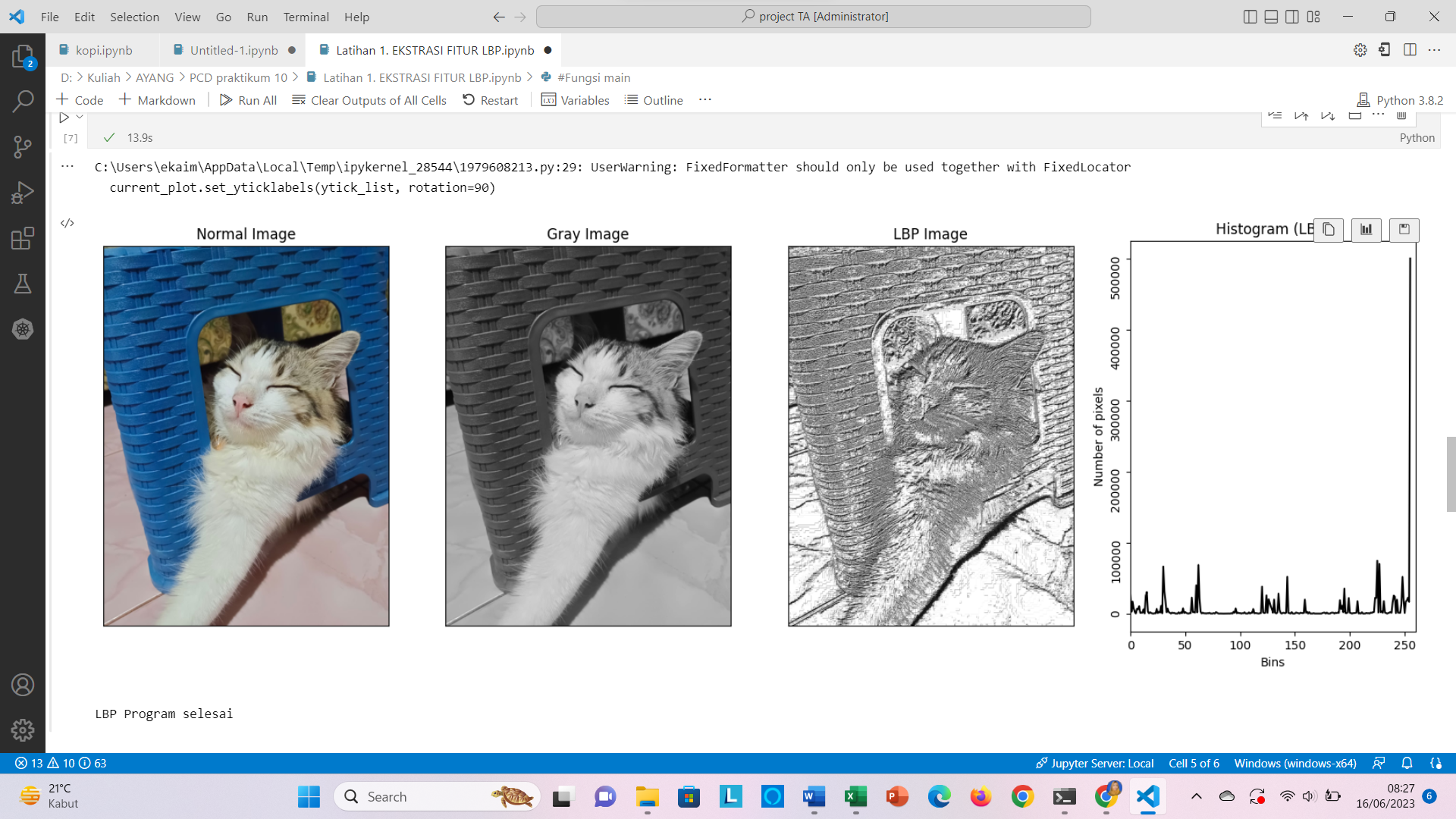
NAMA : ZALFA HUMAIRA SALSABILA

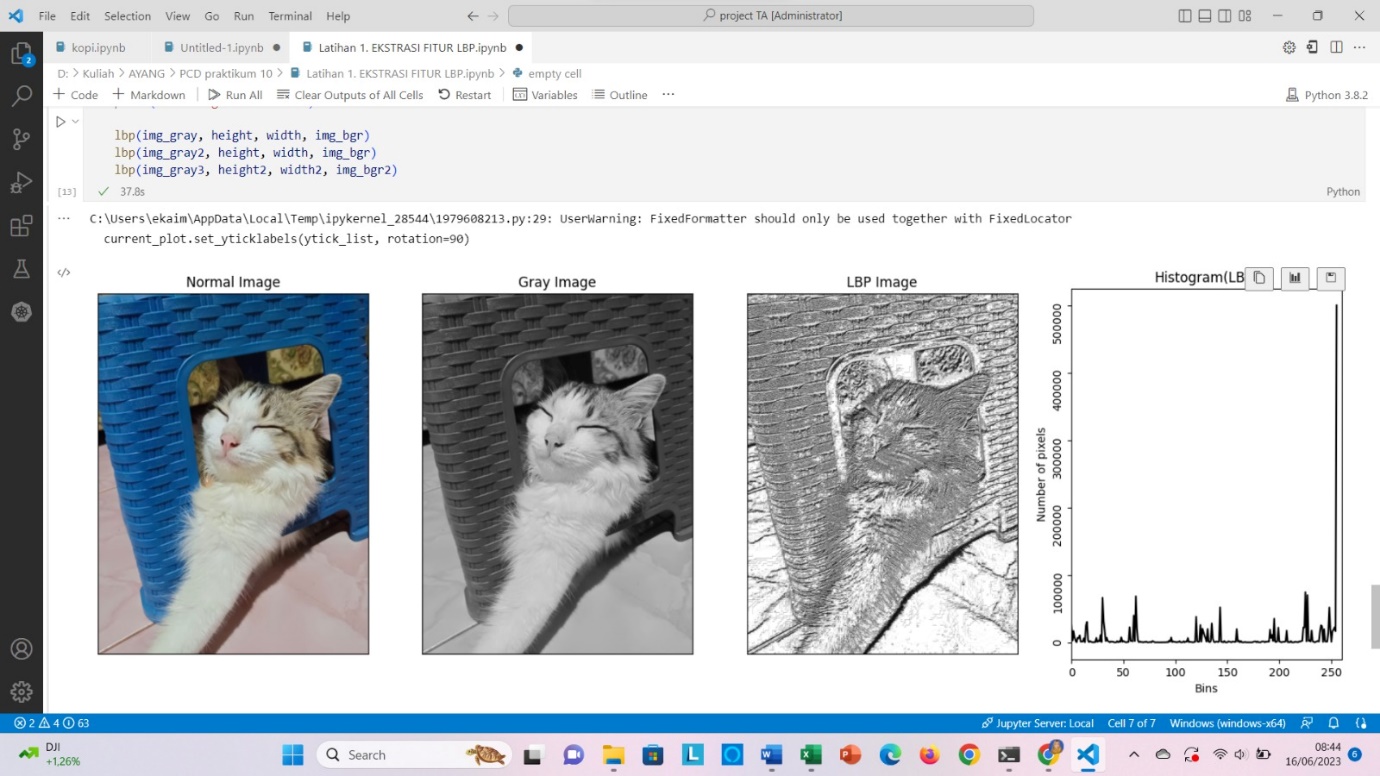
NIM : 1207070130

PCD TKK Praktikum ke-10

***Latihan 1. EKSTRASI FITUR LBP***

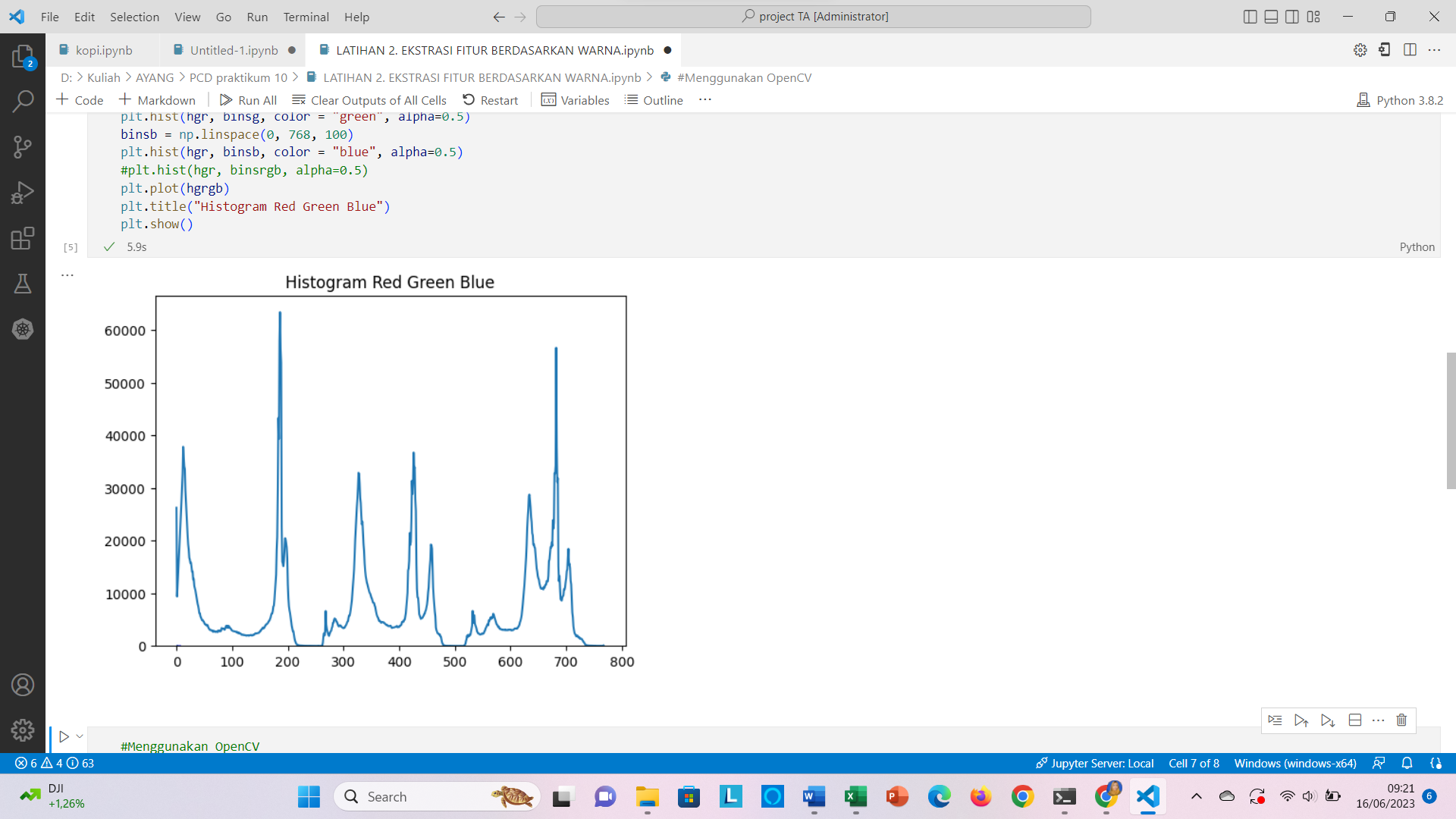
******

*Mencoba untuk menggunakan birghtness yang berbeda*

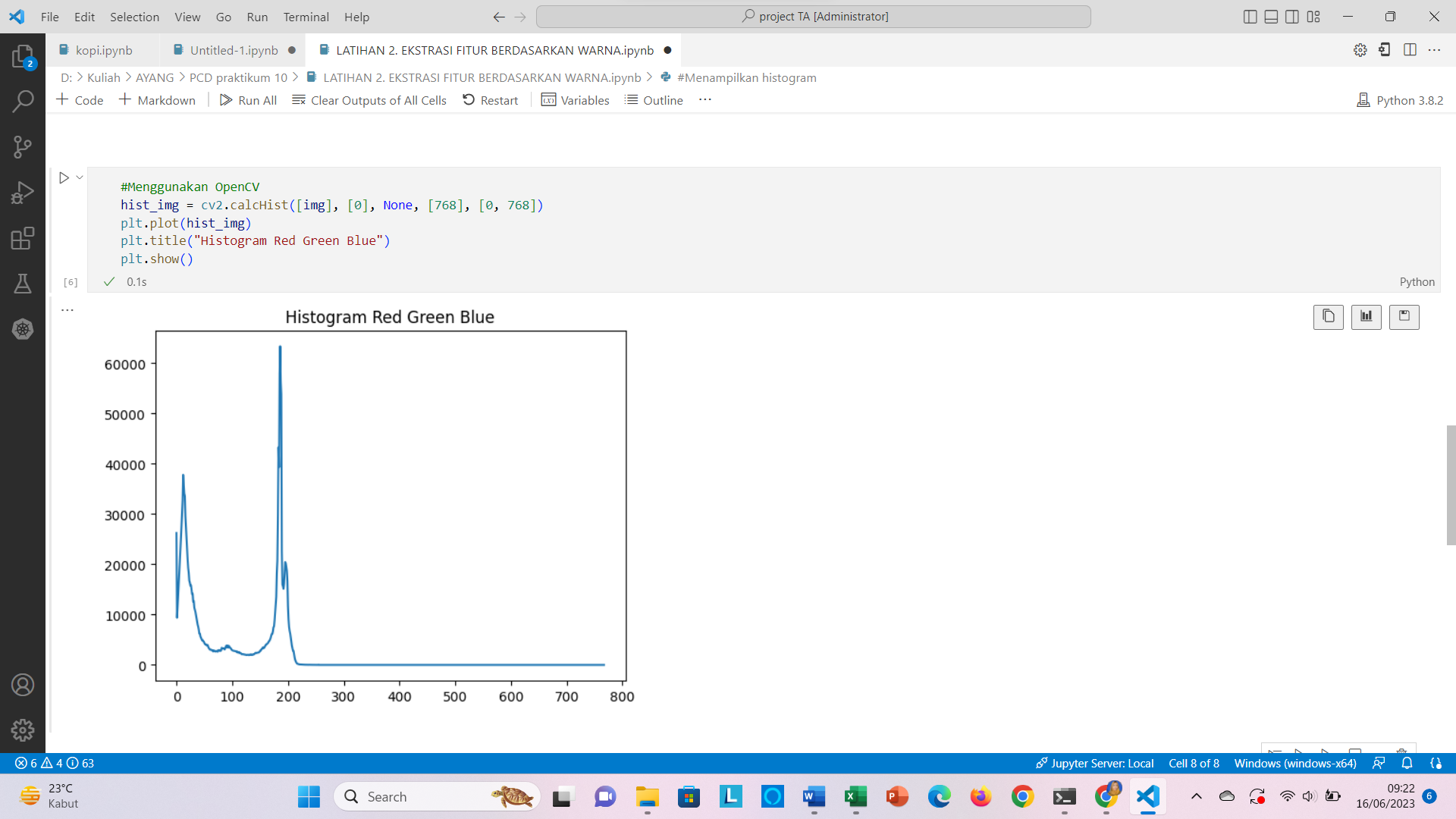
******

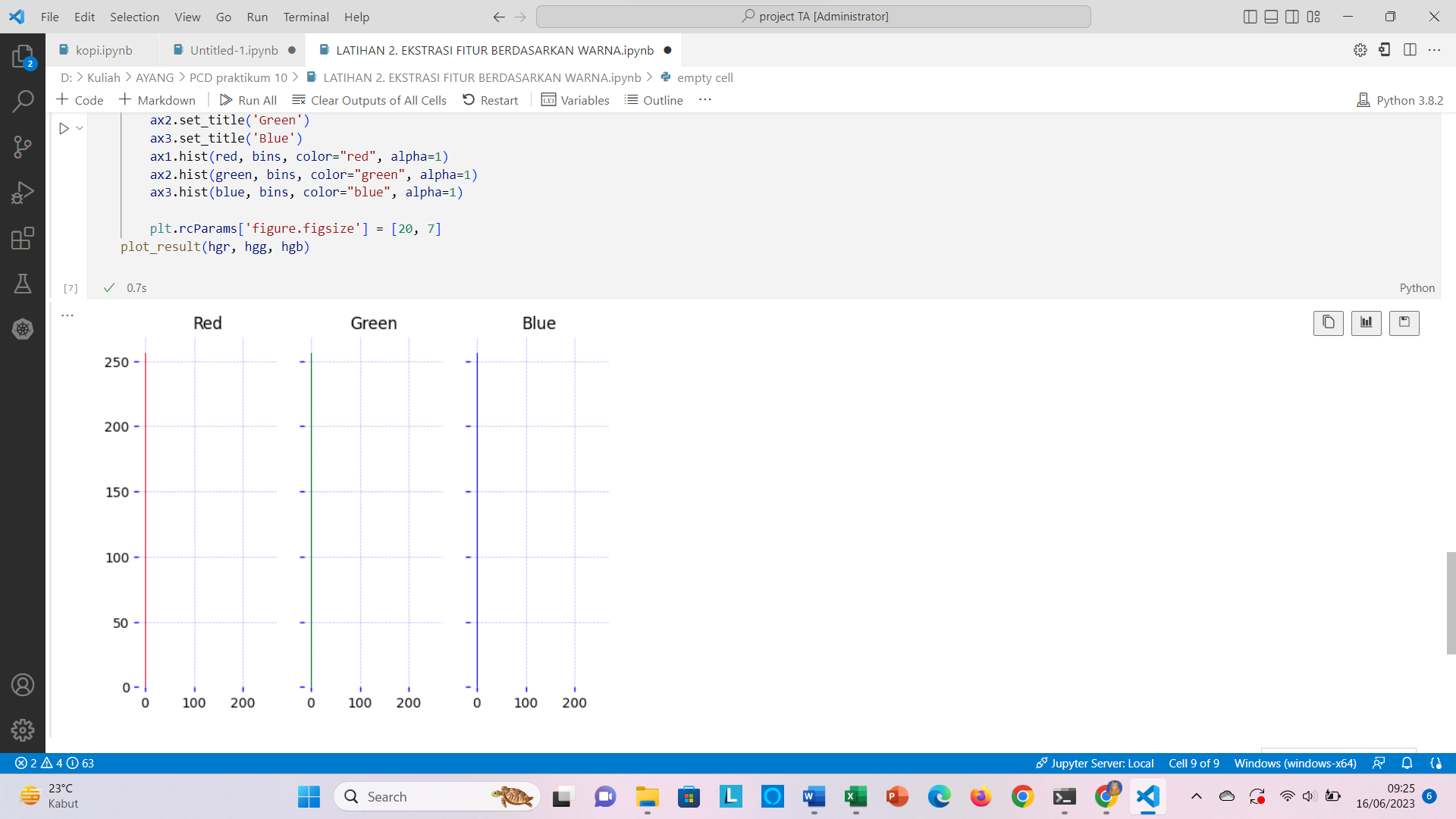
Kedua gambar diatas merupakan output untuk fitur LBP dari kedua codingan yang berbeda, namun output yang dihasilkan tetap sama yaitu setelah dilakukan fitur LBP gambar jadi lebih terlihat teksturnya dari pada gambar pada gray image dan normal image. Proses utama program melibatkan pembacaan gambar, konversi ke citra grayscale, perhitungan LBP pada setiap pixel, perhitungan histogram LBP, dan menampilkan output. Fitur ini menggunakan fungsi get\_pixel untuk mendapatkan nilai pixel tetangga dengan membandingkannya dengan nilai pixel pusat.

***LATIHAN 2. EKSTRASI FITUR BERDASARKAN WARNA***

******

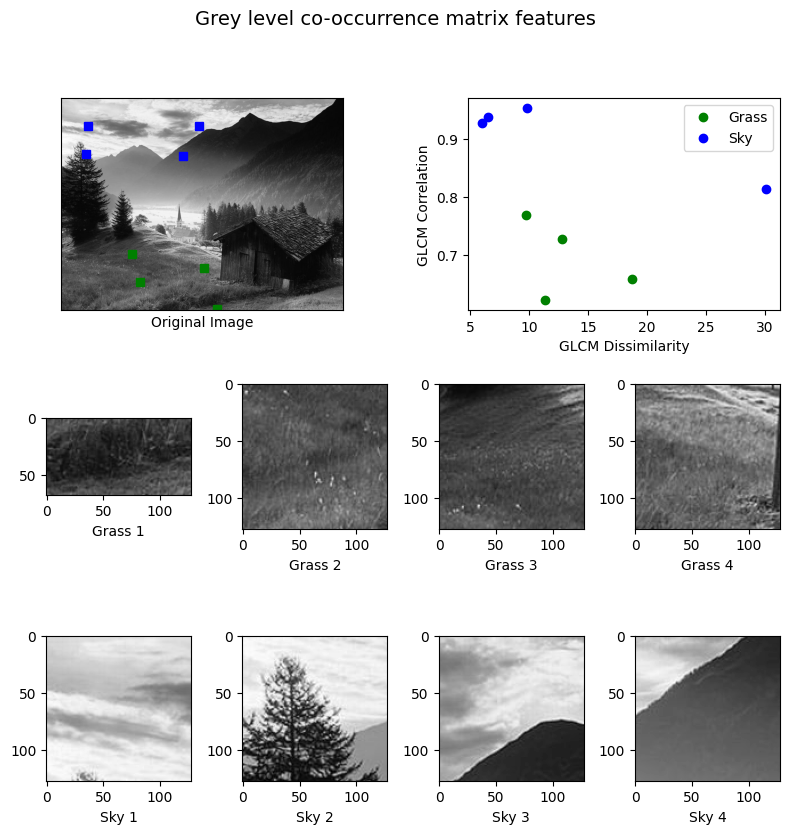
*Menggunakan Open Cv*

**

**

Pada latihan ini yaitu menampilkan histogram ekstraksi fitur berdasarkan warna. Menampilkan histogram chanel red green blue menggunakan metode perulangan dan menggunakan opencv dengan cukup menuliskan cv2.calcHist, namun sebetulnya ada perbedaan hasil pada histogram kedua codingan. Sementara untuk menampilkan histogram dari masing masing chanel perlu untuk memecah gambar menjadi chanel rgb dalam latihan ini untuk memecahnya memakai perulangan sehingga histogram pada setiap chanel bisa ditampilkan pada gambar ke-3.

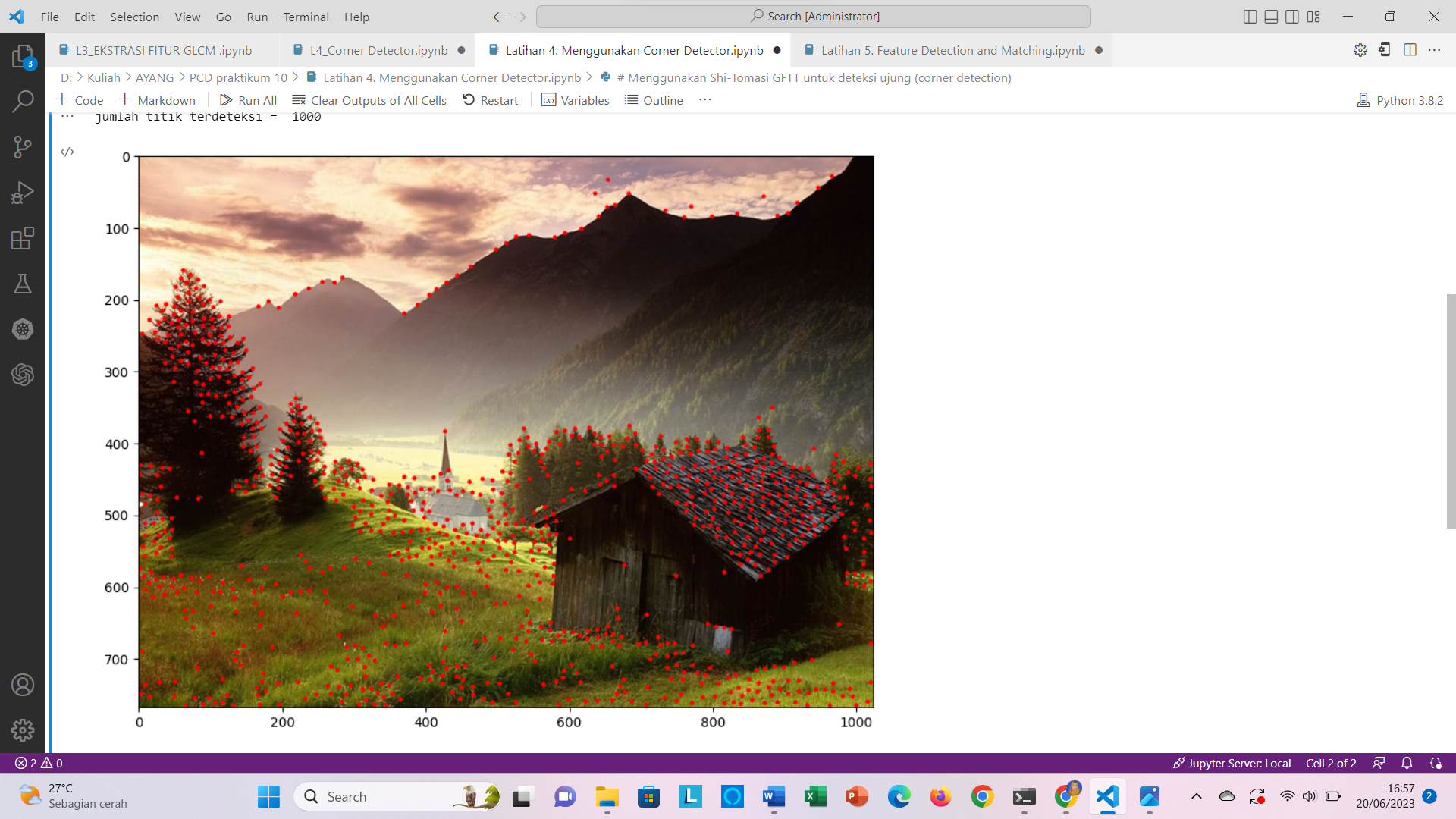
***Latihan 3. Fitur GLCM***

******

Untuk output latihan ke 3 ini tidak di screenshoot melainkan langsung save gambar karena jika di ss akan menyulitkan untuk melihat perbedaannya. Pada fitur GLCM ini yang saya tangkap adalah digunakan untuk memisahkan objek. Misalnya seperti gambar diatas ada titik biru yang mengindikasikan gambar langit, sementara titik hijau mengindikasikan rumput. Lalu output dibawahnya yaitu yg ke 2 menunjukan tampilan dari beberapa titik hijau pada gambar awal yaitu gambar rumput maka terlihat bahwa ke 4 gambar tersebut merupakan gambar rumput yang diberi titik hijau sebelumnya, begitupun pada output ke 3 menunjukan tampilan dari beberapa titik biru pada gambar awal yaitu gambar langit maka terlihat bahwa ke 4 gambar tersebut merupakan gambar langit yang sebelumnya sudah diberi titik biru.

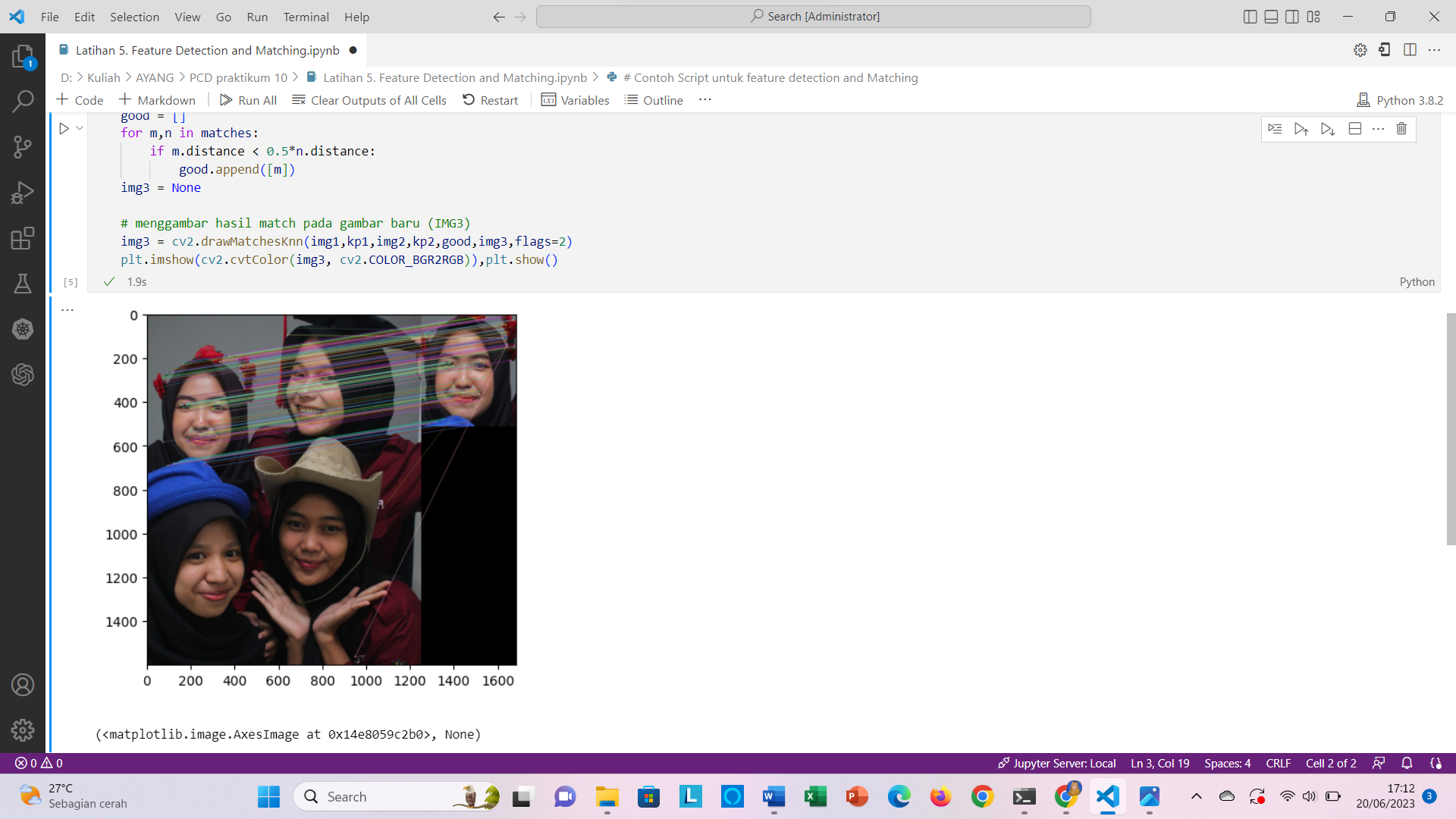
Penempatan titik koordinat untuk meninjau gambar ini dapat diubah ubah melalui codingan pada bagian grass\_locations = [(700, 500), (600, 223), (500, 192), (550, 455)] angka di dalam kurung tersebut akan menentukan dimana posisi titiknya dari atas maupun bawah atau kiri maupun kanan (x,y) dengan x menentukan posisi vertikal dan y menentukan posisi horizontal.

***Latihan 4. Corner Detector***

******

Awalnya gambar yang akan di lakukan corner detection diubah dulu menjadi format grayscale untuk mempermudah proses corner detection. Lalu corner detection ini menggunakan cv2.goodFeaturesToTrack(). Fungsi ini mengambil argumen gambar skala abu-abu (gray), jumlah maksimum titik yang ingin dideteksi (di sini, 1000), kualitas minimum dari fitur yang ingin dideteksi (di sini, 0.01), dan jarak minimum antara titik-titik yang dideteksi (di sini, 10). Hasil deteksi pojok disimpan dalam variabel corners. Setelah proses corner detection selesai barulah gambar dikembalikan ke format rgb. Untuk setiap titik pojok yang terdeteksi, dilakukan perulangan menggunakan for i in corners. Koordinat x dan y dari setiap titik diambil menggunakan i.ravel(). Kemudian, lingkaran kecil dengan jari-jari 3 piksel ditarik di sekitar koordinat pojok menggunakan cv2.circle(). Gambar yang telah diberi lingkaran tersebut disimpan dalam variabel rgb.

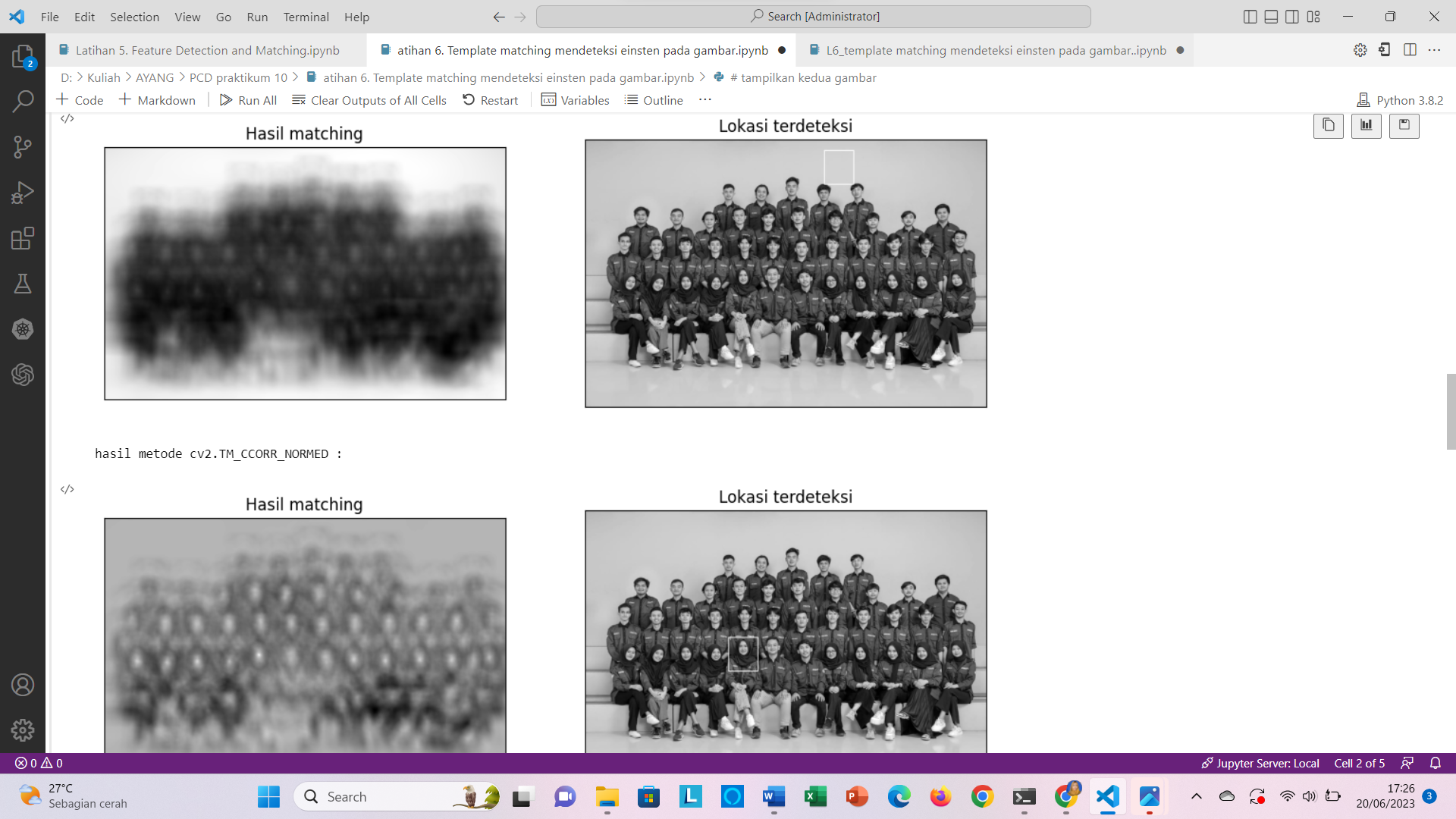
***Latihan 5. Fitur Detection and Matching***

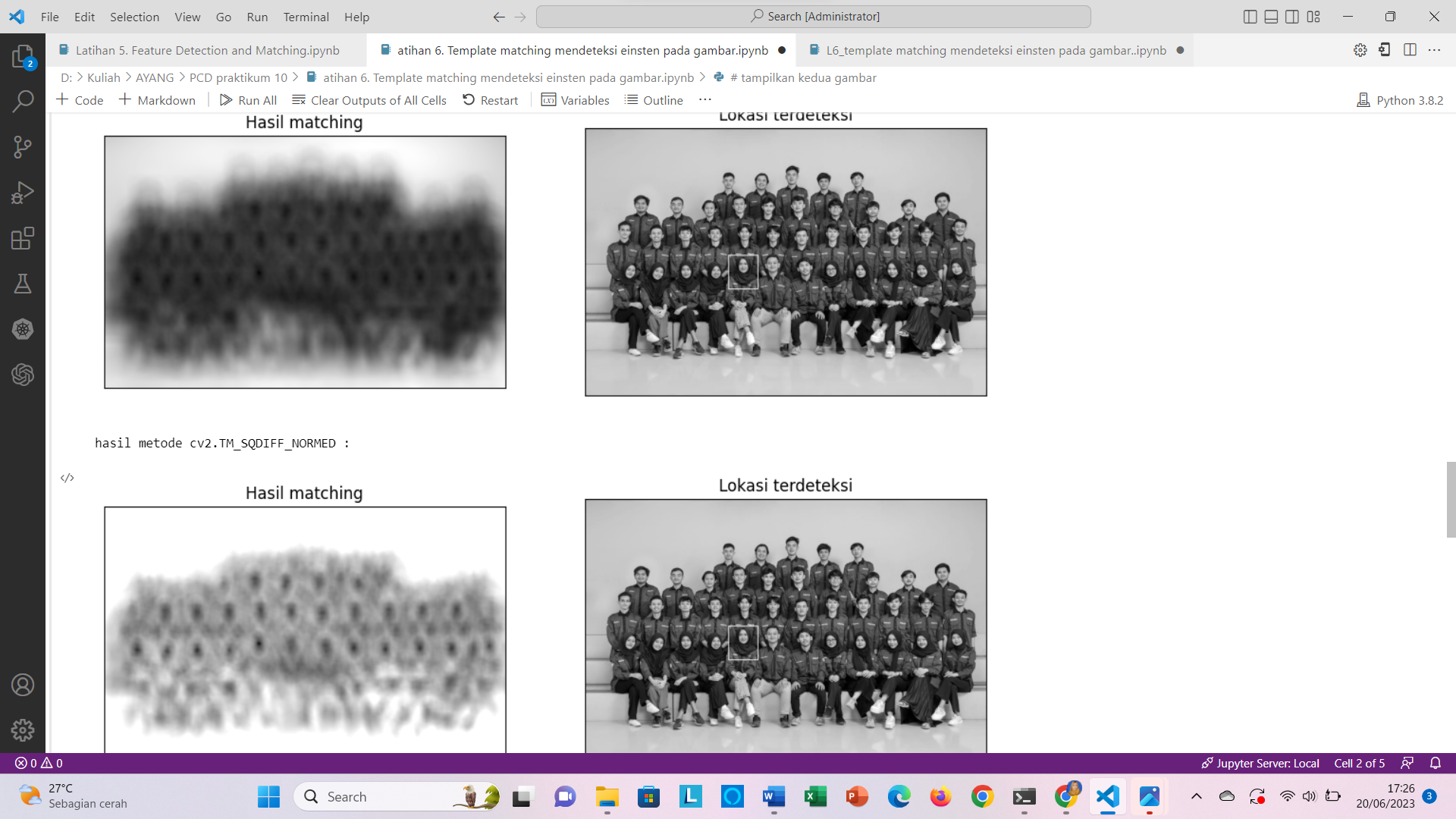
******

Program di atas merupakan contoh implementasi untuk melakukan pencocokan fitur antara dua gambar menggunakan algoritma SIFT (Scale-Invariant Feature Transform) dan Brute-Force Matcher. Pada awalnya kita harus memasukan 2 gambar yang ingin di cocokan , gambar tersebut diubah terlebih dahulu menjadi greyscale. Kemudian dibuat objek detektor SIFT menggunakan cv2.SIFT\_create(). Objek ini digunakan untuk mendeteksi titik kunci (keypoints) dan menghitung deskriptor menggunakan metode SIFT. Menggunakan objek detektor SIFT yang telah dibuat, melakukan deteksi titik kunci dan menghitung deskriptor pada kedua gambar menggunakan sift.detectAndCompute(). Hasil deteksi titik kunci dan deskriptor untuk masing-masing gambar disimpan dalam variabel kp1, des1 (gambar pertama) dan kp2, des2 (gambar kedua). Kemudian objek Brute-Force Matcher menggunakan cv2.BFMatcher() digunakan untuk melakukan pencocokan fitur antara deskriptor dari kedua gambar. Untuk pencocokan fitu antara antara des1 dan des2 menggunakan bf.knnMatch(). Metode ini mengembalikan pasangan keypoints terbaik yang paling cocok berdasarkan jarak antara deskriptor. Untuk menggambar hasil matching pada gambar baru (img3), kita dapat menggunakan fungsi cv2.drawMatchesKnn() sehingga menghasilkan gambar yang menunjukkan hasil matching antara dua gambar dengan keypoint yang cocok.

***Latihan 6. template matching mendeteksi einstein pada gambar***

******

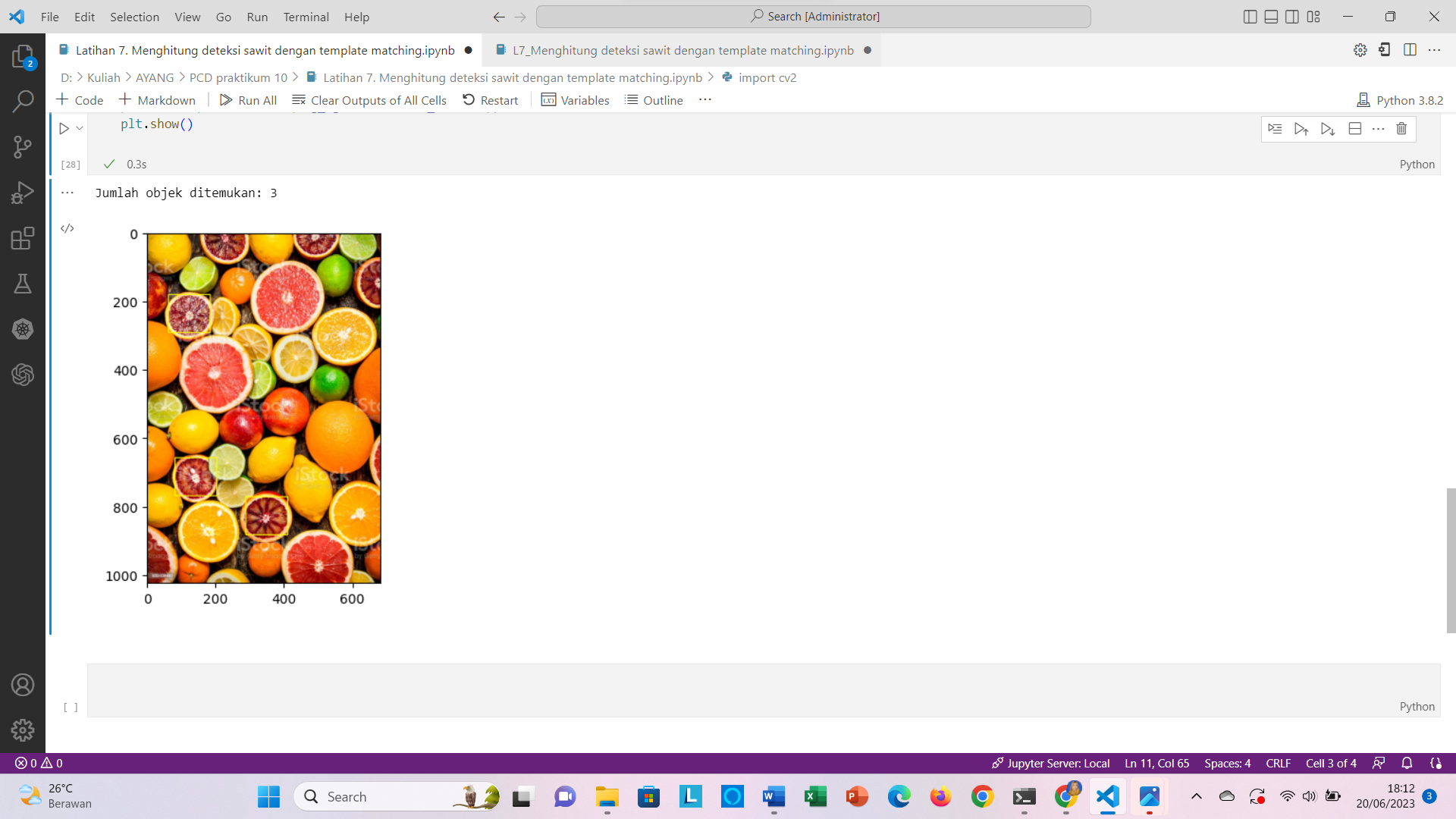
******

******

Program di atas merupakan implementasi dari template matching untuk menemukan kesamaan antara gambar template dan gambar masukan. Gunakan fungsi cv2.imread untuk membaca input dan gambar template. Gambar template ini akan dicocokkan dengan gambar masukan menggunakan metode pencocokan template. Program ini menggunakan enam metode berbeda untuk perbandingan, termasuk cv2.TM\_CCOEFF, cv2.TM\_CCOEFF\_NORMED, cv2.TM\_CCORR, cv2.TM\_CCORR\_NORMED, cv2.TM\_SQDIFF, dan cv2.TM\_SQDIFF\_NORMED. Metode-metode ini akan diterapkan secara berurutan ke gambar input. Selanjutnya, program menggunakan plt.rcParams["figure.figsize"] untuk memperbesar ukuran tampilan hasil plot. Dalam setiap iterasi, citra masukan disalin ke citra baru, yang diproses menggunakan metode pencocokan templat yang saat ini digunakan. Hasil pencocokan template disimpan dalam variabel res. Program ini juga mencari ukuran gambar template untuk menggambar kotak di mana ia menemukannya. Kemudian, periksa metode cv2.TM\_SQDIFF dan cv2.TM\_SQDIFF\_NORMED. Jika metode yang digunakan saat ini adalah salah satu dari metode tersebut, maka nilai top\_left diambil dari min\_loc, karena semakin kecil nilai yang dihasilkan maka semakin baik kemiripannya. Jika menggunakan metode lain, nilai top\_left akan diambil dari max\_loc, karena semakin besar nilai yang dihasilkan, semakin baik kesamaannya. Selanjutnya, program membuat persegi dengan lebar garis 2 pada posisi yang ditemukan menggunakan fungsi cv2.rectangle. Program mencetak hasil dari metode template matching yang digunakan. Terakhir, program menampilkan gambar yang cocok dengan template dan gambar persegi yang menunjukkan posisi yang terdeteksi menggunakan matplotlib

Jika input untuk tamplate deteksi wajah menggunakan gambar lain hasil yang keluar adalah sistem tidak bisa mendeteksi wajah dengan baik karena ada perbedaan input yang dimasukan.

***Latihan 7: Menghitung deteksi sawit dengan template matching***

******

Program melakukan template matching dengan metode cv2.TM\_CCOEFF\_NORMED menggunakan cv2.matchTemplate. Nilai ambang batas deteksi kemiripan ditentukan dengan threshold 0.3. Kemudian, program mencari lokasi-lokasi yang memiliki nilai kemiripan di atas atau sama dengan threshold menggunakan np.where. Nilai treshold ini jika diubah ubah mempengaruhi pendeteksian, jika nilainya tidak pas bisa saja pendeteksian meleset. Program melakukan iterasi pada setiap lokasi yang ditemukan dan melakukan beberapa tindakan. Pertama, program menggambar persegi kuning pada lokasi tersebut menggunakan cv2.rectangle. Kemudian, program menyimpan koordinat x dan y dalam list lspoint dan lspoint2 dengan rentang (+/- 9) dari titik tengah persegi. Tujuan dari ini adalah untuk menghindari duplikasi deteksi yang berdekatan. Jumlah deteksi yang ditemukan juga dihitung menggunakan variabel count. Terakhir, program menampilkan gambar dengan objek yang terdeteksi menggunakan cv2.imshow. Program menunggu tombol keyboard yang ditekan sebelum menutup jendela tampilan menggunakan cv2.waitKey dan cv2.destroyAllWindows. Deteksi objek dilakukan dengan menggambar kotak persegi pada lokasi yang memiliki kemiripan di atas atau sama dengan threshold yang ditentukan. Jumlah objek yang ditemukan juga dicetak dan hasil deteksi objek sebanyak 3