SchreenShoot Hasil Latihan Praktikum 10

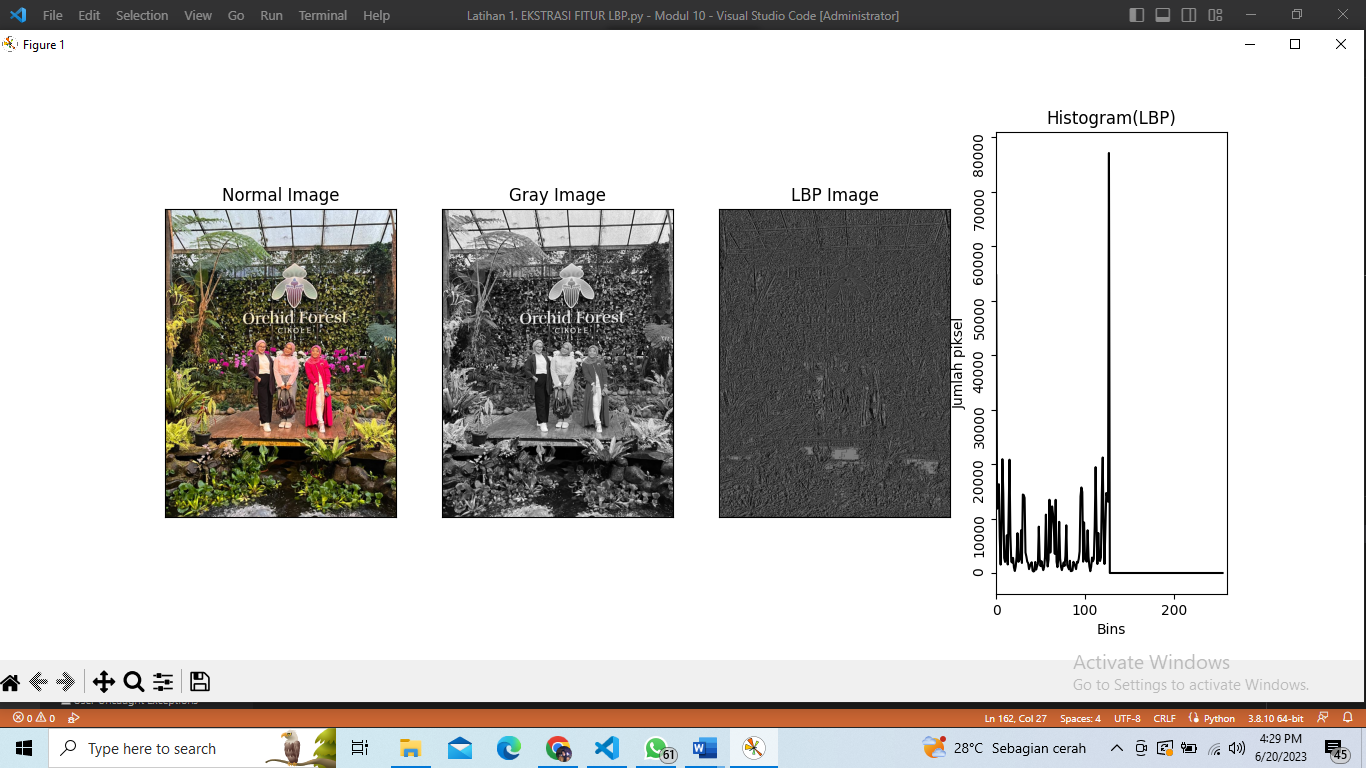
*Praktikum Citra Digital*

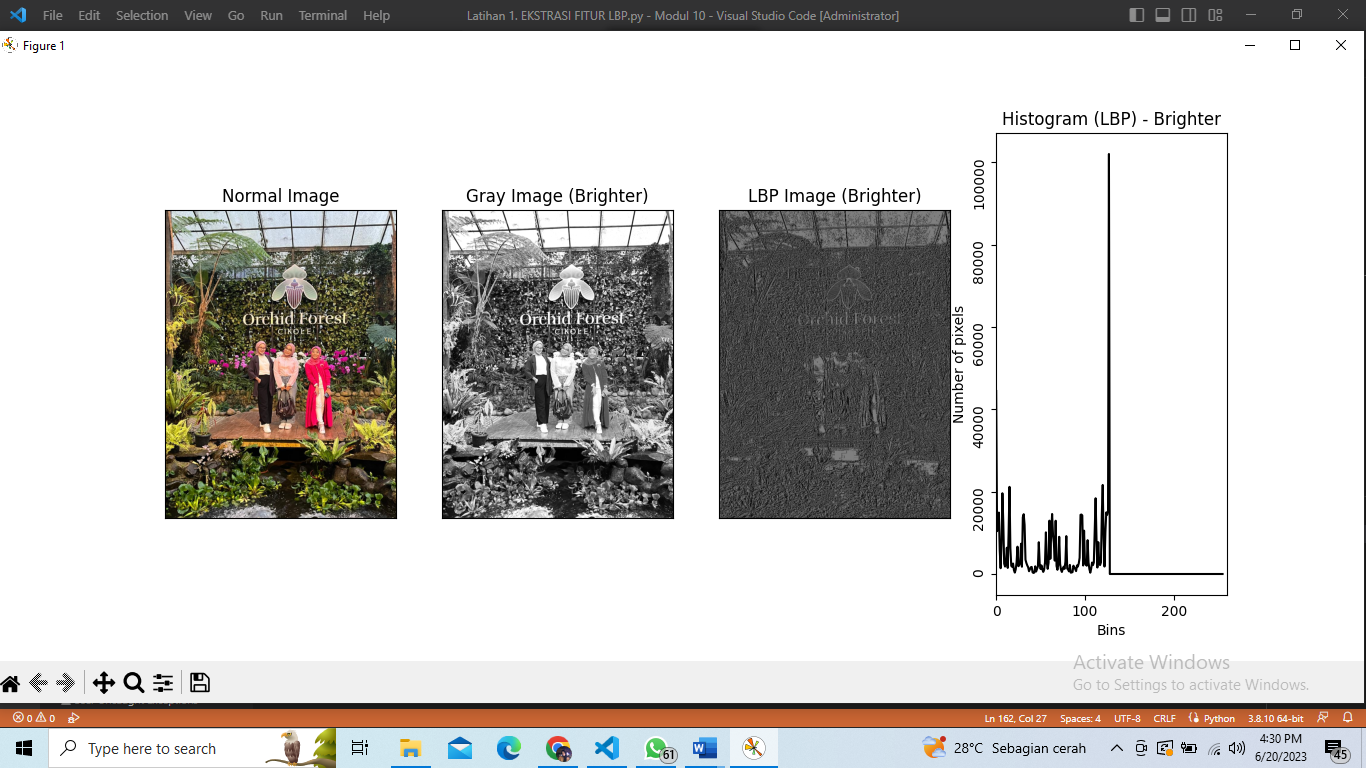
Nama : Shifa Mutia Fauziah

NIM : 1207070114

Kelas : TKK

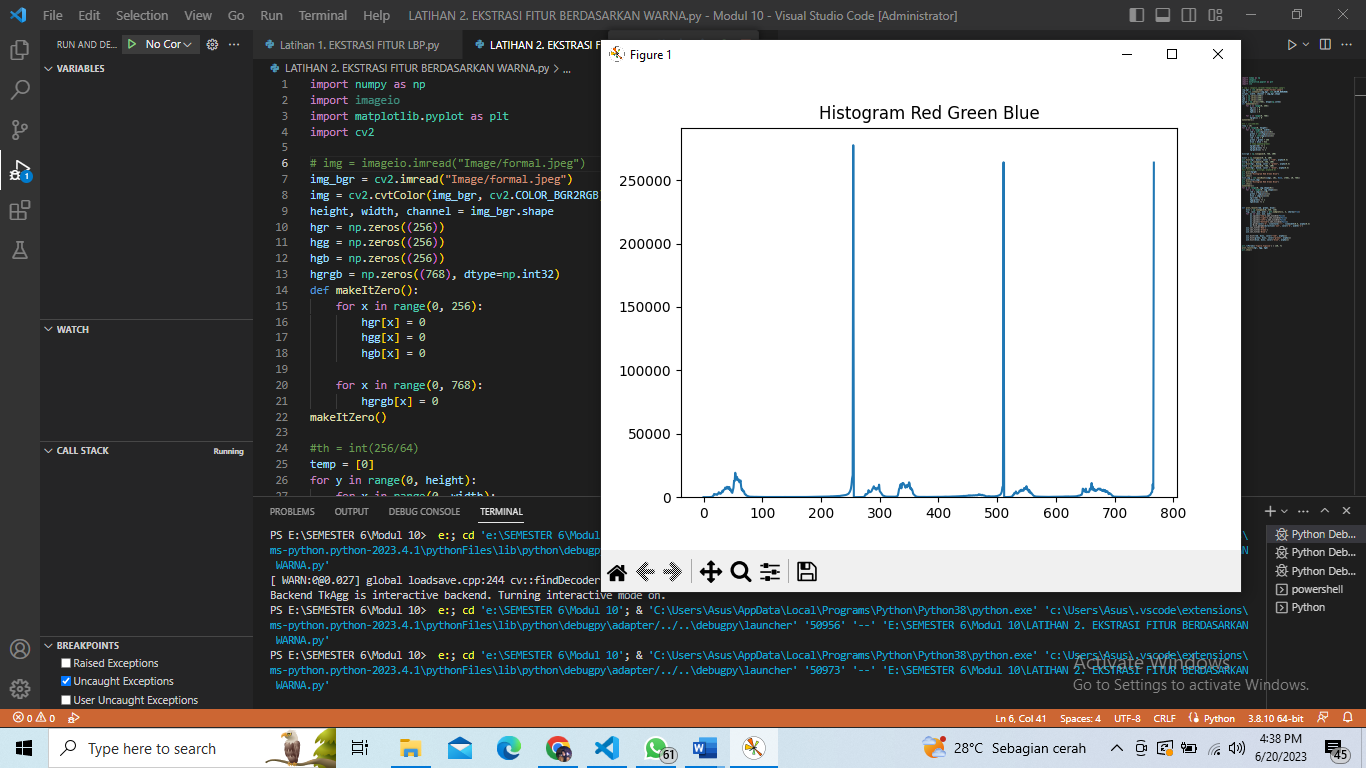
# EKSTRASI FITUR LBP

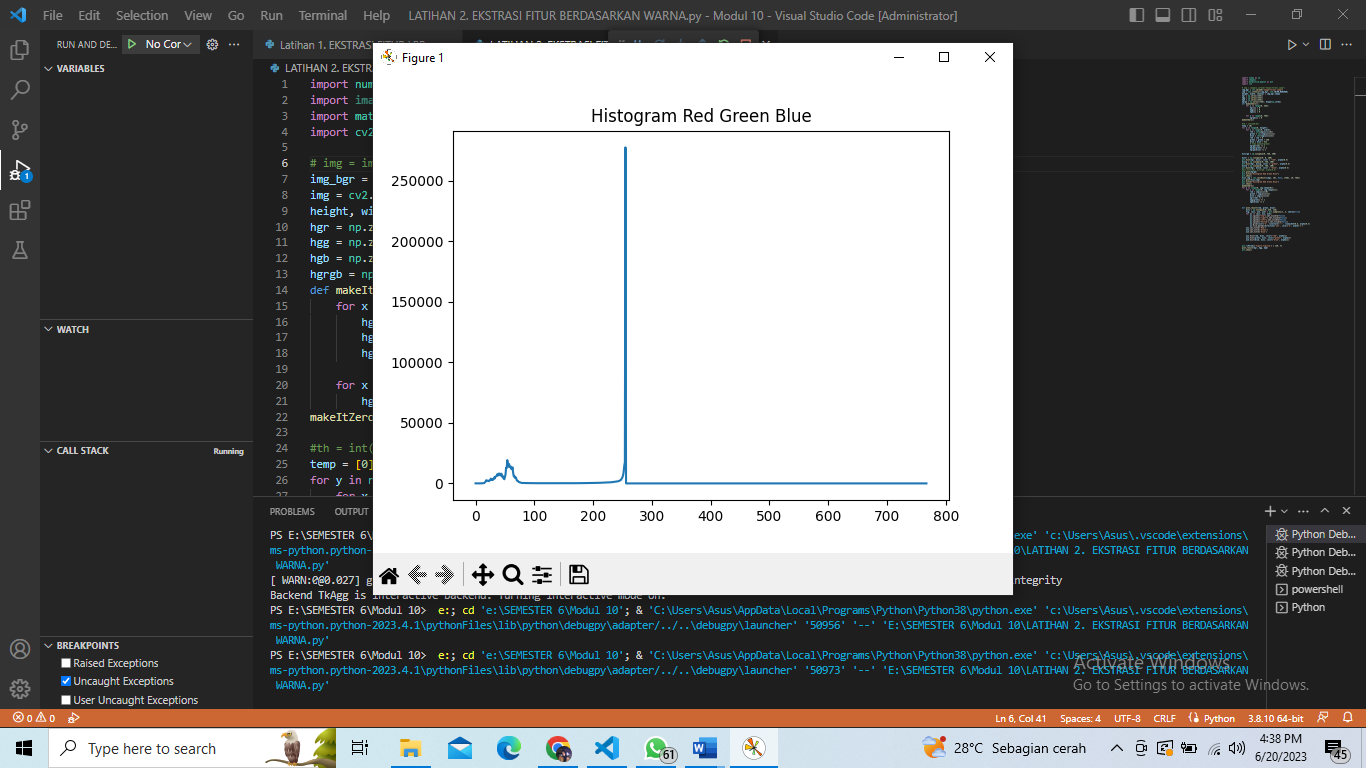


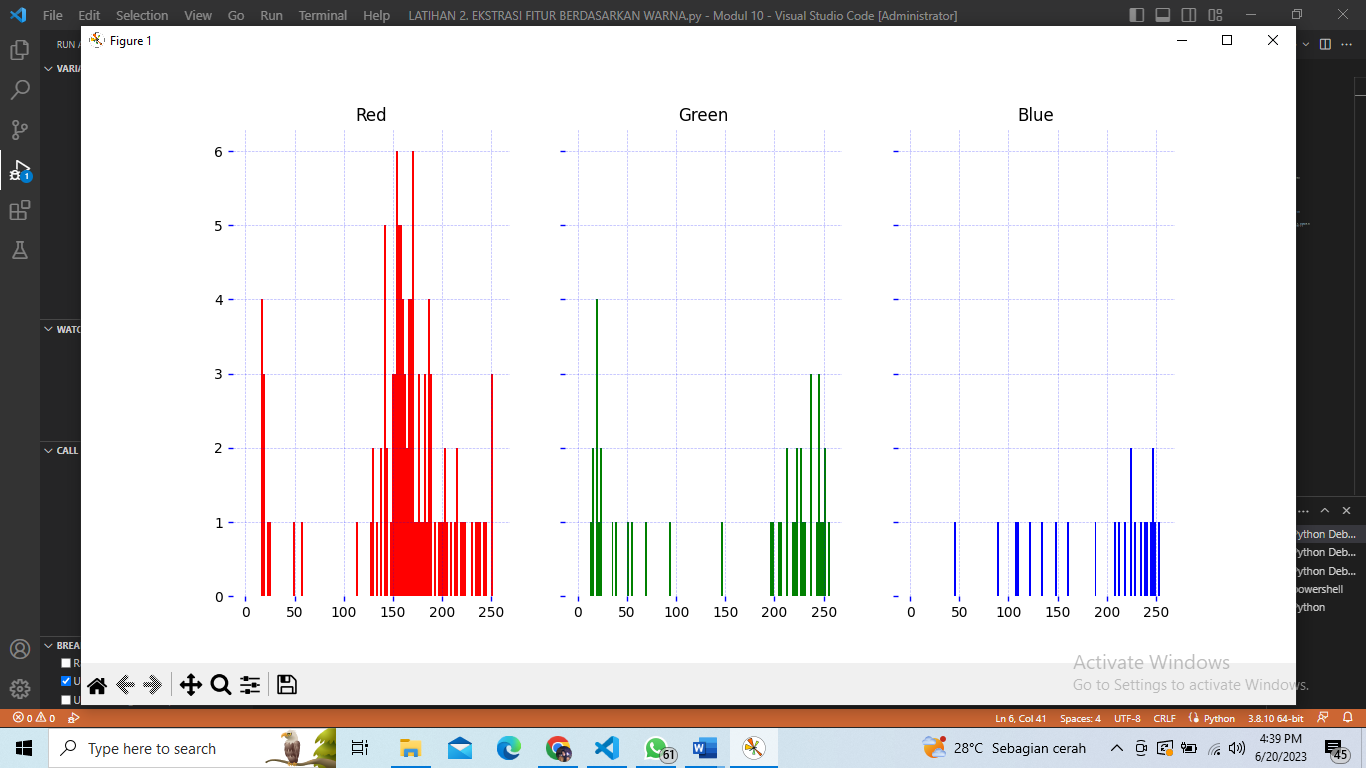


Pada percobaan pertama ini yaitu ekstrasi fitur Local Binary Patterns (LBP). LBP adalah metode yang digunakan untuk mengekstraksi tekstur lokal dari sebuah gambar dengan membandingkan setiap piksel dengan tetangganya. Proses dimulai dengan membaca gambar input dan mengubahnya menjadi gambar keabuan (grayscale). 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. Dalam contoh kode di atas, analisis LBP dilakukan pada dua gambar: gambar asli,, dan gambar kedua yang dicerahkan. Output dari gambar pertmana ini mencakup gambar-gambar tersebut dalam skala keabuan dan hasil LBP serta histogram LBP, kemudian Output dari gambar kedua ini hampir sama dengan yang pertama yaitu mencakup gambar-gambar tersebut dalam skala keabuan tetapi dicerahkan dan hasil LBP serta histogram LBP untuk masing-masing gambar.

1. EKSTRASI FITUR BERDASARKAN WARNA

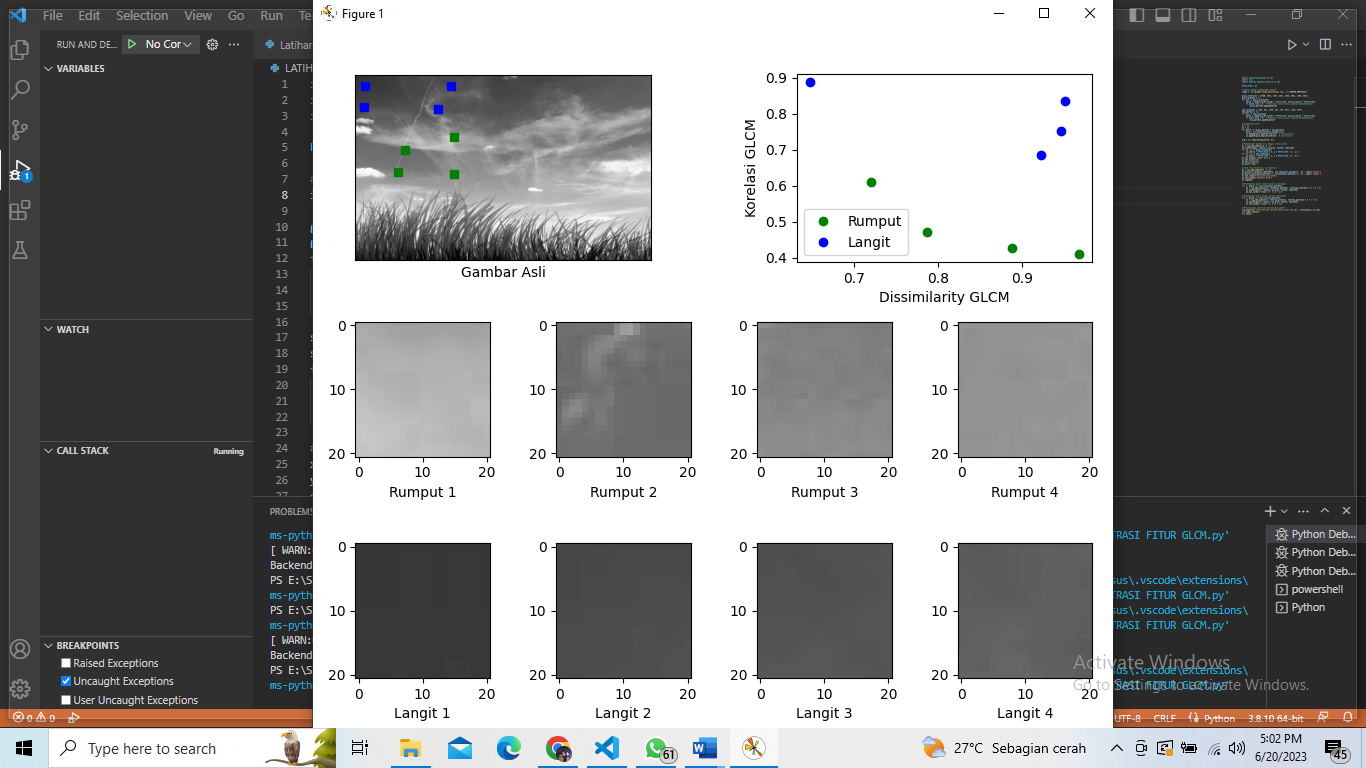






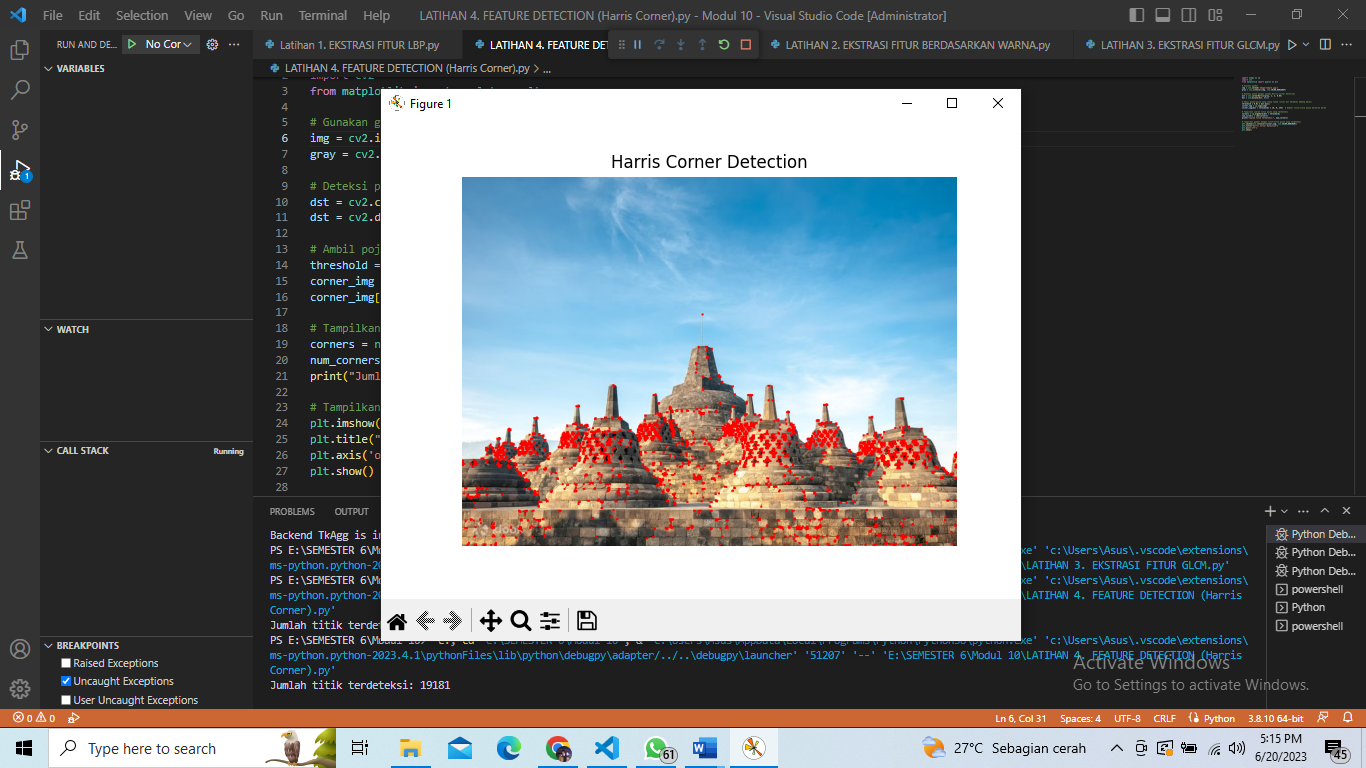
Percobaan kedua adalah Ekstrasi Fitur Berdasarkan Warna. Kodingan yang dipakai pada percobaan ini mengimplementasikan metode Local Binary Pattern (LBP) untuk melakukan deskripsi tekstur pada suatu citra. Fungsi `get\_pixel` digunakan untuk membandingkan nilai piksel sekitar pusat dengan nilai piksel 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. Kemudian 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



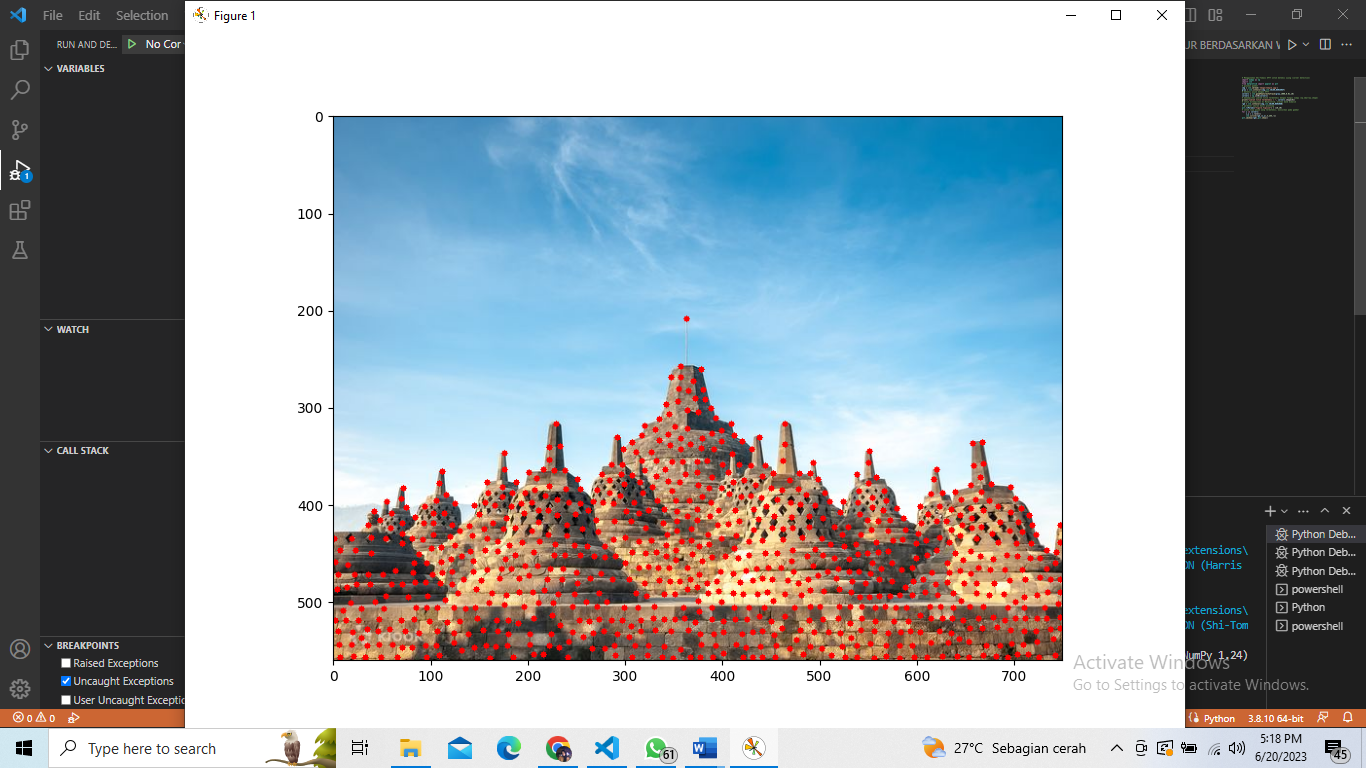
Percobaan ketiga menggunakan ekstrasi fitur GLCM. Ekstraksi fitur GLCM (Grey-Level Co-occurrence Matrix) adalah proses untuk mendapatkan informasi statistik dari citra berdasarkan pola kemunculan intensitas piksel yang berdekatan. GLCM dapat memberikan informasi tentang tekstur dan struktur citra. Pada program ini dilakukan perulangan untuk setiap patch yang telah ditentukan. Pada setiap iterasi, dilakukan perhitungan GLCM. 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



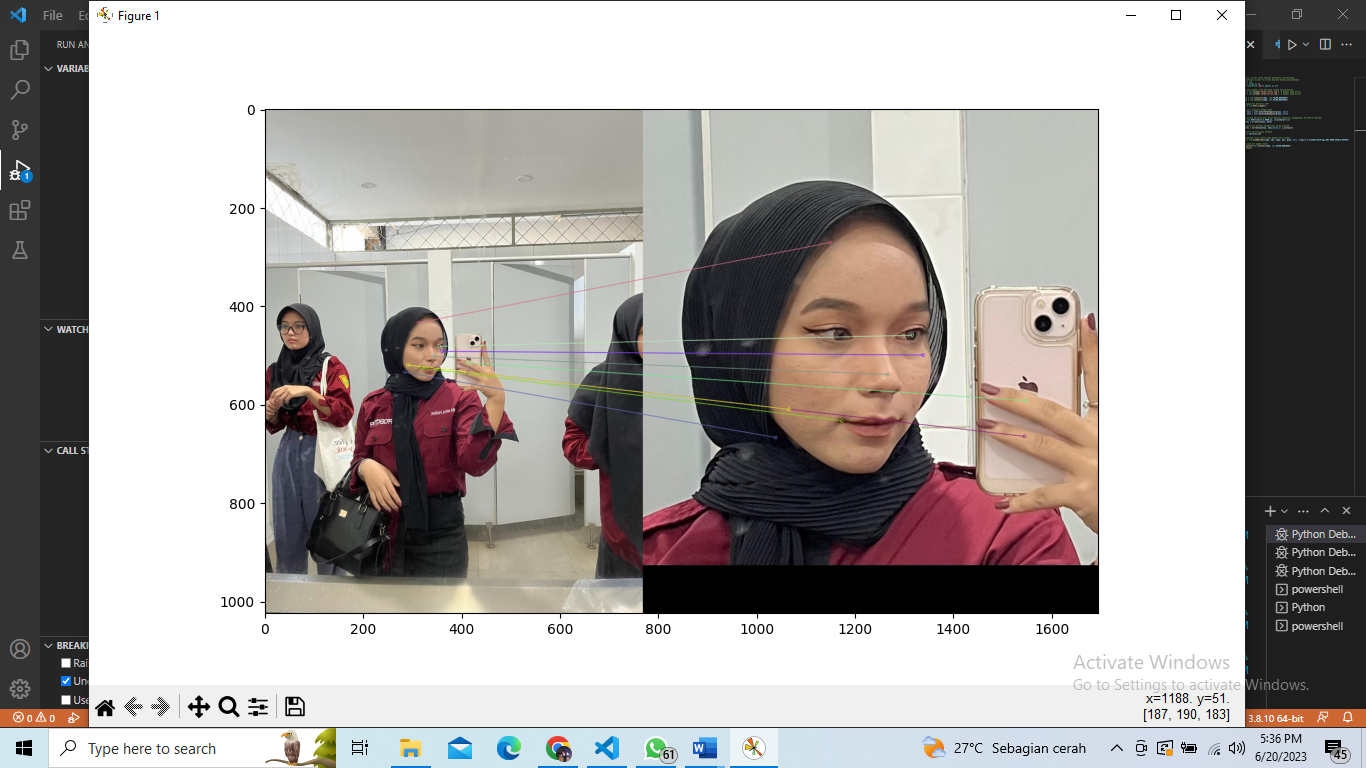
Percobaan keempat yaitu feature detection dengan Harris Corner. fitur ini melakukan deteksi sudut pada gambar menggunakan metode Harris Corner Detection. Metode ini membantu mengidentifikasi sudut-sudut penting dalam gambar yang dapat digunakan untuk berbagai tujuan, seperti pelacakan objek, pengenalan 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: 1918.

1. Dengan Shi-Tomasi



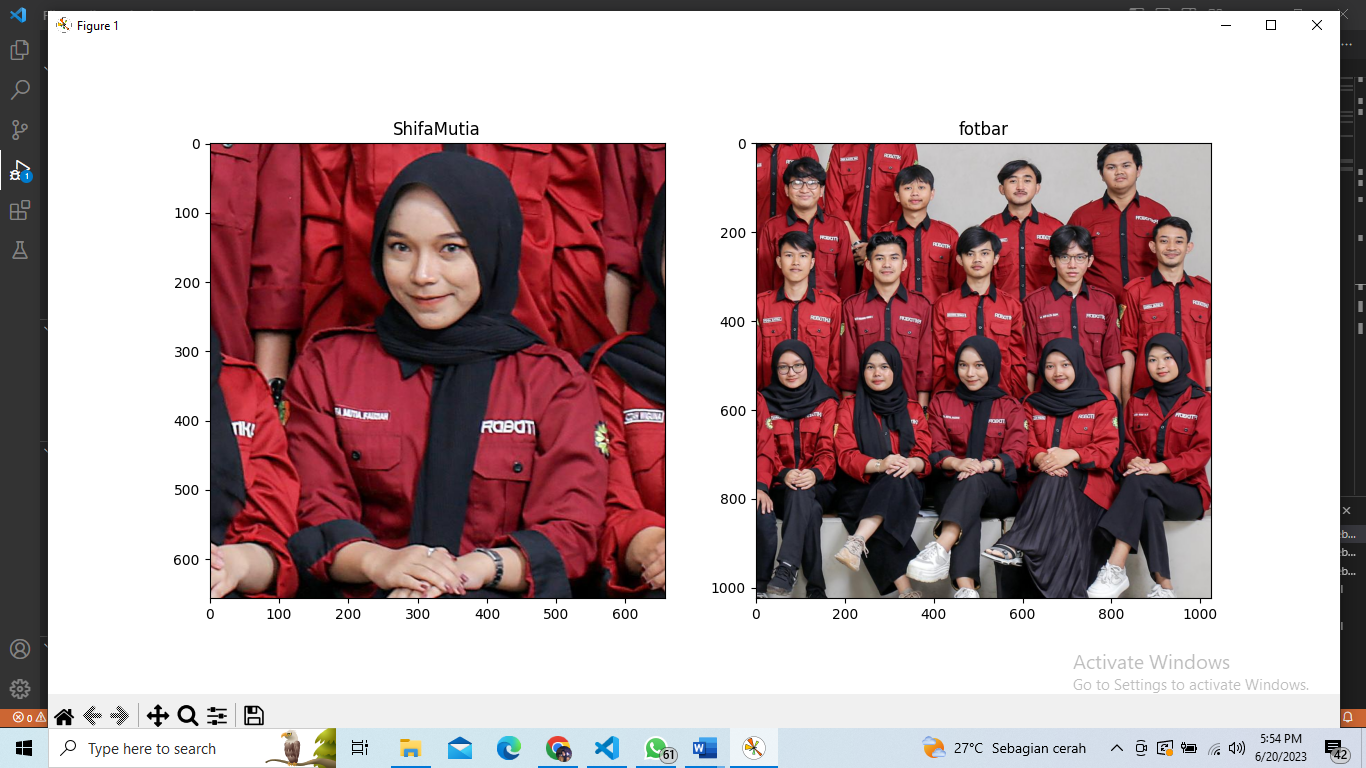
Percobaan keempat yaitu feature detection dengan Shi-Tomasi. Metode ini dapat digunakan untuk menemukan titik sudut penting yang berpotensi sebagai fitur dalam berbagai aplikasi pengolahan gambar, seperti pelacakan objek, deteksi objek, atau pencocokan pola. Hasil deteksi sudut ditampilkan pada gambar 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 840 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. Feature Detection and Matching



Percobaan kelima yaitu Feature Detection and Matching. fitur menggunakan algoritma SIFT (Scale-Invariant Feature Transform). Pertama, dua gambar dimasukkan sebagai input, yaitu ` Image/shifa1.jpg` yang merupakan gambar yang dituju, dan 'Image/shifa2.jpg' 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.

1. Template matching mendeteksi ShifaMutia pada gambar

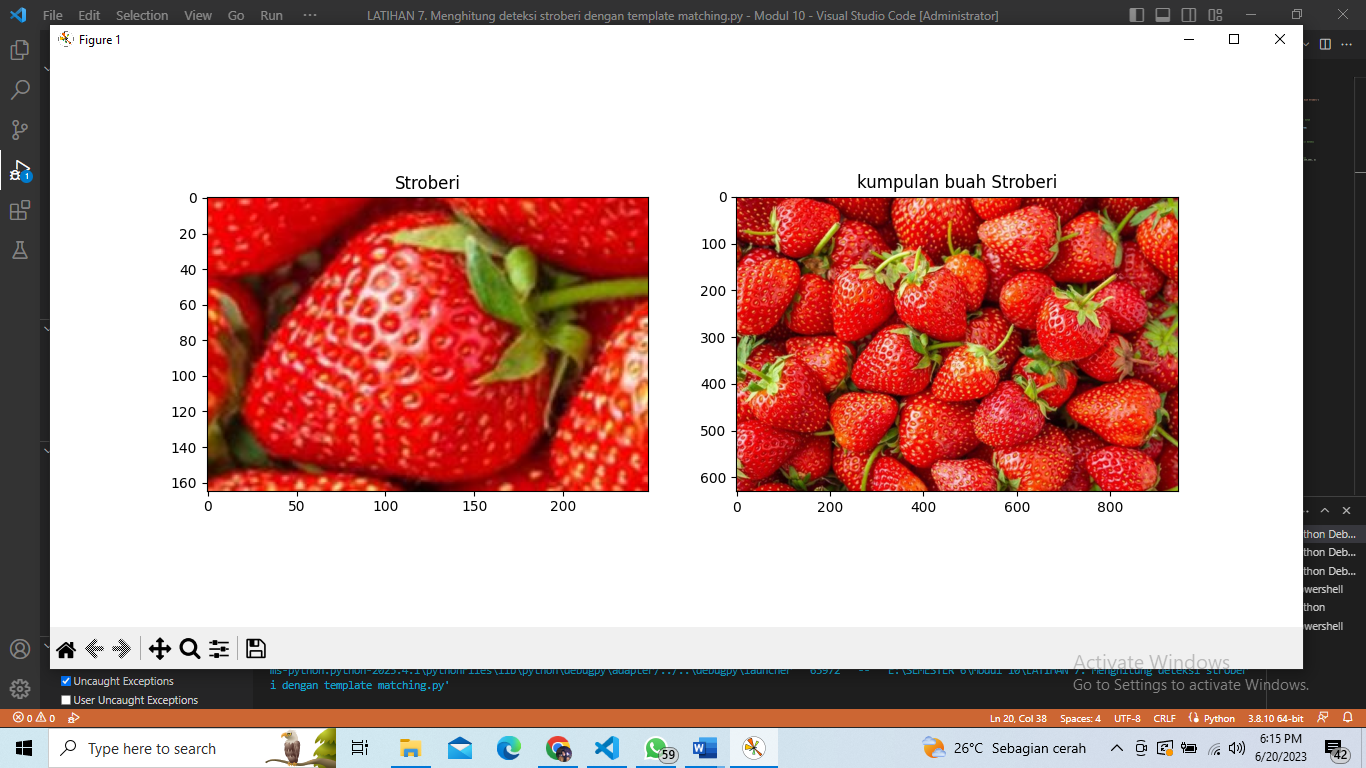


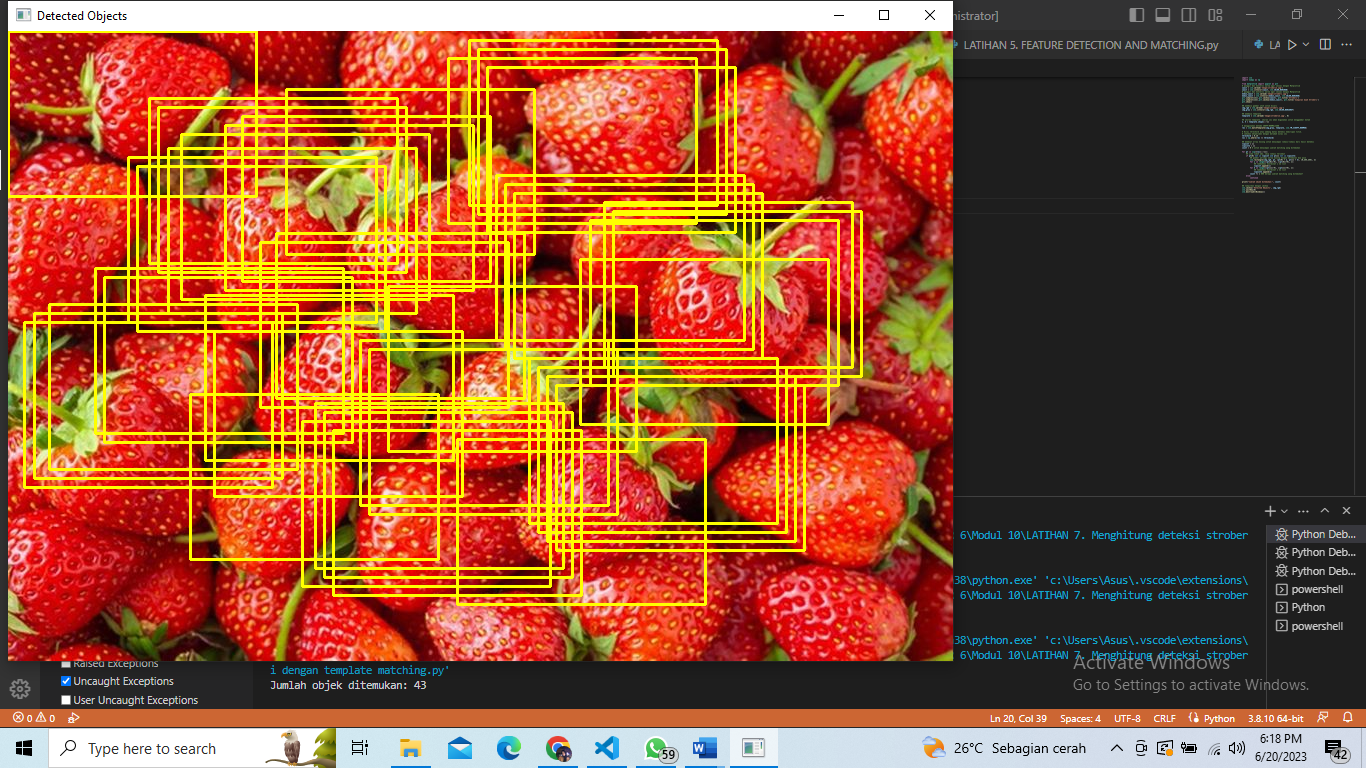
Percobaan keenam adalah penggunaan Matplotlib untuk menampilkan dua gambar secara bersamaan. Awalnya, gambar dengan nama 'Image/shifamutia2.jpg' dipanggil dan diubah ke mode warna RGB menggunakan fungsi cv2.cvtColor(). Kemudian, gambar tersebut ditampilkan pada subplot pertama dengan nomor 121 dan diberi judul 'ShifaMutia'. Selanjutnya, gambar dengan nama 'Image/shifamutia.jpg' dipanggil dan juga diubah ke mode warna RGB. Gambar tersebut kemudian ditampilkan pada subplot kedua dengan nomor 122 dan diberi judul 'Fotbar'. 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.



Pada codingan tersebut, dilakukan template matching menggunakan berbagai metode yang tersimpan dalam list 'methods'. Setiap metode diiterasi menggunakan variabel 'met'. Hasil template matching disimpan dalam variabel 'res'. Posisi terbaik dari hasil template matching dihitung dengan menggunakan fungsi 'cv2.minMaxLoc()'. Posisi terbaik ditentukan berdasarkan nilai minimum atau maksimum, tergantung pada metode yang digunakan. Dilakukan penggambaran persegi pada citra kerja ('img') untuk menunjukkan lokasi terdeteksi menggunakan fungsi 'cv2.rectangle()'.

1. Menghitung deteksi Stroberi dengan template matching





Percobaan terakhir yaitu Menghitung deteksi Stroberi dengan template matching. Pada kodingan yang dipakai fungsi cv2.matchTemplate() untuk mendeteksi kemiripan antara template dan gambar. Metode ini mencari korelasi terbaik antara template dan bagian gambar yang serupa. Parameter Threshold: Anda menggunakan nilai threshold sebesar 0.15 dalam baris res = cv2.matchTemplate(img\_gray, template, cv2.TM\_CCOEFF\_NORMED). Nilai threshold ini digunakan untuk mengatur ambang batas deteksi kemiripan titik. Semua nilai korelasi yang melebihi threshold akan dianggap sebagai kemiripan yang valid. Output dari kodingan tersebut adalah dua subplot yang menampilkan gambar-gambar pohon pinus. Pada subplot pertama (121), gambar 'pinus1.jpg' ditampilkan dengan judul 'pohon pinus'. Pada subplot kedua (122), gambar 'pinus.jpg' ditampilkan dengan judul 'kumpulan pohon pinus'. Dengan menggunakan plt.show(), jendela plot yang berisi kedua gambar akan ditampilkan.