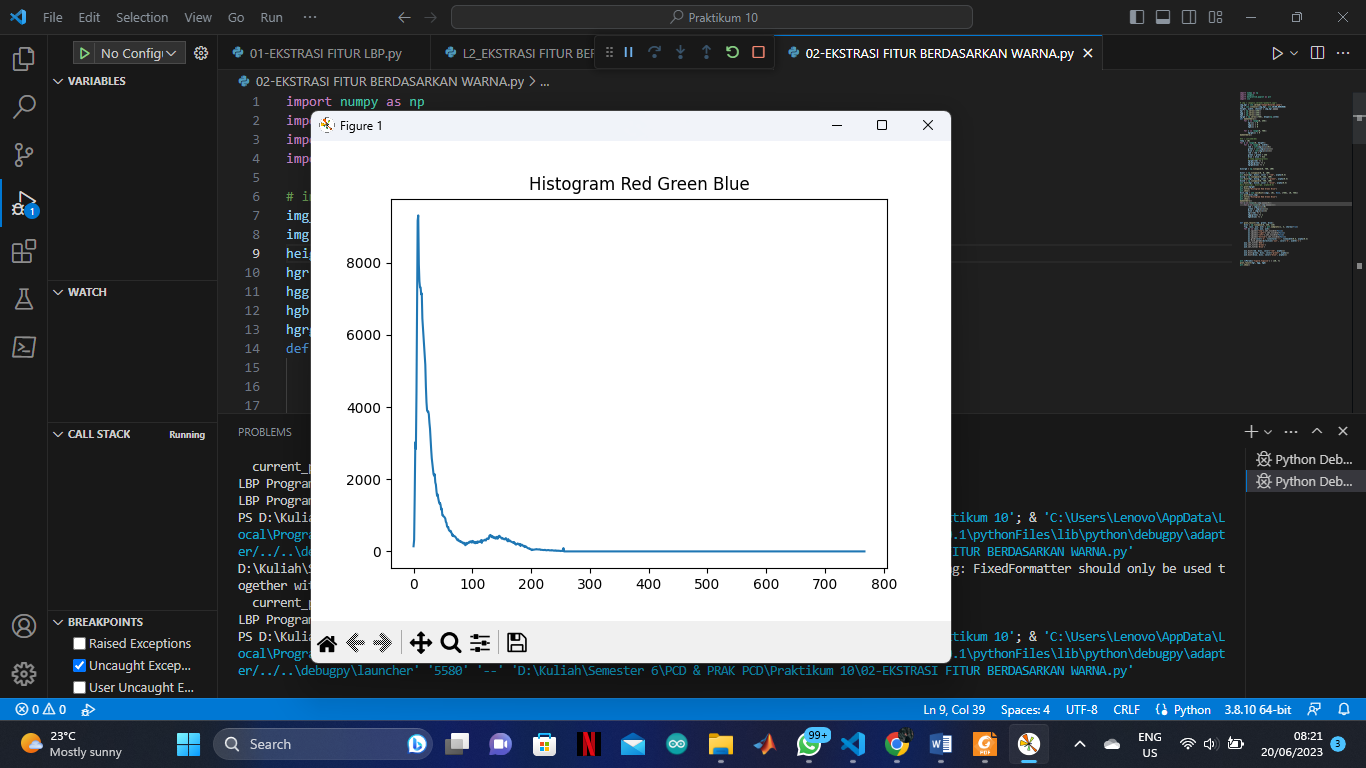
Percobaan ini merupakan implementasi analisis Local Binary Patterns (LBP) pada sebuah gambar. LBP adalah metode yang digunakan untuk mengekstraksi tekstur lokal dari sebuah gambar dengan membandingkan setiap piksel dengan tetangganya. Proses dimulai dengan membaca gambar input dan mengubahnya menjadi gambar keabuan (grayscale). Setelah itu, dilakukan iterasi pada setiap piksel gambar keabuan untuk menghitung nilai LBP menggunakan aturan pola biner. Setiap piksel dibandingkan dengan tetangganya, dan jika nilai piksel tetangga lebih besar atau sama dengan piksel pusat, nilai LBP akan menjadi 1, jika tidak, nilai LBP akan tetap 0. Hasil dari perhitungan LBP disimpan dalam gambar LBP. Selanjutnya, dilakukan perhitungan histogram LBP menggunakan fungsi `cv2.calcHist`. Histogram LBP menggambarkan distribusi frekuensi nilai LBP yang ada dalam gambar.

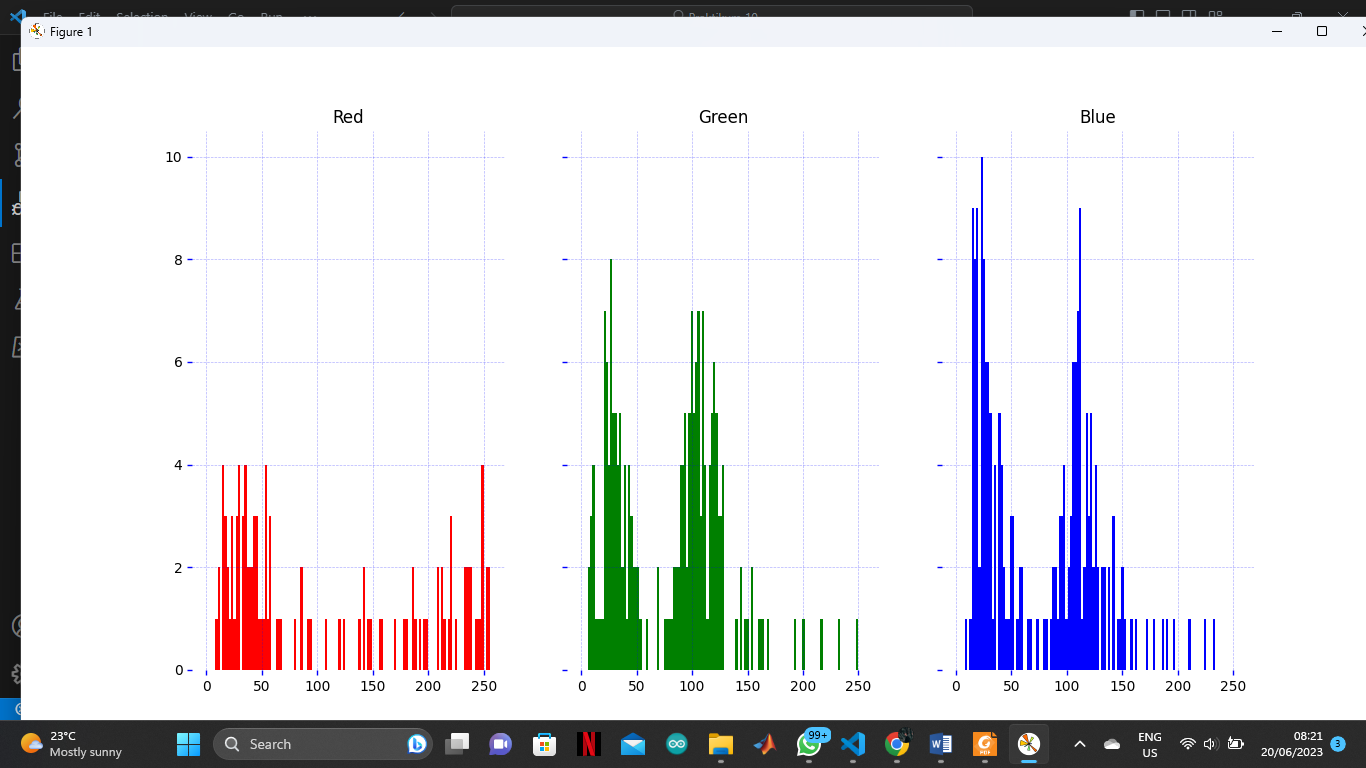
Output dari analisis LBP ditampilkan dalam beberapa visualisasi. Output pertama menampilkan gambar asli, gambar keabuan, gambar LBP, dan histogram LBP tanpa peningkatan kecerahan. Output kedua menunjukkan hasil yang sama, namun dengan peningkatan kecerahan sebesar 50 pada gambar asli sebelum melakukan konversi keabuan..Melalui analisis LBP ini, kita dapat memperoleh pemahaman tentang tekstur lokal dalam gambar dan mengidentifikasi pola-pola yang ada.

1. EKSTRASI FITUR BERDASARKAN WARNA



Kode tersebut mengimplementasikan metode Local Binary Pattern (LBP) untuk melakukan deskripsi tekstur pada suatu citra. Fungsi `get\_pixel` digunakan untuk membandingkan nilai piksel sekitar pusat dengan nilai piksel pusat. Jika nilai piksel sekitar pusat lebih besar atau sama dengan nilai piksel pusat, fungsi tersebut mengembalikan nilai 1, dan jika tidak, mengembalikan nilai 0. Fungsi `lbp\_calculated\_pixel` menghitung nilai LBP untuk piksel tertentu dengan menggunakan fungsi `get\_pixel` untuk mendapatkan nilai biner dari piksel sekitar pusat, sesuai dengan format LBP yang telah ditentukan. Fungsi `show\_output` digunakan untuk menampilkan citra dan histogramnya menggunakan library matplotlib.

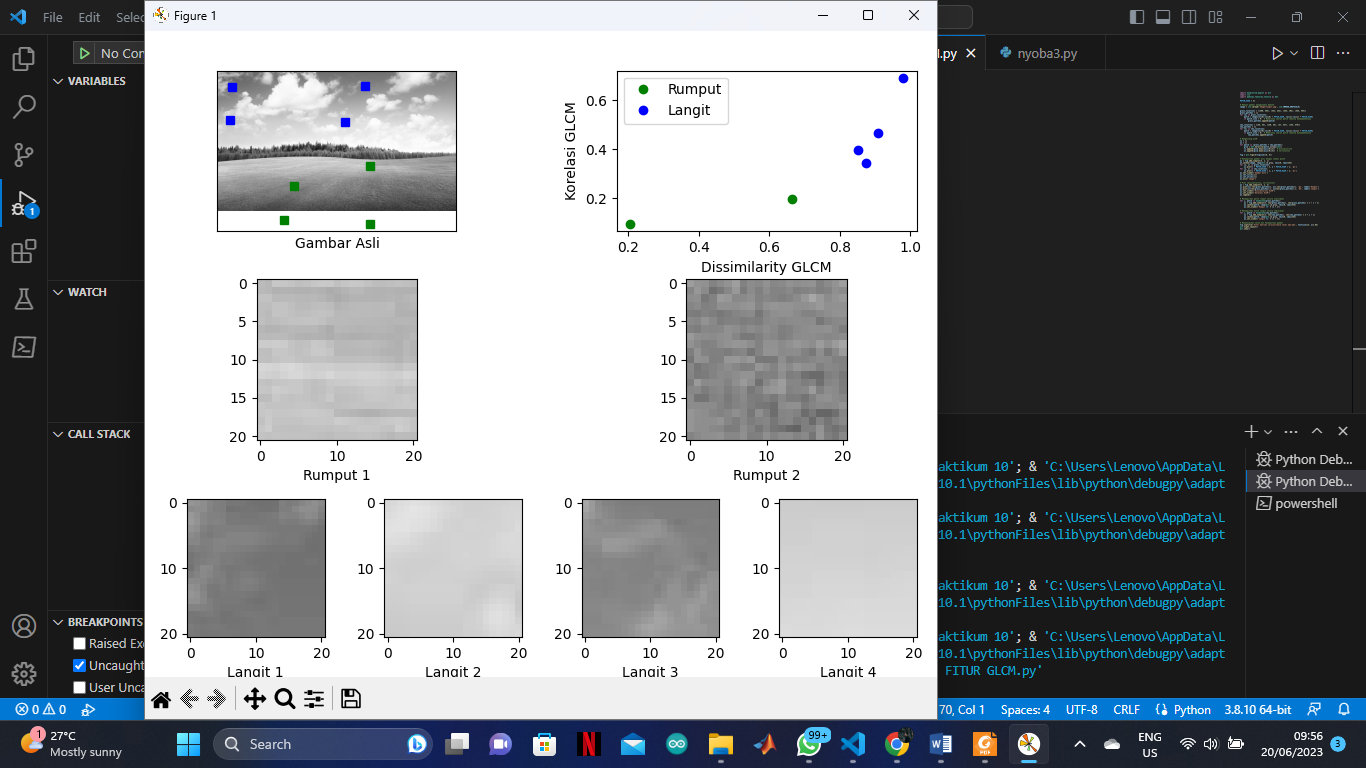




Pada percobaan ini terdapat fungsi brighter yang digunakan untuk menerangi citra dengan menambahkan nilai ke setiap saluran warna. Fungsi ini mengembalikan citra yang telah diterangi. Selanjutnya, dilakukan pemrosesan citra menggunakan fungsi lbp. Pertama, citra asli diubah menjadi citra grayscale dan diolah menggunakan fungsi lbp. Selanjutnya, citra yang telah diterangi dengan nilai tertentu juga diubah menjadi citra grayscale dan diolah menggunakan fungsi lbp.

1. EKSTRASI FITUR GLCM

Pada percobaan ini liblary mahotas digunakan sebagai alternatif untuk menghitung fitur GLCM karena fungsi greycomatrix pada liblalry scikit-image tidak tersedia dan eror di laptop saya. Mahotas adalah pustaka komputasi ilmiah untuk pemrosesan gambar yang menyediakan berbagai fitur ekstraksi yang kuat

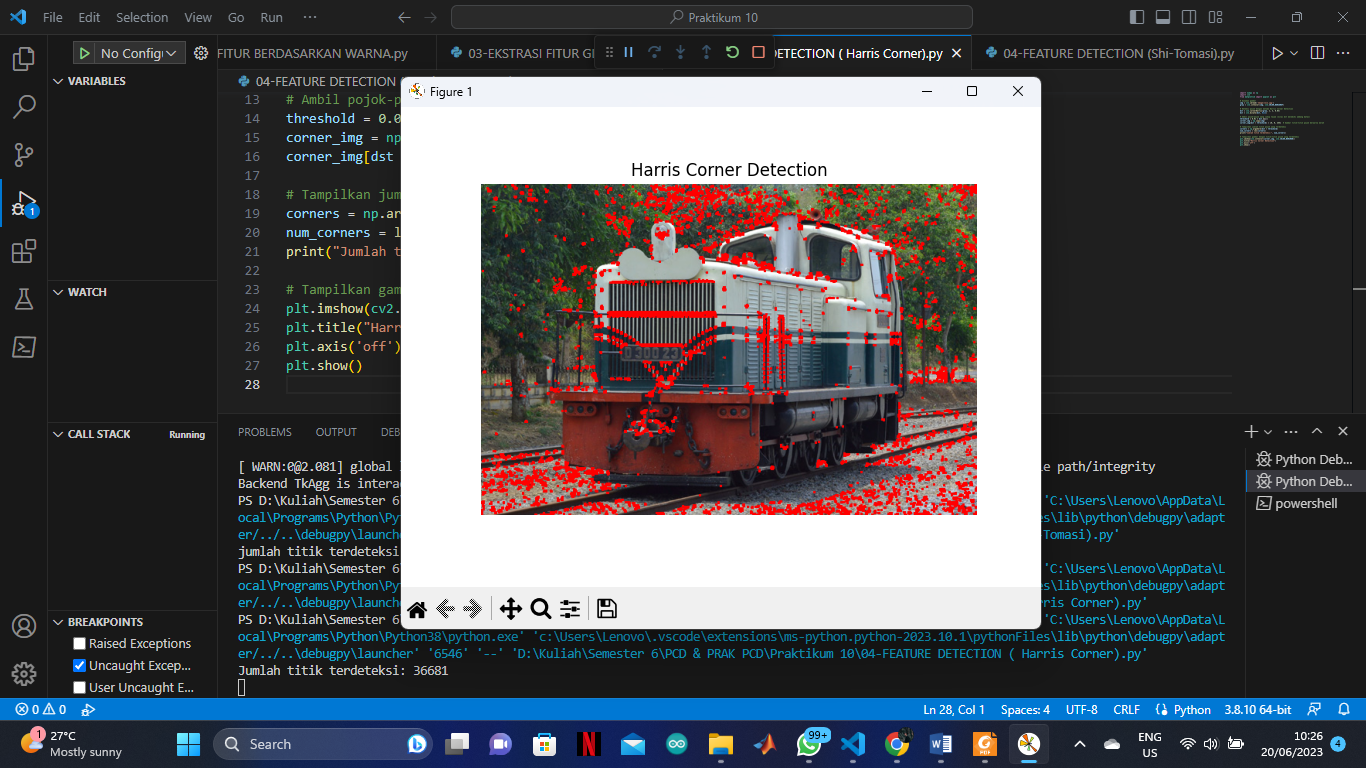


Pada percobaan ini merupakan contoh implementasi untuk menghitung fitur matriks co-occurrence level abu-abu (GLCM). Pada program ini dilakukan perulangan untuk setiap patch yang telah ditentukan. Pada setiap iterasi, dilakukan perhitungan GLCM menggunakan fungsi `mht.haralick` dari library mahotas pada patch yang bersangkutan. Dari hasil perhitungan GLCM, diambil nilai rata-rata dari fitur Dissimilarity dan Correlation, yang kemudian disimpan dalam daftar `xs` dan `ys`.Setelah itu, dilakukan visualisasi menggunakan matplotlib.

Pada gambar pertama, ditampilkan gambar asli dengan lokasi patch yang ditandai oleh titik-titik hijau (rumput) dan biru (langit). Pada gambar kedua, ditampilkan plot scatter antara Dissimilarity dan Correlation untuk setiap kategori. Pada gambar-gambar selanjutnya, ditampilkan patch-patch rumput dan langit secara individual.

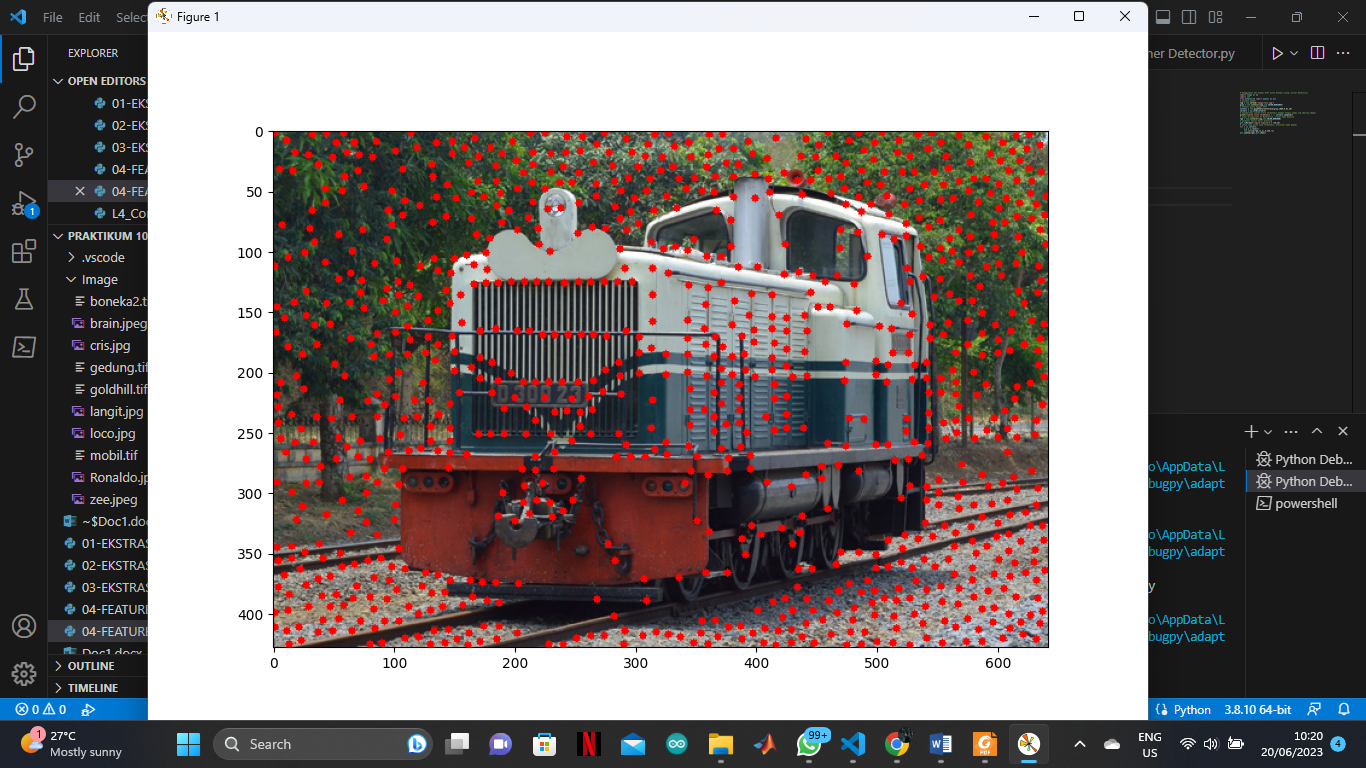
Program ini menganalisis perbedaan fitur tekstur antara kategori "rumput" dan "langit" pada gambar. Dengan menggunakan GLCM, fitur-fitur seperti Dissimilarity dan Correlation dapat dihitung dan digunakan untuk membedakan kedua kategori tersebut. Visualisasi yang dihasilkan membantu dalam pemahaman dan interpretasi hasil analisis.

1. FEATURE DETECTION
2. Dengan Harris Corner



Pada percobaan ini melakukan deteksi sudut pada gambar menggunakan metode Harris Corner Detection. Metode ini membantu mengidentifikasi sudut-sudut penting dalam gambar yang dapat digunakan untuk berbagai tujuan, seperti pelacakan objek, pengenalan objek, atau rekonstruksi 3D. Hasil deteksi sudut ditandai dengan warna merah pada gambar output, sehingga memudahkan dalam memvisualisasikan sudut-sudut yang terdeteksi. Jumlah titik sudut yang terdeteksi juga ditampilkan sebagai informasi tambahan. Pada deteksi sudut dengan metode Harris corner ini Jumlah titik terdeteksi yaitu 36681 titik

1. Dengan Shi-Tomasi



Pada percobaan ini yaitu melakukan deteksi sudut pada gambar menggunakan metode Shi-Tomasi GFTT. Metode ini dapat digunakan untuk menemukan titik sudut penting yang berpotensi sebagai fitur dalam berbagai aplikasi pengolahan gambar, seperti pelacakan objek, deteksi objek, atau pencocokan pola. Hasil deteksi sudut ditampilkan pada gambar dengan menandai setiap titik sudut dengan lingkaran kecil. Jumlah titik sudut yang terdeteksi juga ditampilkan sebagai informasi tambahan. Pada metode Shi-tomasi ini jumlah titik terdeteksi yaitu 1000 titik

Metode Harris Corner Detection dan Shi-Tomasi Good Features to Track (GFTT) adalah dua metode yang digunakan untuk mendeteksi sudut atau pojok dalam gambar. Harris Corner Detection menggunakan perhitungan perbedaan intensitas piksel dalam berbagai arah untuk mengidentifikasi titik sudut yang tajam. Metode ini memperhitungkan struktur intensitas di sekitar titik yang sedang dianalisis dan menghasilkan skor respons sudut. Di sisi lain, Shi-Tomasi GFTT menggunakan perhitungan nilai kecil dari eigenvalue matriks kecil untuk mengidentifikasi titik sudut yang signifikan. Metode ini lebih sensitif terhadap sudut yang tajam dan mempertimbangkan nilai eigenvalue terkecil. Hasil yang dihasilkan oleh kedua metode memiliki perbedaan. Harris Corner Detection cenderung menghasilkan respons sudut yang lebih banyak, dapat dilihat pada hasil percobaan diatas Jumlah titik terdeteksi yaitu 36681 titik sementara Shi-Tomasi GFTT menghasilkan sudut yang lebih tajam dan memiliki respons sudut yang lebih selektif. Pada percobaan diatas jumlah titik terdeteksi yaitu 1000 titik

Pemilihan antara kedua metode ini tergantung pada konteks aplikasi dan preferensi pengguna, serta harus mempertimbangkan faktor-faktor seperti parameter yang digunakan, jenis gambar, dan lingkungan visual.

1. FeatureDetectionandMatching

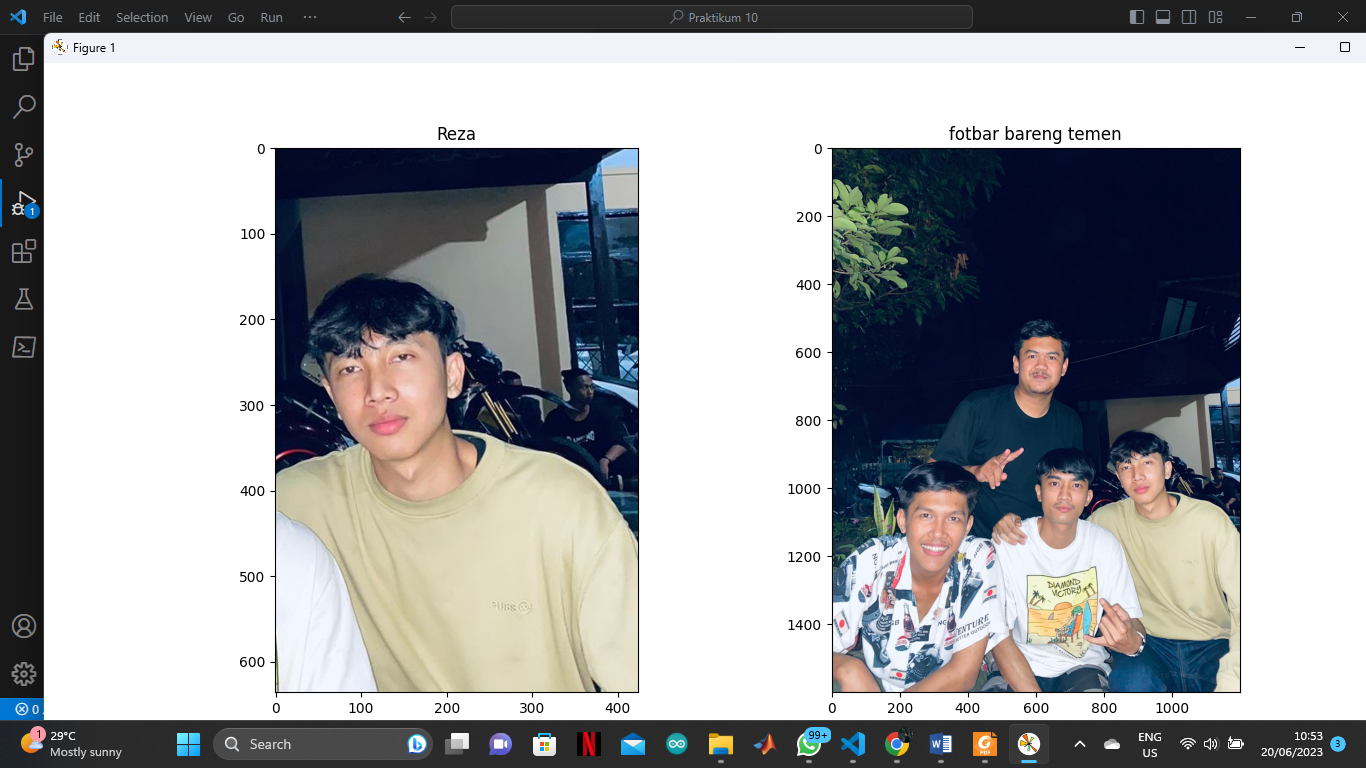


Kodingan di atas merupakan contoh implementasi deteksi dan pencocokan fitur menggunakan algoritma SIFT (Scale-Invariant Feature Transform). Pertama, dua gambar dimasukkan sebagai input, yaitu `img1` yang merupakan gambar yang dituju, dan `img2` yang merupakan gambar yang dicari. Gambar-gambar tersebut kemudian dikonversi menjadi citra grayscale agar lebih mudah diolah. Metode `detectAndCompute()` dari objek SIFT digunakan untuk mengidentifikasi keypoints (titik-titik penting) pada gambar dan menghasilkan deskriptor untuk setiap titik. Deskriptor ini merepresentasikan fitur-fitur yang ada pada gambar. Setelah itu, dilakukan pencocokan fitur antara kedua gambar menggunakan matcher Brute-Force .Matcher ini membandingkan deskriptor dari kedua gambar dan mencari kecocokan berdasarkan jarak terdekat antara deskriptor tersebut.

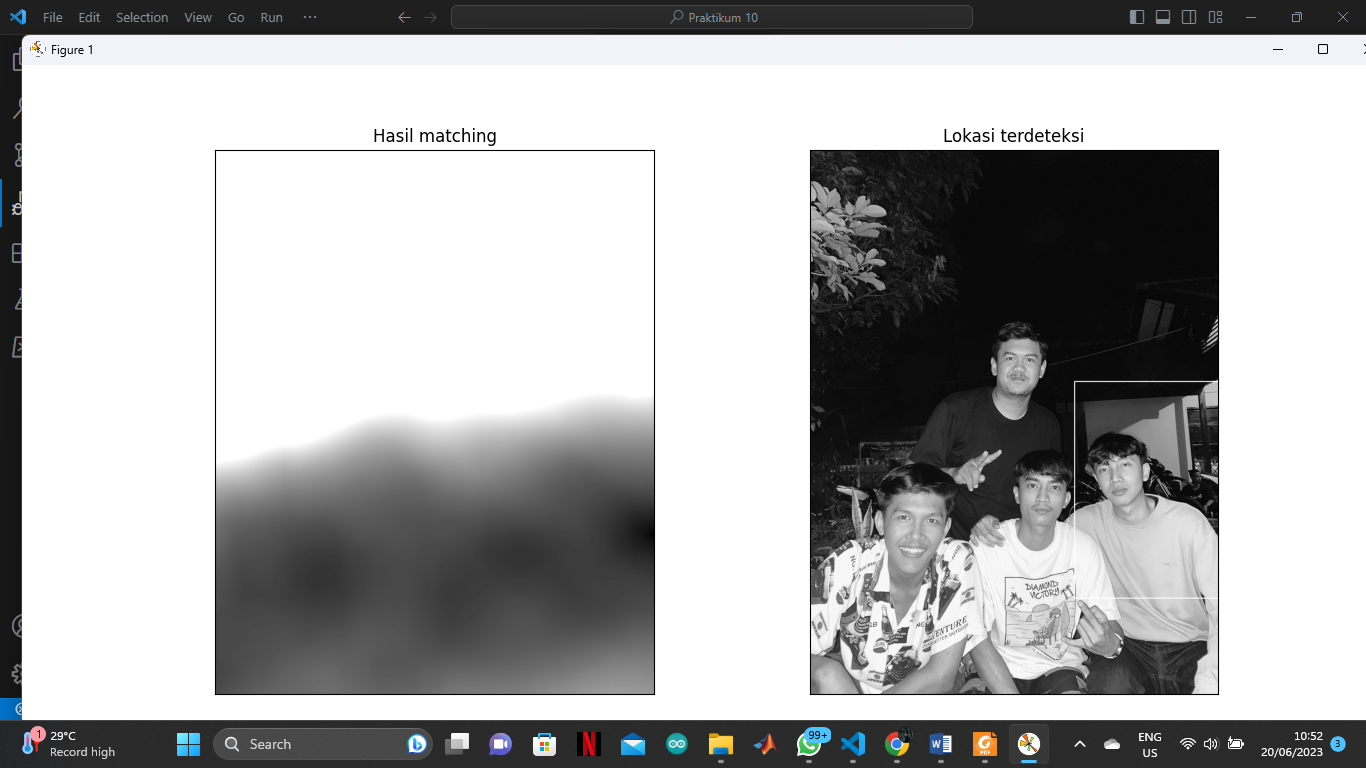
Hasil kecocokan kemudian diurutkan berdasarkan jarak terdekat dan hanya 10 kecocokan terbaik yang dipilih. Ke-10 kecocokan tersebut ditampilkan pada gambar baru (`img3`) dengan menggunakan fungsi `drawMatches()`. Gambar ini menunjukkan kecocokan antara kedua gambar dengan menandai fitur-fitur yang cocok menggunakan tanda panah.

Metode ini sangat berguna dalam berbagai aplikasi seperti pencocokan objek, pencarian gambar, atau pemosisian gambar. Dengan mendeteksi dan mencocokkan fitur-fitur yang unik pada gambar, kita dapat mendapatkan informasi tentang kecocokan antara dua gambar dan bahkan mengetahui transformasi yang terjadi antara keduanya. Dengan demikian, metode ini memberikan kemampuan untuk memahami dan memanipulasi gambar-gambar secara lebih lanjut.

1. template matching mendeteksi Reza pada gambar



Pada kodingan di atas, terdapat penggunaan Matplotlib untuk menampilkan dua gambar secara bersamaan. Awalnya, gambar dengan nama 'sha1.png' dipanggil dan diubah ke mode warna RGB menggunakan fungsi cv2.cvtColor(). Kemudian, gambar tersebut ditampilkan pada subplot pertama dengan nomor 121 dan diberi judul 'Reza'. Selanjutnya, gambar dengan nama 'sha.jpg' dipanggil dan juga diubah ke mode warna RGB. Gambar tersebut kemudian ditampilkan pada subplot kedua dengan nomor 122 dan diberi judul 'Foto bareng temen'. Terakhir, dengan menggunakan plt.show(), jendela plot yang berisi kedua gambar ditampilkan. Dengan menggunakan kodingan ini, kita dapat melihat dan membandingkan kedua gambar secara visual dalam satu tampilan yang mudah dipahami.



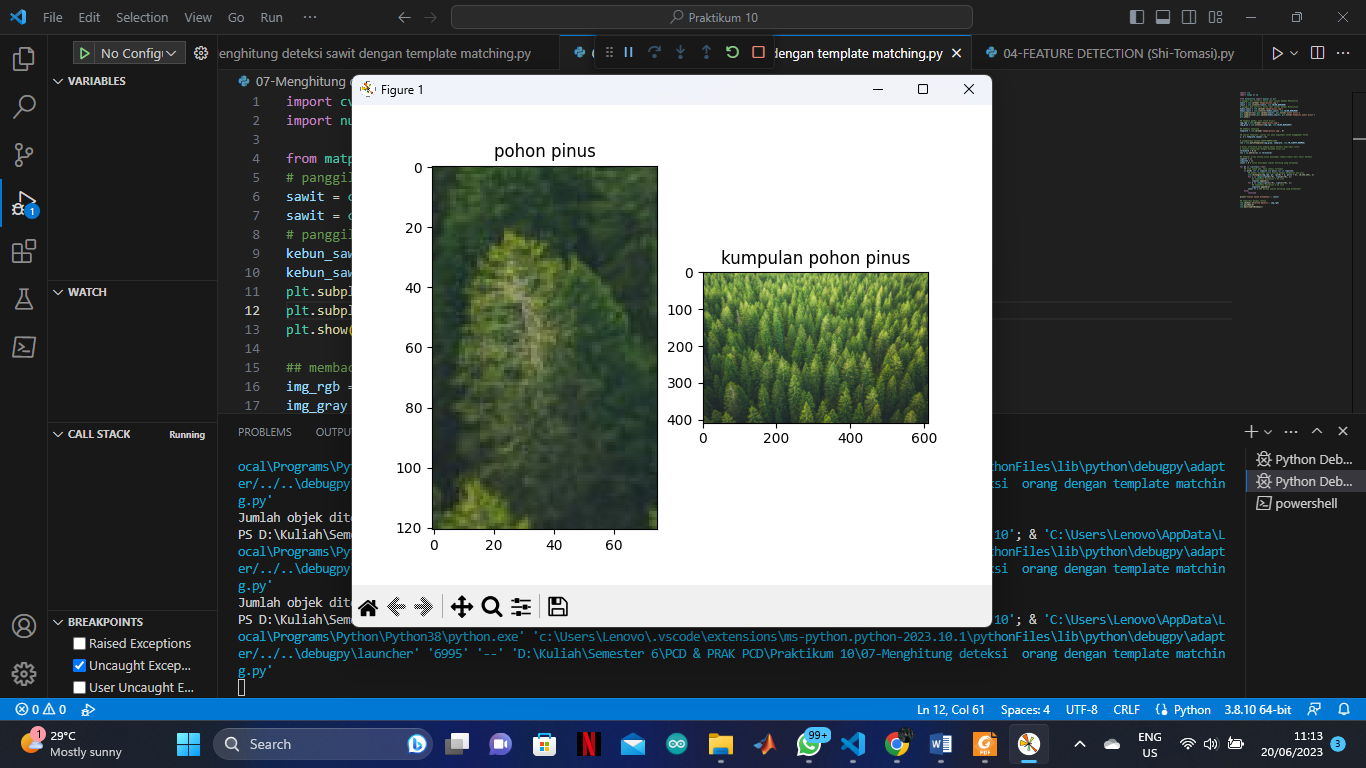
Output dari percobaan ini adalah tampilan visual dari proses template matching yang dilakukan pada citra utama. Pada subplot pertama (121) dengan judul 'Hasil matching', ditampilkan citra hasil template matching (res) yang berupa citra grayscale. Citra ini menunjukkan tingkat kesamaan antara

citra utama dan citra template pada setiap piksel. Nilai yang lebih tinggi menandakan tingkat kesamaan yang lebih tinggi antara kedua citra. Pada subplot kedua (122) dengan judul 'Lokasi terdeteksi', ditampilkan citra utama dengan kotak yang menunjukkan lokasi terdeteksi. Kotak tersebut menandai lokasi di mana citra template cocok dengan citra utama secara maksimum atau minimum berdasarkan metode yang digunakan. Ketebalan garis kotak adalah 2 dan warna kotak adalah putih.

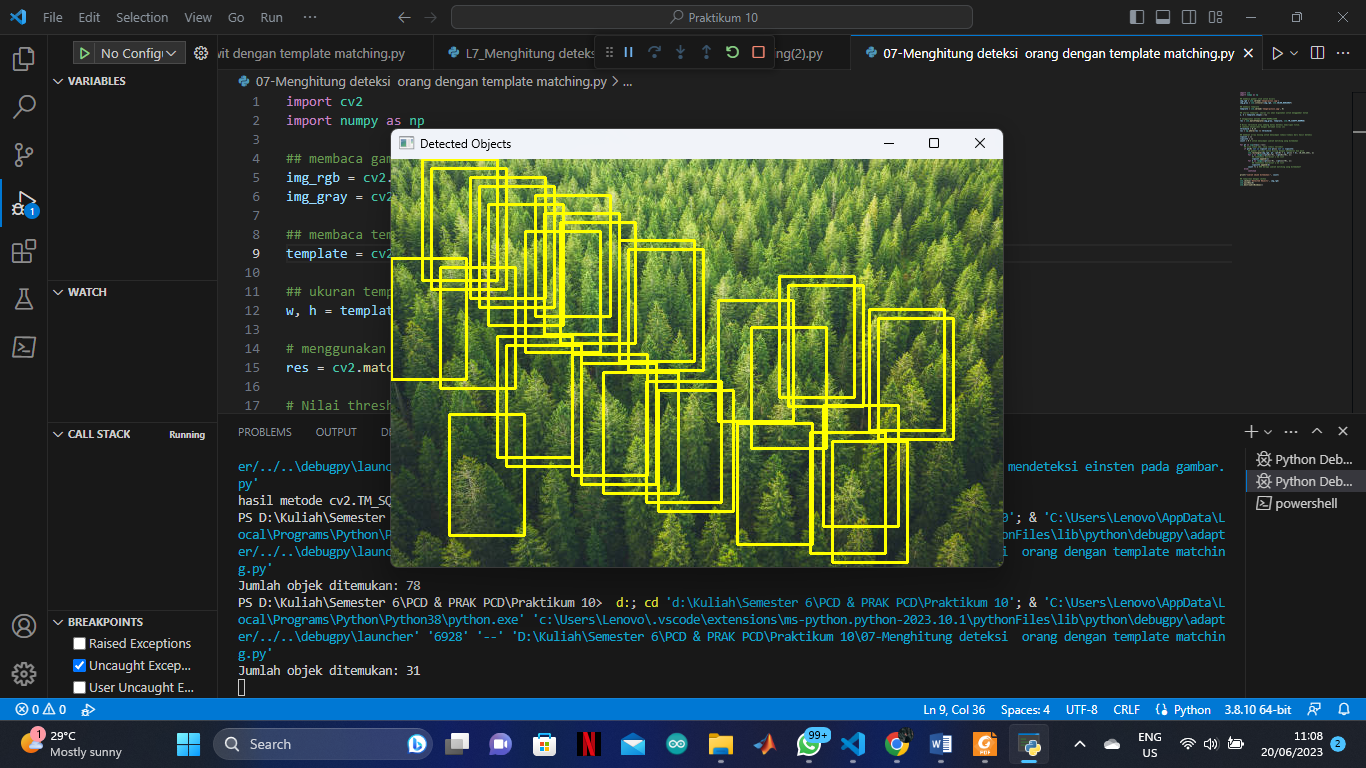
Dengan melihat output tersebut, kita dapat menganalisis bagaimana citra template cocok dengan citra utama menggunakan berbagai metode template matching yang telah diimplementasikan. Kita dapat melihat tingkat kesamaan antara kedua citra dan lokasi di mana kesamaan tersebut terjadi. Hal ini dapat membantu dalam aplikasi pengenalan objek, pencocokan pola, atau pemrosesan citra lainnya.

.

1. Menghitung deteksi pinus dengan template matching



Output dari kodingan tersebut akan terdiri dari dua bagian. Pertama, akan ditampilkan plot dengan dua gambar, yaitu 'pohon pinus' (subplot pertama) dan 'kumpulan pohon pinus' (subplot kedua). Kedua, akan ditampilkan gambar asli dengan kotak-kotak kuning yang menandai objek yang terdeteksi. Pada plot pertama, gambar 'pinus1.jpg' akan ditampilkan dengan judul 'pohon pinus'. Plot kedua akan menampilkan gambar 'pinus.jpg' dengan judul 'kumpulan pohon pinus'. Dengan menggunakan plt.show(), jendela plot yang berisi kedua gambar akan ditampilkan. Selanjutnya, akan dilakukan deteksi objek pada gambar 'pinus.jpg' menggunakan template matching. Gambar tersebut akan diubah menjadi grayscale menggunakan cv2.cvtColor(). Template yang digunakan adalah gambar 'pinus1.jpg' yang juga akan diubah menjadi grayscale. Metode yang digunakan dalam template matching adalah TM\_CCOEFF\_NORMED, yang menghasilkan nilai korelasi normalisasi antara citra dan template. Threshold (ambang batas) digunakan untuk menentukan tingkat kemiripan yang diterima sebagai deteksi. Nilai threshold dapat diubah untuk mengatur sensitivitas deteksi.



Output dari kodingan ini adalah tampilan gambar utuh ('pinus.jpg') dengan kotak persegi kuning yang menandai lokasi objek yang mirip dengan template ('pinus1.jpg'). Pada awalnya, gambar utuh dibaca dan dikonversi ke dalam citra skala abu-abu menggunakan cv2.cvtColor(). Selanjutnya, template juga dibaca sebagai citra skala abu-abu. Kemudian, dilakukan proses template matching menggunakan metode TM\_CCOEFF\_NORMED dengan memanggil cv2.matchTemplate(). Hasil template matching disimpan dalam variabel 'res'. Selanjutnya, ditentukan threshold atau ambang batas untuk deteksi kemiripan titik. Nilai threshold dapat diubah sesuai kebutuhan. Lokasi-lokasi yang memenuhi threshold kemudian diidentifikasi menggunakan np.where() dan disimpan dalam variabel 'loc'.

Dalam proses iterasi, dilakukan pengecekan untuk setiap lokasi yang ditemukan. Jika lokasi tersebut belum ada dalam list 'lspoint' dan 'lspoint2', maka kotak persegi kuning dengan ketebalan 2 piksel digambar pada gambar utuh menggunakan cv2.rectangle(). Selain itu, koordinat x dan y dari kotak persegi tersebut ditambahkan ke dalam list 'lspoint' dan 'lspoint2' untuk menghindari deteksi ganda pada lokasi yang sama. Jumlah matching yang ditemukan dihitung dengan menginkremen 'count'.

Output yang berupa gambar utuh dengan kotak persegi kuning yang menandai lokasi objek ditampilkan menggunakan cv2.imshow(). Pengguna dapat melihat jumlah objek yang ditemukan yang dicetak di layar.

Dengan menggunakan metode ini, kita dapat mendeteksi objek yang mirip dengan template pada gambar utuh dengan menggambar kotak persegi kuning pada lokasi yang sesuai. Output ini dapat digunakan untuk berbagai aplikasi, seperti deteksi objek, identifikasi, atau pemrosesan citra berbasis template matching.