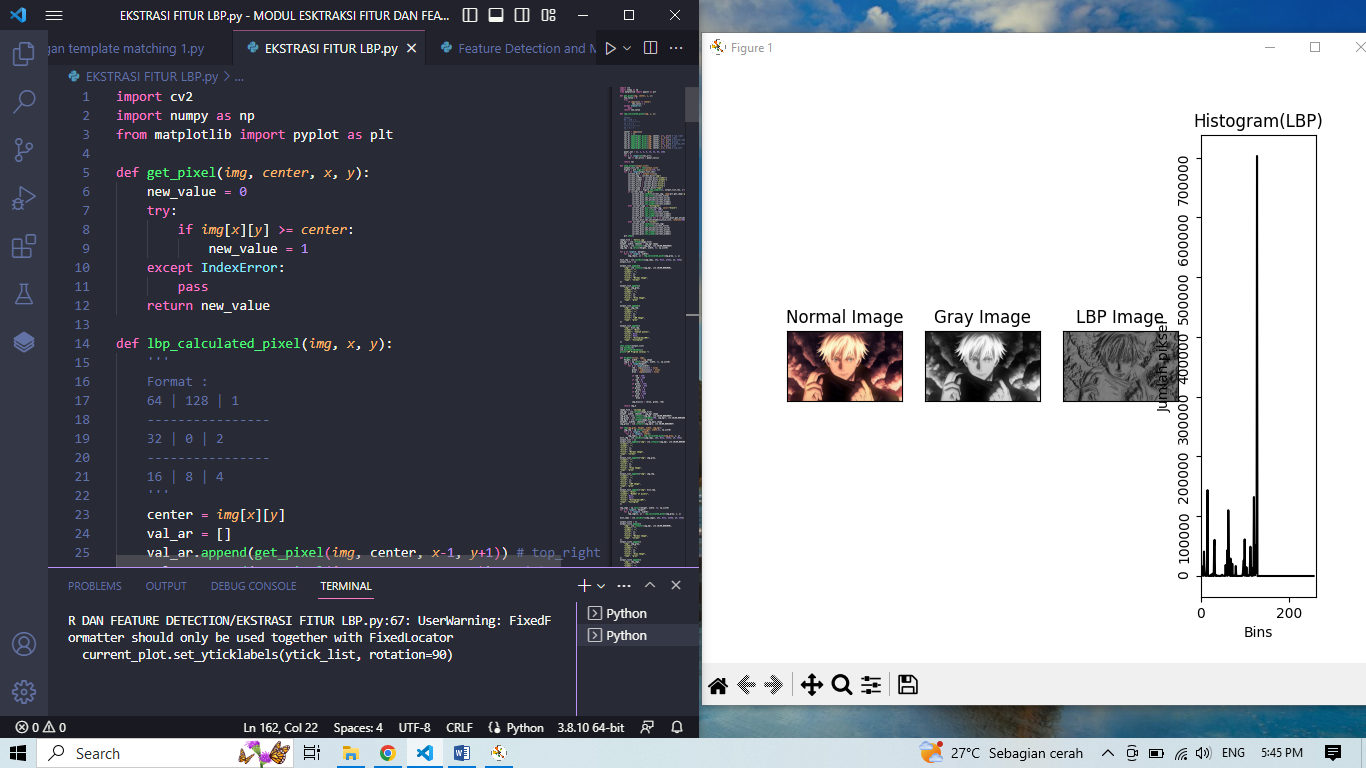
Nama: Haddy Firdaus

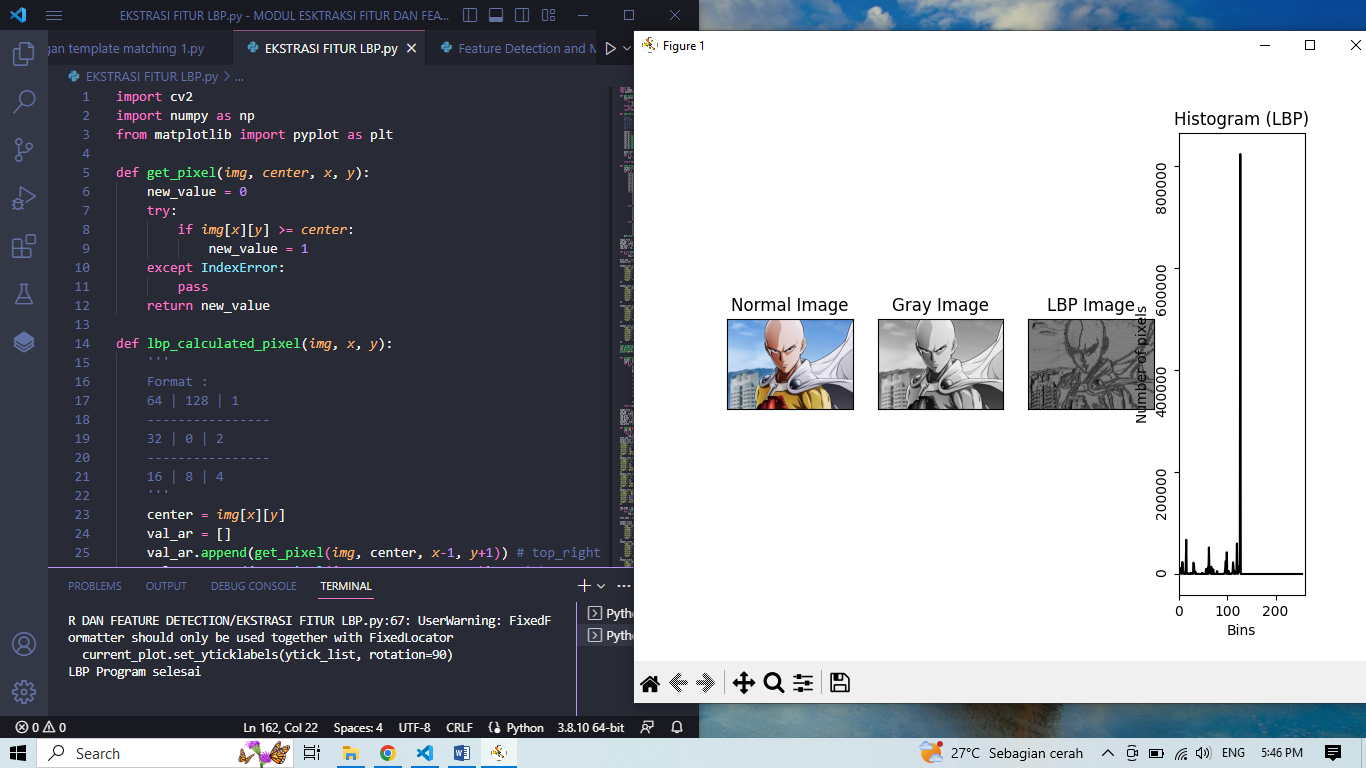
NIM : 1207070050

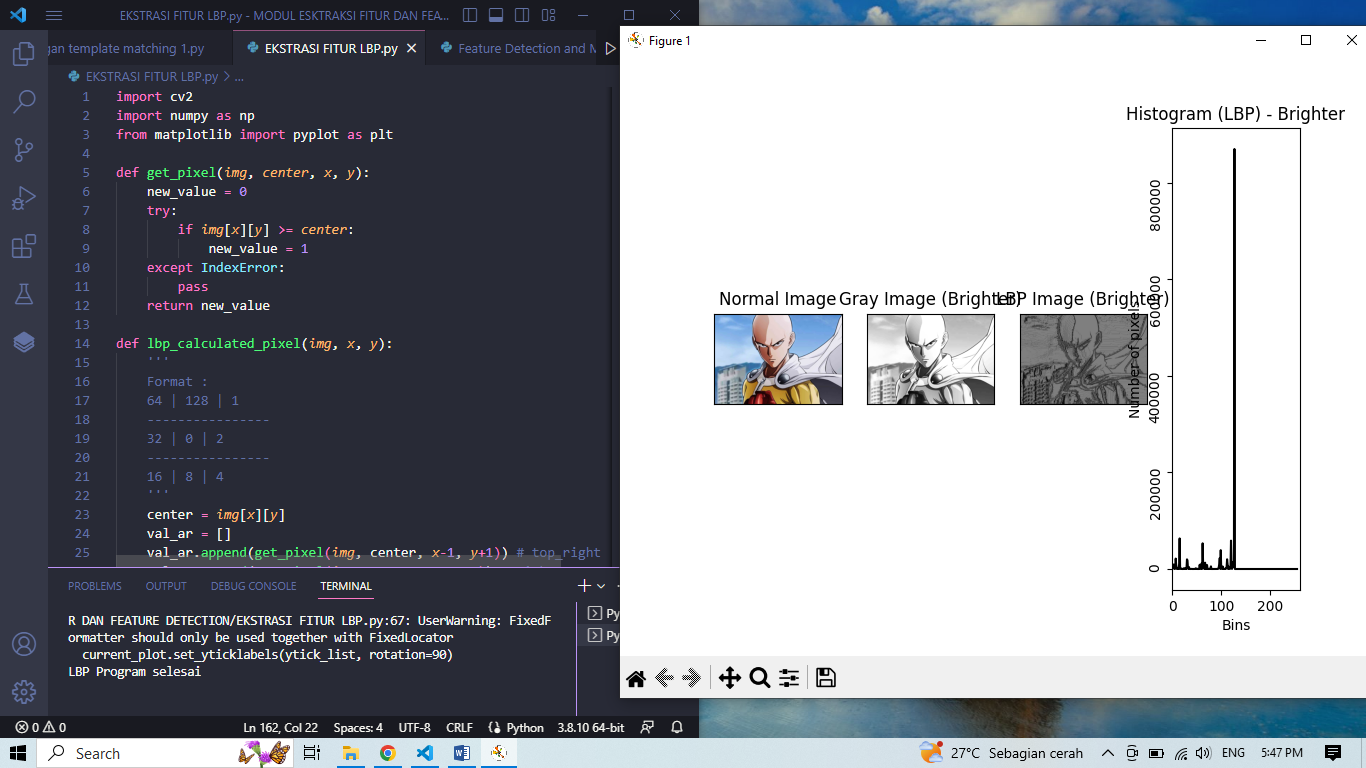
Kelas : Telekomunikasi

PRAKTIKUM PCD

* **EKSTRASI FITUR LBP**





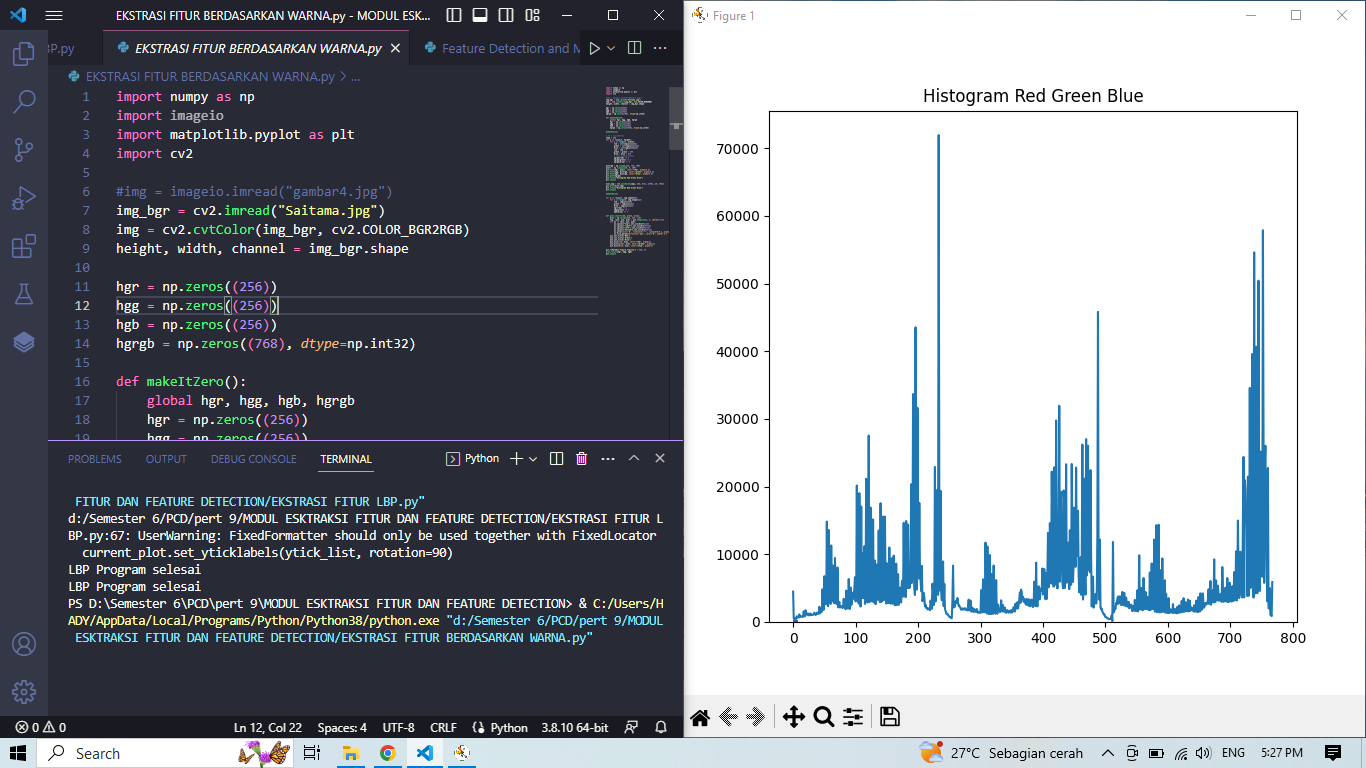


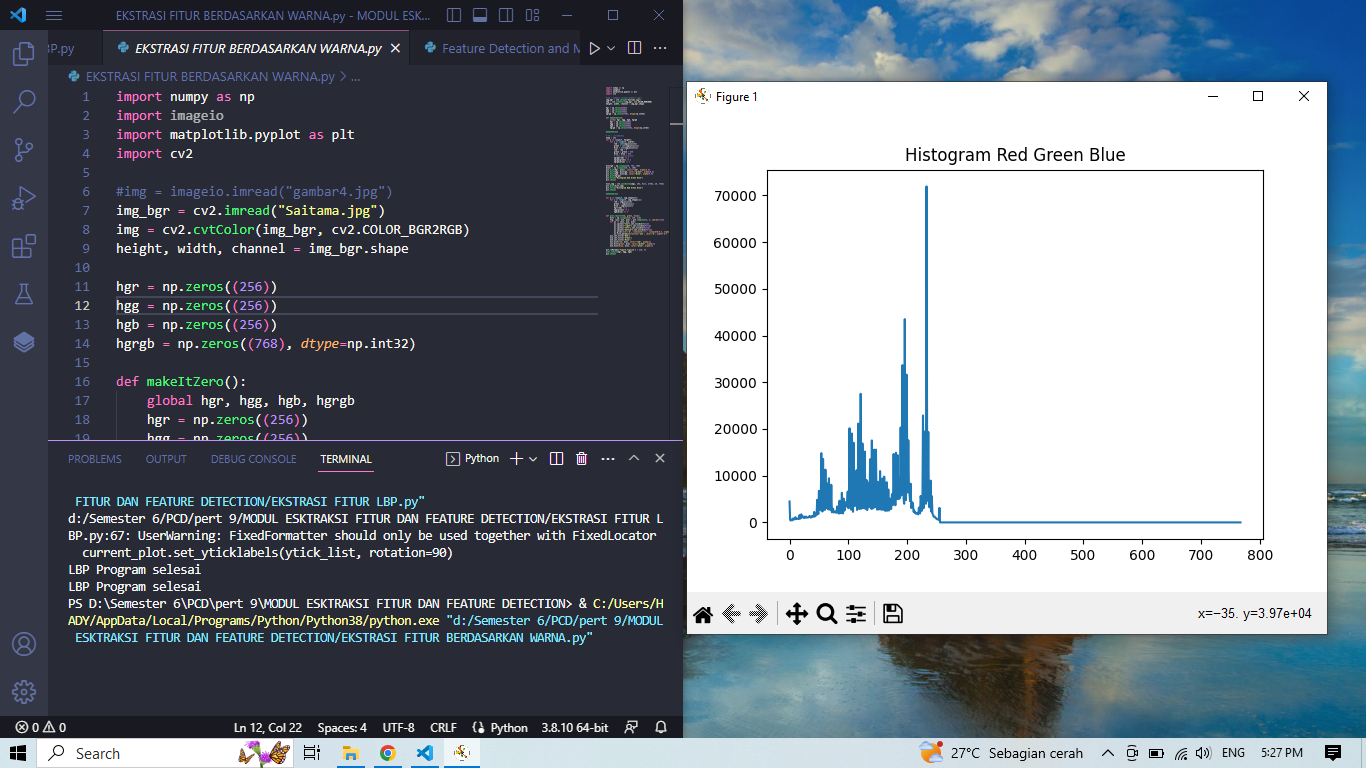
Analisis:

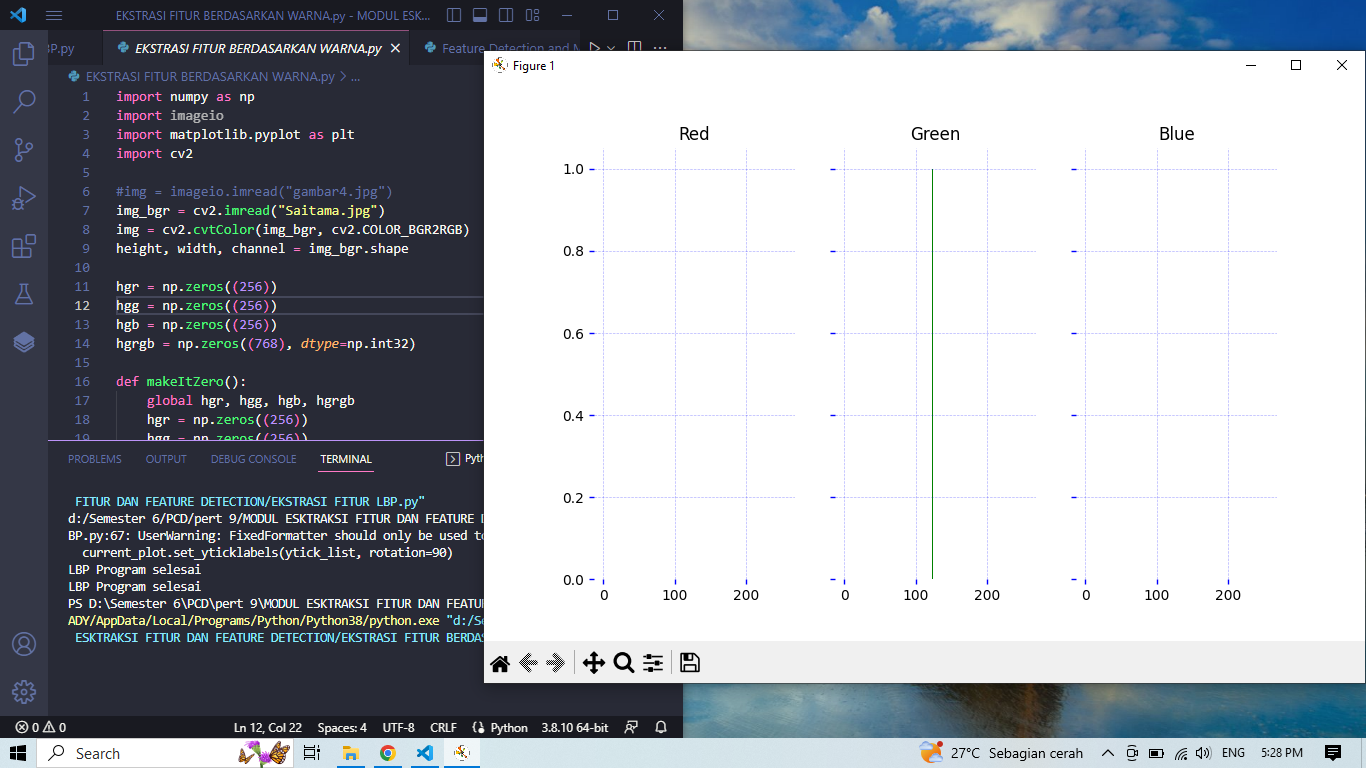
terdapat implementasi Local Binary Pattern (LBP) untuk menghasilkan gambar LBP dan histogramnya dari tiga gambar yang berbeda. Kode tersebut juga mencakup fungsi `brighter` yang digunakan untuk meningkatkan kecerahan gambar.

gambar pertama yang diproses adalah 'Satoru.jpeg'. Gambar tersebut diubah menjadi citra keabuan menggunakan fungsi `cv2.cvtColor()`,Hasilnya disimpan dalam citra LBP yang ditunjukkan oleh `img\_lbp`. Hasil akhir ditampilkan dalam bentuk empat gambar: gambar asli, gambar keabuan, gambar LBP, dan histogram LBP lalu gambar 2 diproses adalah 'Saitama.jpeg'. Gambar tersebut juga diubah menjadi citra keabuan dan ditingkatkan kecerahannya menggunakan fungsi `brighter()`. Langkah-langkah yang sama dilakukan seperti pada gambar sebelumnya, yaitu perhitungan LBP, histogram LBP, dan tampilan gambar asli, gambar keabuan, gambar LBP, serta histogram LBP dan melakukan perhitungan LBP, menghitung histogram LBP, dan menampilkan hasilnya dalam bentuk empat gambar dan mengimplementasikan metode LBP untuk menghasilkan gambar LBP dan histogramnya dari tiga gambar yang berbeda. Metode ini dapat digunakan untuk ekstraksi fitur dalam pengolahan citra, dan histogram LBP memberikan informasi tentang distribusi tekstur pada gambar. Dengan menggunakan fungsi `brighter()`, kecerahan gambar juga dapat ditingkatkan sebelum melakukan ekstraksi fitur.

* **EKSTRASI FITUR BERDASARKAN WARNA**



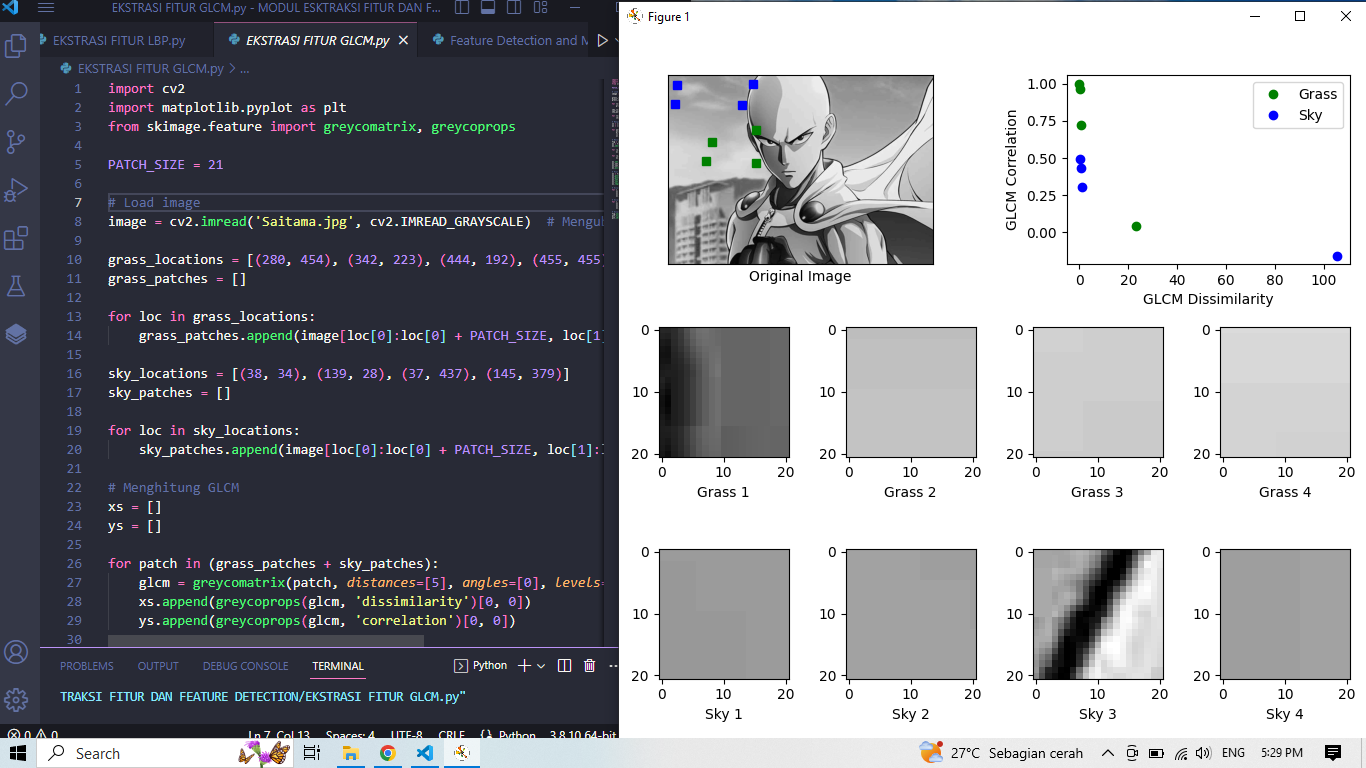




Analisis :

mengenai penghitungan dan visualisasi histogram citra berwarna menggunakan modul numpy, imageio, matplotlib.pyplot, dan cv2 (OpenCV)dan dilakukan beberapa langkah untuk menghasilkan histogram citra dan kemudian, dilakukan penghitungan histogram dengan mengiterasi setiap piksel citra dan menambahkan jumlah kemunculan nilai intensitas merah, hijau, dan biru ke dalam array histogram yang sesuai. Selain itu, juga dilakukan penghitungan histogram menggunakan fungsi cv2.calcHist dari modul cv2. Setelah selesai, histogram divisualisasikan menggunakan fungsi plt.hist dan plt.plot untuk memberikan gambaran tentang distribusi intensitas piksel dan komposisi warna dalam citra. Seluruh proses ini digambarkan dengan jelas melalui fungsi-fungsi seperti makeItZero, plot\_result, dan pemanggilan fungsi-fungsi visualisasi histogram dan memberikan kemudahan menghitung dan memvisualisasikan histogram citra berwarna, sehingga pengguna dapat menganalisis dan memahami karakteristik citra secara lebih mendalam.

* **EKSTRASI FITUR GLCM**



Analisis:

mengimplementasikan fitur Grey Level Co-occurrence Matrix (GLCM) pada citra grayscale. GLCM adalah sebuah metode yang digunakan untuk menggambarkan hubungan spasial antara pasangan nilai intensitas piksel dalam citra.

citra grayscale dibaca menggunakan cv2.imread dan disimpan dalam variabel image. Kemudian, dilakukan inisialisasi lokasi-lokasi patch yang akan diekstrak dari citra untuk mewakili daerah rumput (grass\_locations) dan langit (sky\_locations).

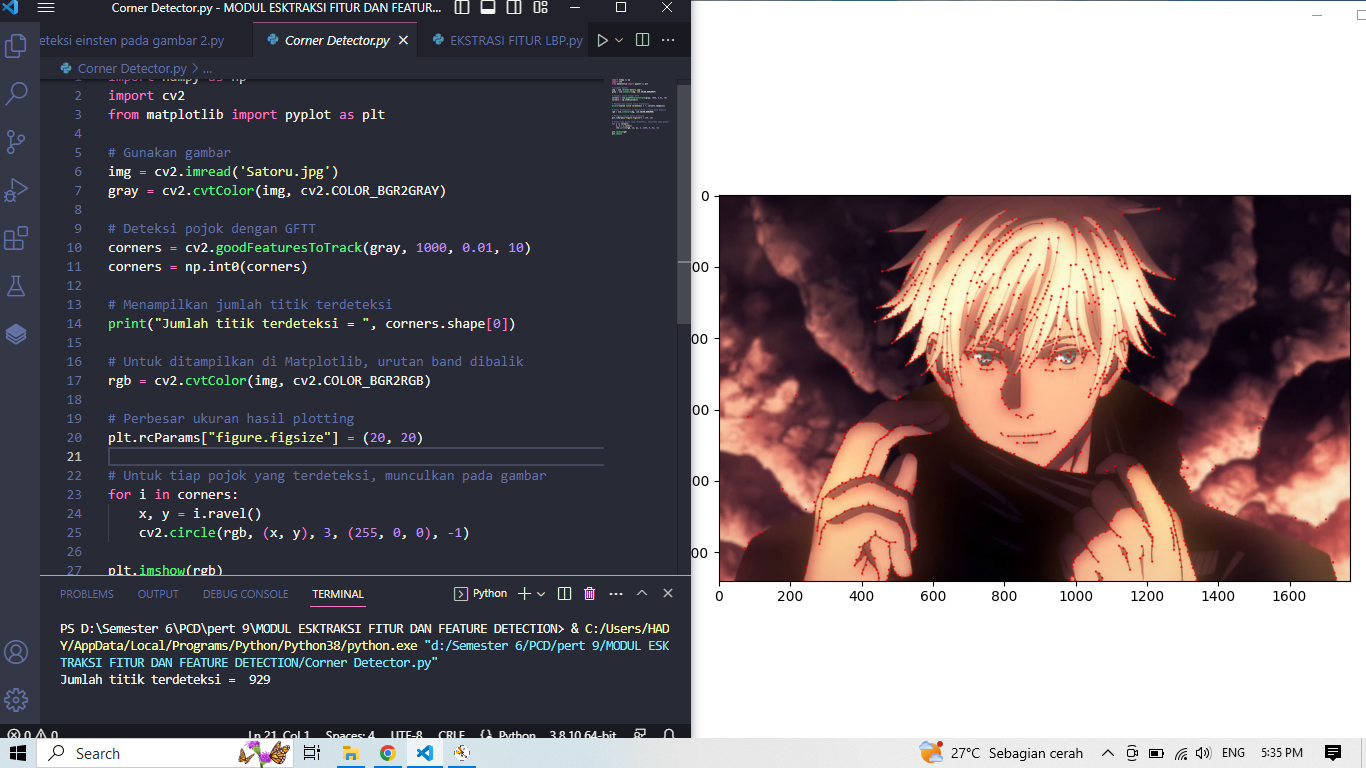
Selanjutnya, dilakukan perhitungan GLCM pada setiap patch menggunakan fungsi greycomatrix dari modul skimage.feature. Pada tahap ini, jarak (distances), sudut (angles), level (levels), simetris (symmetric), dan normalisasi (normed) GLCM dapat diatur sesuai kebutuhan. Kemudian, nilai fitur dissimilarity dan correlation dari GLCM dihitung menggunakan fungsi greycoprops dan disimpan dalam list xs dan ys untuk masing-masing patch.

Setelah selesai menghitung fitur GLCM, dilakukan visualisasi hasilnya. Gambar asli ditampilkan dalam subplot pertama dengan menandai lokasi patch rumput dan langit menggunakan simbol 'gs' dan 'bs'. Plot kedua menampilkan scatter plot fitur dissimilarity dan correlation untuk setiap patch, dengan warna hijau ('go') untuk patch rumput dan biru ('bo') untuk patch langit.

Selanjutnya, dilakukan tampilan visualisasi patch rumput dan patch langit dalam subplot terpisah. Setiap patch ditampilkan sebagai citra grayscale dengan keterangan 'Grass' atau 'Sky' di bawahnya.

Dengan menggunakan GLCM, source code tersebut menggambarkan hubungan dan pola spasial antara piksel dalam citra grayscale, dengan tujuan untuk membedakan antara daerah rumput dan langit.

* **Corner Detector**



Analisis

melakukan deteksi pojok pada sebuah gambar menggunakan metode Good Features to Track (GFTT) dan menampilkan jumlah titik terdeteksi serta menandai titik-titik tersebut pada gambar.

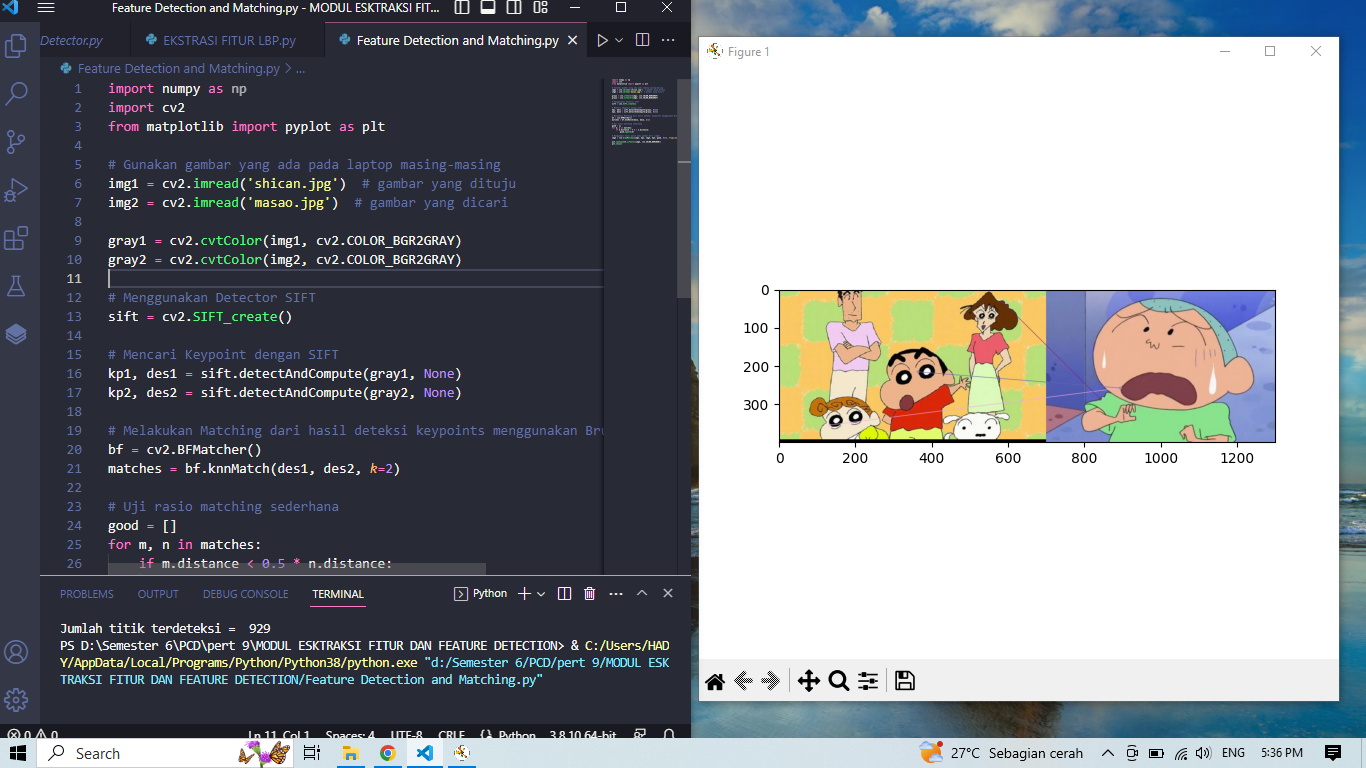
Pertama, gambar dibaca menggunakan cv2.imread dan diubah menjadi citra grayscale menggunakan cv2.cvtColor dengan mode cv2.COLOR\_BGR2GRAY. Citra grayscale ini akan digunakan untuk deteksi pojok.

Selanjutnya, deteksi pojok dilakukan menggunakan cv2.goodFeaturesToTrack. Metode ini menerima parameter citra grayscale (gray), jumlah maksimum titik terdeteksi (1000 dalam contoh ini), quality level (0.01), dan jarak minimum antara titik-titik terdekat (10). Hasil deteksi pojok disimpan dalam variabel corners.

Setelah itu, jumlah titik terdeteksi ditampilkan menggunakan corners.shape[0]. Hal ini memberikan informasi tentang berapa banyak titik pojok yang berhasil dideteksi pada gambar.

Kemudian, untuk setiap titik pojok yang terdeteksi, dilakukan pengulangan menggunakan loop for. Setiap titik koordinat (x, y) diambil dari corners dan digambar dengan fungsi cv2.circle menggunakan warna biru (255, 0, 0) dan ketebalan -1 yang berarti mengisi lingkaran. Titik-titik ini ditandai pada citra asli yang sudah dikonversi menjadi mode RGB.

* **Feature Detection and Matching**



Analisis:

dilakukan pencocokan atau matching antara dua gambar menggunakan algoritma SIFT (Scale-Invariant Feature Transform).

Gambar pertama (`img1`) dan gambar kedua (`img2`) dibaca menggunakan `cv2.imread` dan diubah menjadi citra grayscale menggunakan `cv2.cvtColor` dengan mode `cv2.COLOR\_BGR2GRAY`. Citra grayscale ini akan digunakan untuk deteksi keypoint.

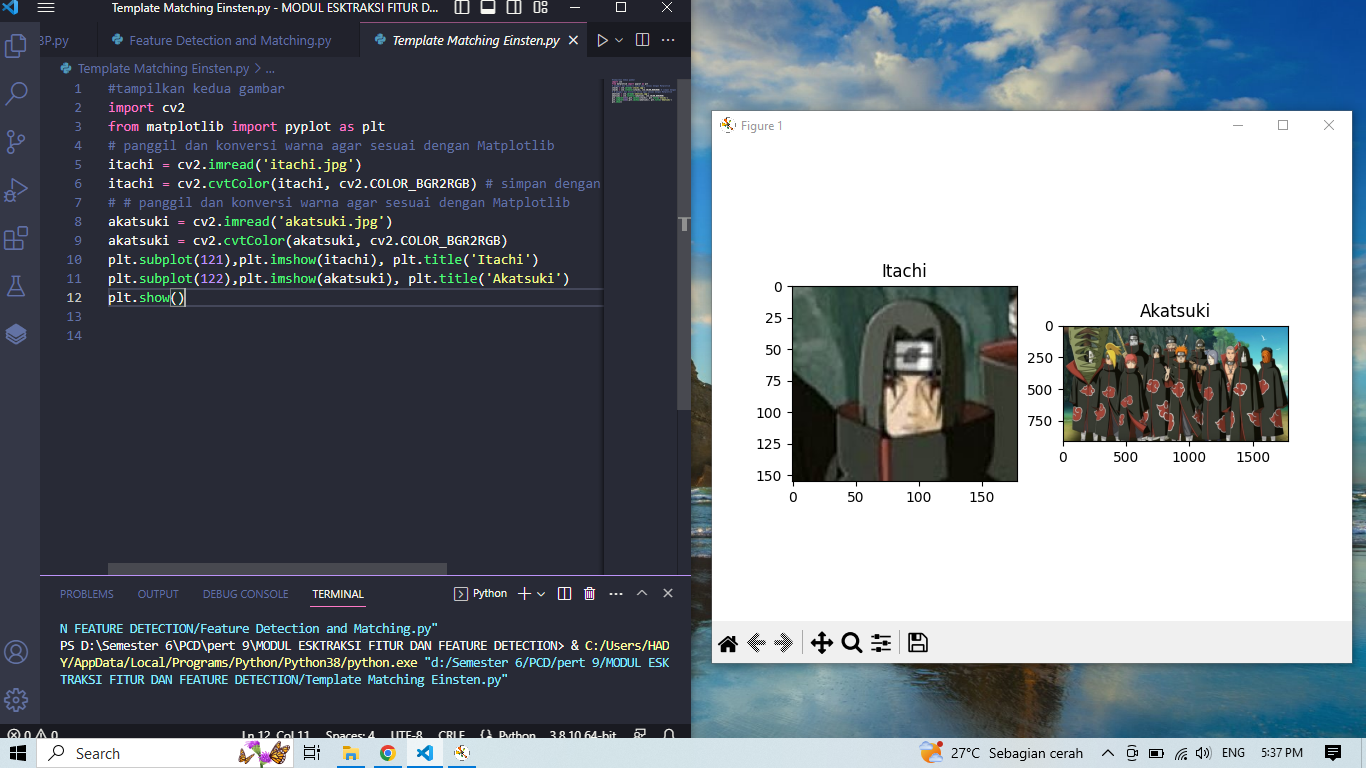
Selanjutnya, objek detektor SIFT dibuat menggunakan `cv2.SIFT\_create()`. Detektor ini akan digunakan untuk mendeteksi keypoint pada kedua citra.

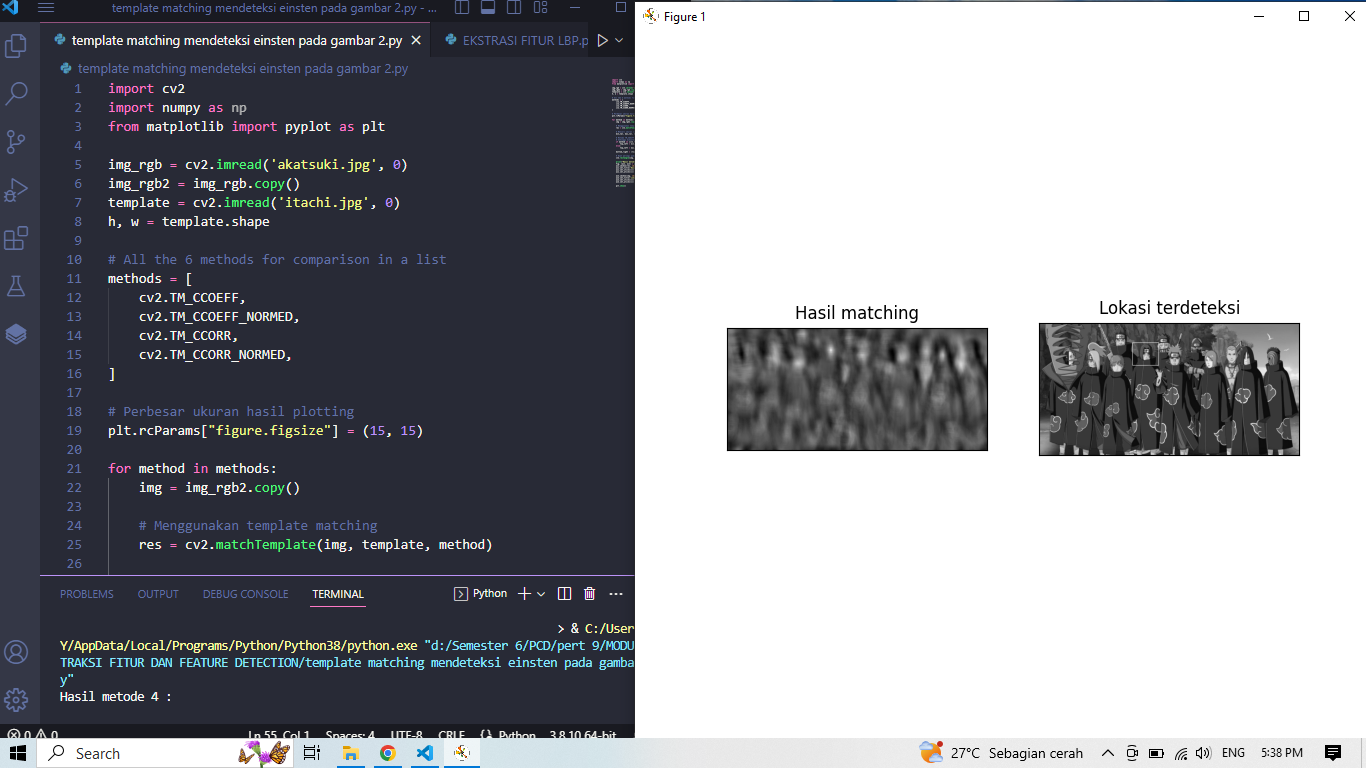
Kemudian, dilakukan deteksi dan komputasi keypoints serta deskriptor menggunakan metode `detectAndCompute` dari detektor SIFT. Citra grayscale (`gray1` dan `gray2`) digunakan sebagai input, dan keypoints (`kp1` dan `kp2`) serta deskriptor (`des1` dan `des2`) dihasilkan.

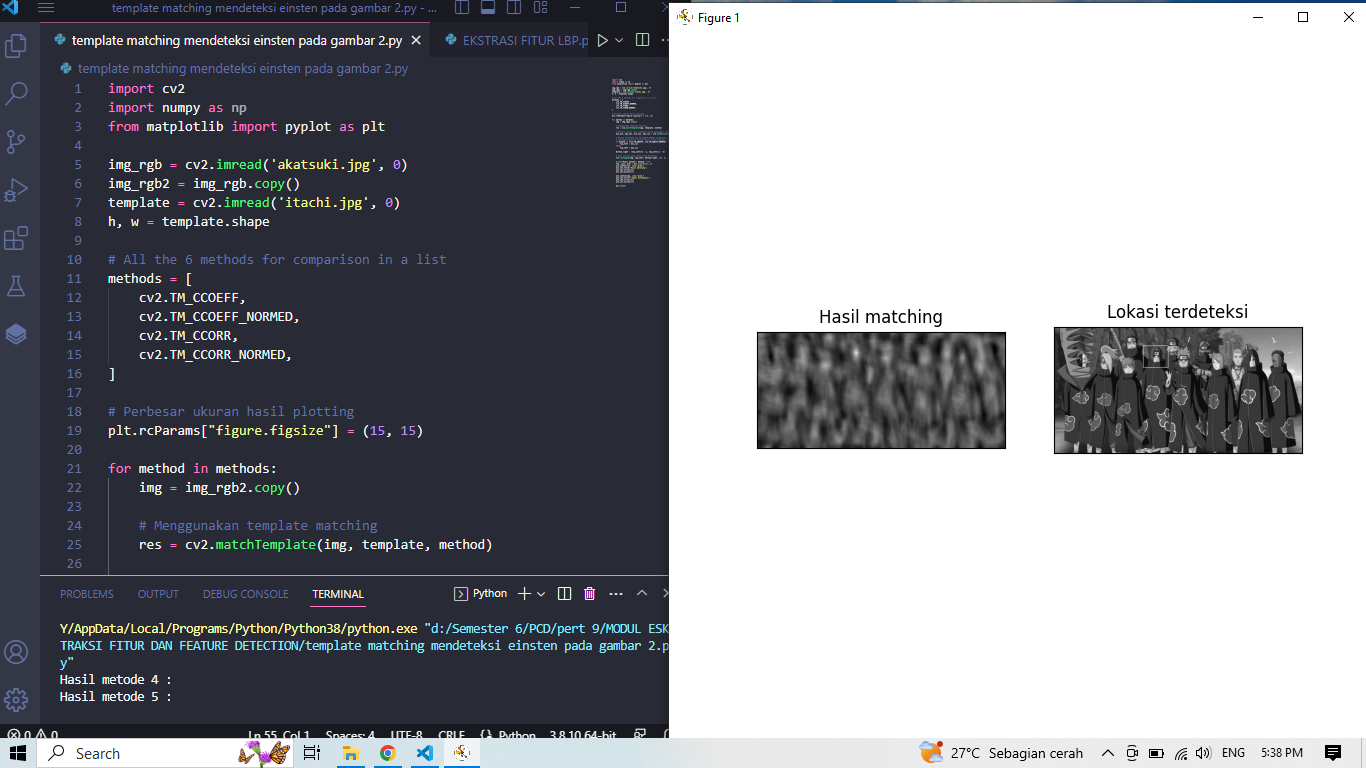
Selanjutnya, dilakukan uji rasio matching sederhana dengan membandingkan jarak Euclidean antara dua tetangga terdekat untuk setiap pasangan keypoints. Jika jarak Euclidean terdekat (`m.distance`) lebih kecil dari 0.5 kali jarak Euclidean kedua terdekat (`n.distance`), maka pasangan tersebut dianggap cocok dan dimasukkan ke dalam list `good`.

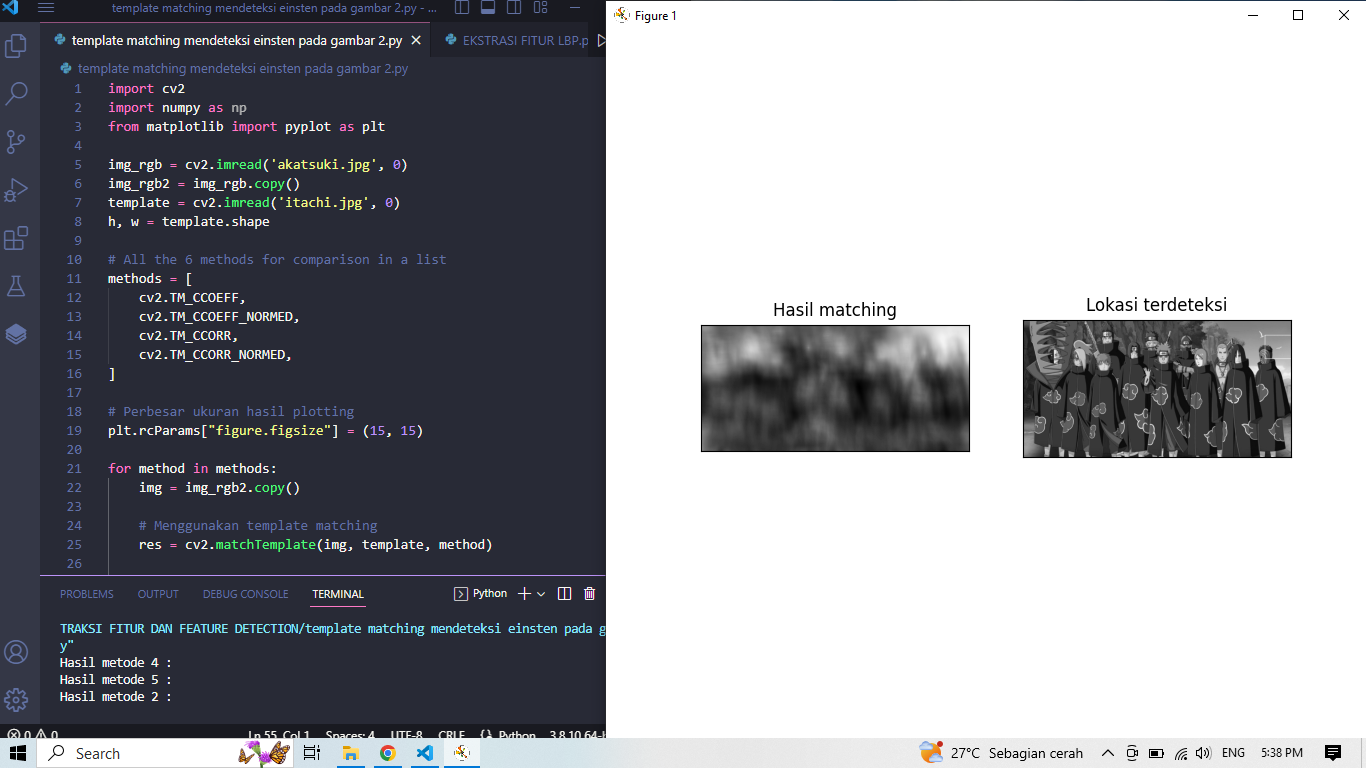
Terakhir, dilakukan penggambaran hasil pencocokan pada gambar baru (`img3`) menggunakan `cv2.drawMatches`. Keypoints yang cocok dari `img1` dan `img2` (yang terdapat dalam list `good`) ditandai pada kedua gambar dan hasilnya disimpan dalam `img3`. Citra `img3` kemudian ditampilkan menggunakan `plt.imshow` setelah diubah mode warna dari BGR ke RGB menggunakan `cv2.cvtColor` dan akan menampilkan gambar `img3` dengan menampilkan titik-titik keypoints yang cocok antara kedua gambar tersebut.

* **Matching mendeteksi seseorang pada gambar.**











Analisis:

dilakukan proses template matching antara sebuah template (gambar Itachi) dan citra target (gambar Akatsuki) menggunakan berbagai metode yang tersedia. Template matching digunakan untuk mencari kecocokan atau kesamaan pola antara template dan citra target. Setelah template matching dilakukan, dilakukan visualisasi hasil dengan menampilkan citra grayscale yang menunjukkan tingkat kecocokan antara template dan citra target, serta citra target dengan kotak yang menandai lokasi terdeteksi.

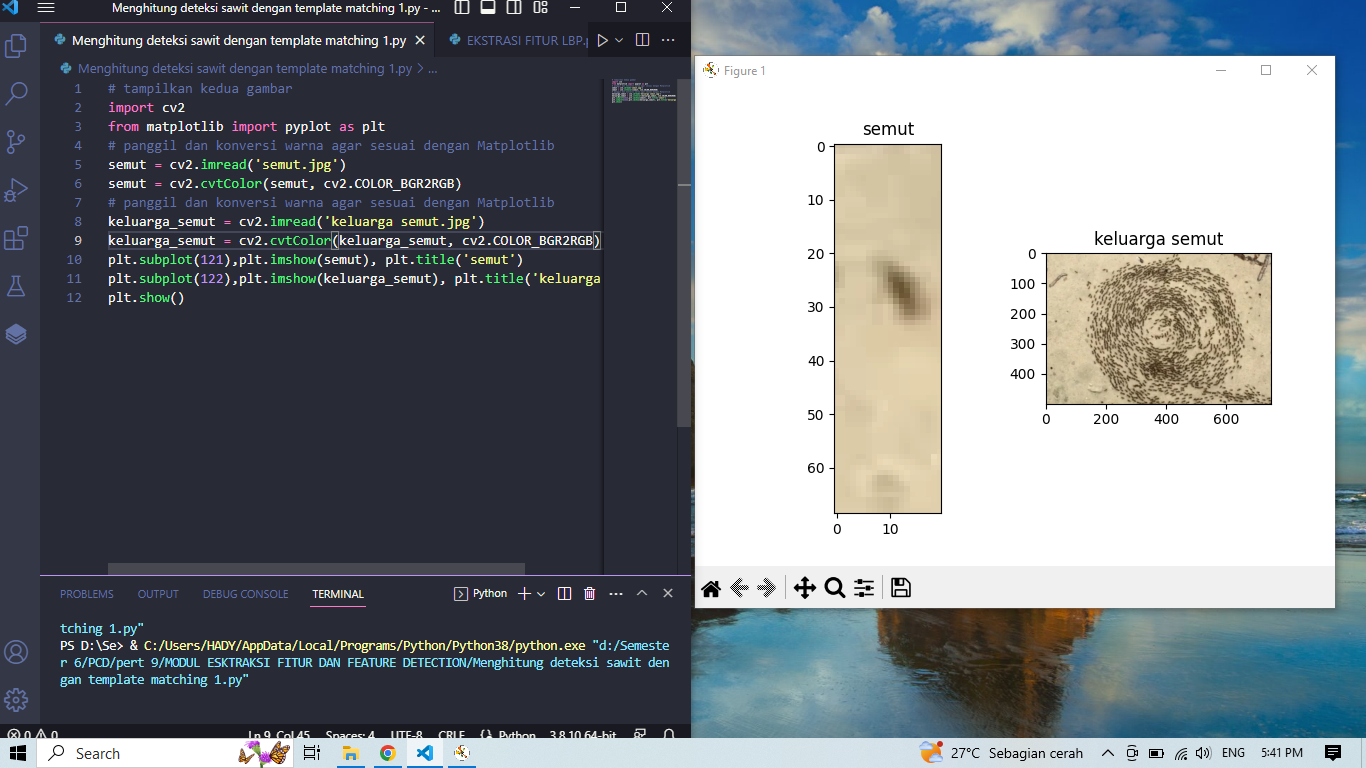
Langkah pertama membaca gambar target dan template menggunakan `cv2.imread` dengan mode grayscale. Hal ini dilakukan agar perhitungan template matching dapat dilakukan dengan lebih efisien. Selanjutnya, citra target diubah menjadi citra grayscale menggunakan `cv2.cvtColor` untuk persiapan perhitungan.

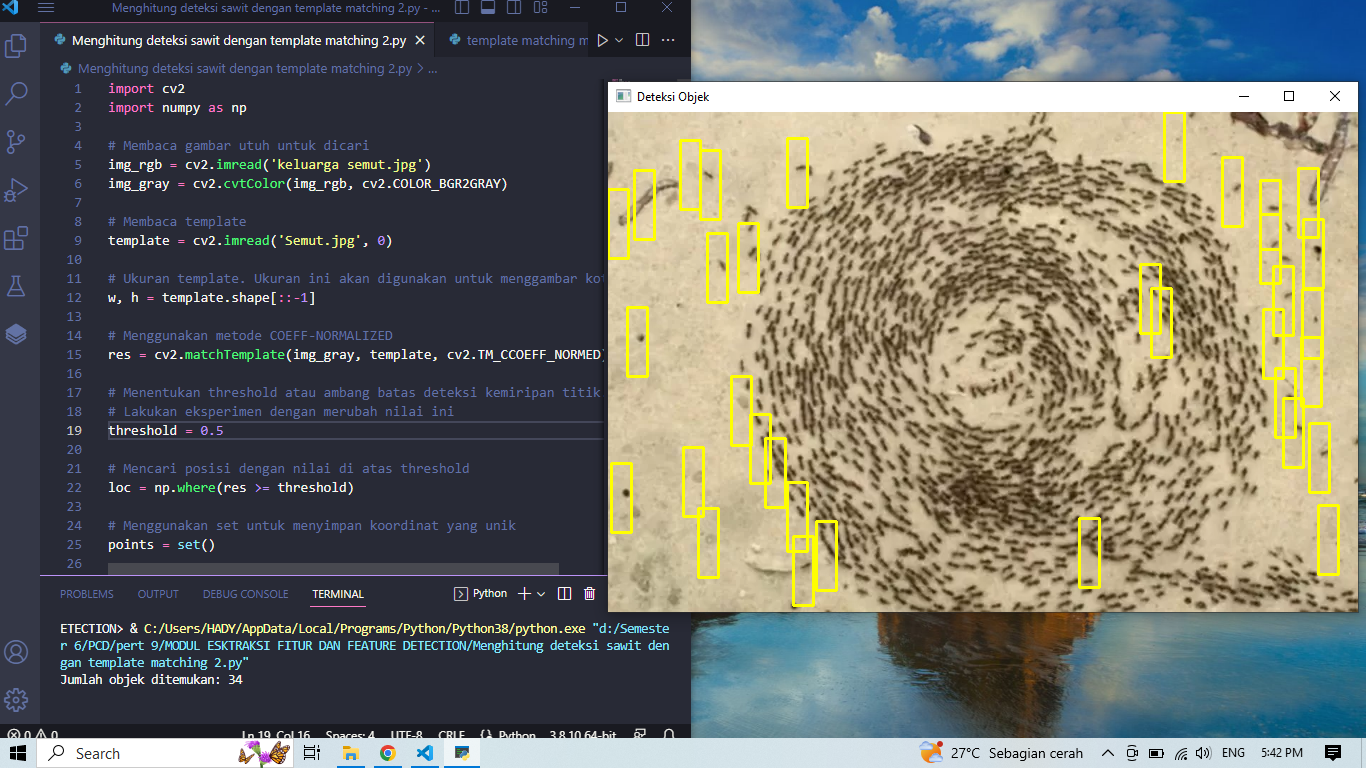
Proses template matching dilakukan dengan iterasi menggunakan berbagai metode yang telah ditentukan.dan dalam setiap iterasi, citra target disalin untuk visualisasi, kemudian template matching dilakukan dengan menggunakan `cv2.matchTemplate`. Hasil pencocokan kemudian dianalisis dengan mencari posisi dan nilai minimum atau maksimum menggunakan `cv2.minMaxLoc`, tergantung dari metode yang digunakan. Kotak lokasi terdeteksi kemudian digambar pada citra target menggunakan `cv2.rectangle`.

Hasil dari template matching kemudian ditampilkan dalam bentuk plot dengan menggunakan `plt.subplot` dan `plt.imshow`dan terdapat dua plot yang ditampilkan untuk setiap metode, yaitu plot pertama menunjukkan tingkat kecocokan dalam bentuk citra grayscale, sedangkan plot kedua menampilkan citra target dengan kotak yang menandai lokasi yang terdeteksi.

bahwa template matching dengan berbagai metode telah dilakukan untuk mencari kecocokan antara template dan citra target serta hasil pencocokan ditampilkan dengan visualisasi yang memperlihatkan tingkat kecocokan dan lokasi terdeteksi pada citra target. Teknik ini dapat berguna dalam berbagai aplikasi seperti deteksi objek, pencocokan pola, atau pencarian kecocokan visual antara gambar-gambar.

* **Menghitung deteksi objek dengan template matching**





Analisis:

dilakukan proses deteksi objek pada gambar utuh menggunakan template matching. Gambar utuh (img\_rgb) dibaca dan diubah menjadi citra grayscale (img\_gray) untuk mempermudah perhitungan. Selanjutnya, template yang akan digunakan untuk deteksi (Semut.jpg) juga dibaca.

Proses template matching dilakukan dengan menggunakan metode COEFF-NORMALIZED (`cv2.TM\_CCOEFF\_NORMED`) serta hasil pencocokan disimpan dalam variabel "res". Selanjutnya, ditentukan threshold (ambang batas) untuk deteksi kemiripan titik. Nilai threshold yang ditentukan pada source code ini adalah 0.5, namun dapat dilakukan eksperimen dengan mengubah nilai tersebut.

Koordinat lokasi yang memiliki nilai di atas threshold ditentukan dengan menggunakan `np.where`. Kemudian, dilakukan pengolahan untuk menghindari deteksi duplikat objek yang berdekatan. Dalam source code ini, dilakukan pengecekan dengan membandingkan jarak antara koordinat objek yang baru dengan koordinat objek yang telah ditemukan sebelumnya. Jika jaraknya kurang dari suatu radius (diatur dalam variabel `ignore\_radius`), maka objek dianggap sebagai duplikat dan diabaikan. Jika tidak, persegi dengan warna kuning dan ketebalan dua poin akan digambar pada posisi objek yang terdeteksi dan bahwa deteksi objek menggunakan template matching telah dilakukan pada gambar utuh. Objek yang memiliki kemiripan dengan template ditandai dengan persegi kuning. Pengolahan dilakukan untuk menghindari duplikat objek yang berdekatan. Jumlah objek yang berhasil dideteksi juga ditampilkan. Teknik ini dapat berguna dalam berbagai aplikasi seperti deteksi wajah, deteksi objek dalam citra, atau pencocokan pola visual antara template dan citra target.

Link Github:https://github.com/HaddyFirdaus/1207070050\_Haddy-Firdaus\_Esktrasi-Fitur-dan-Feature-Detection