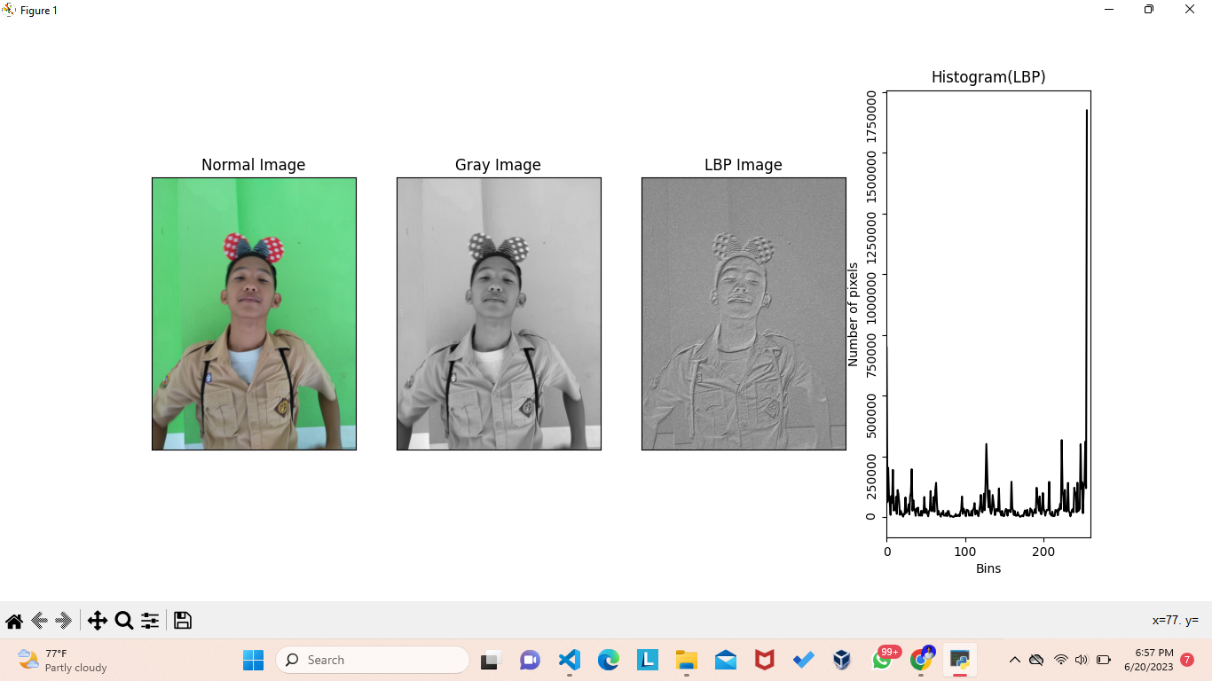
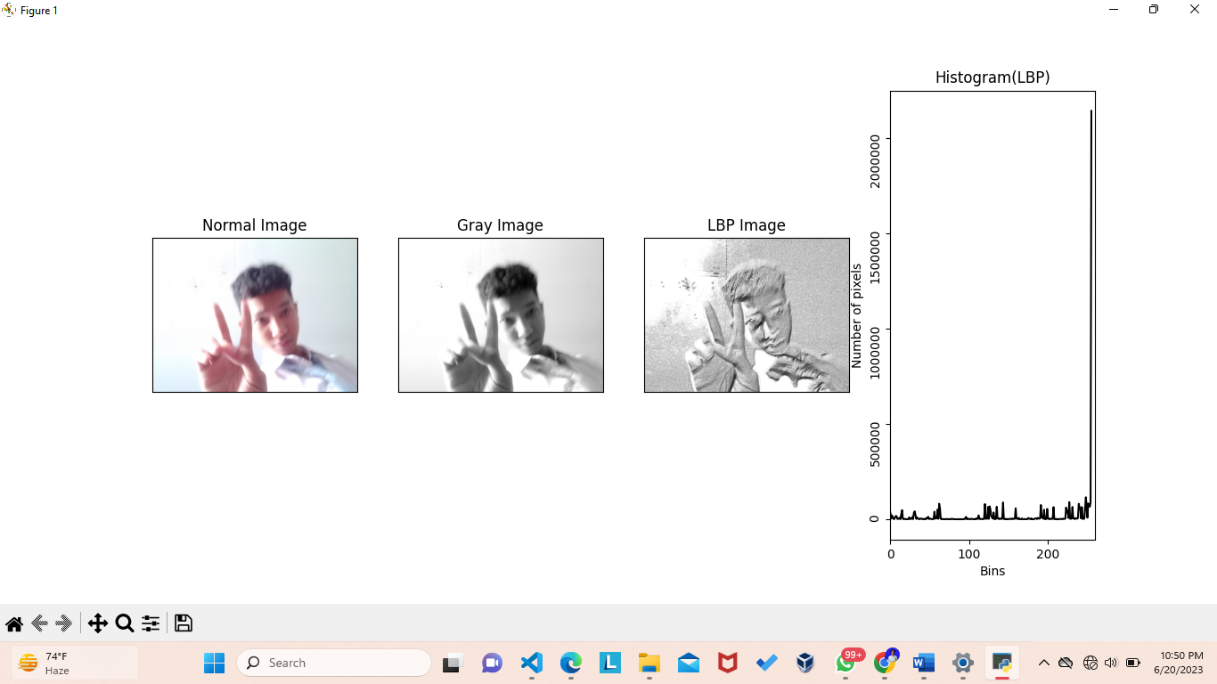
Nama : Rajasabian Ardhi Awangga

NIM : 1207070096

Mata Kuliah : Praktikum Pengolahan Citra Digital – TKK

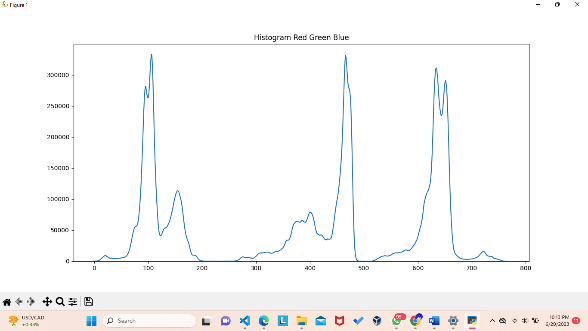
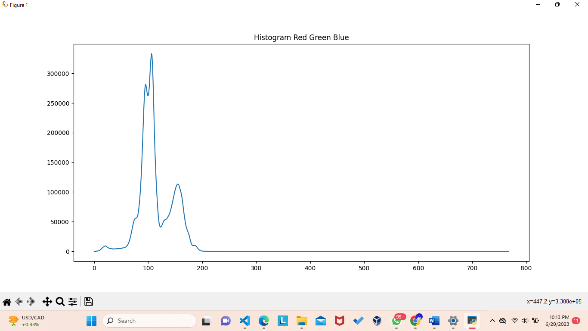
PRAKTIKUM 10

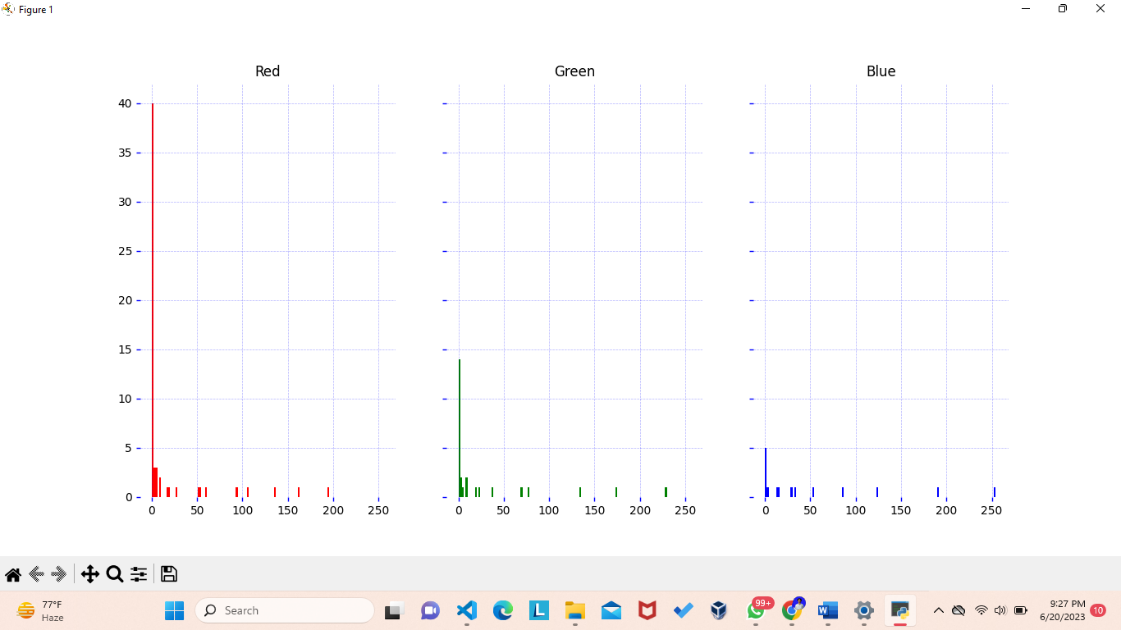
1. **Praktikum 10.1 – Ekstraksi Fitur LBP**



* + Proses utama program melibatkan pembacaan gambar, konversi ke citra grayscale, perhitungan LBP pada setiap pixel, perhitungan histogram LBP, dan menampilkan output.
  + Program juga menyertakan contoh penggunaan dengan menggunakan gambar yang memiliki tingkat kecerahan yang berbeda.
  + Pada akhir program, hasil LBP ditampilkan dalam bentuk gambar dan histogram.

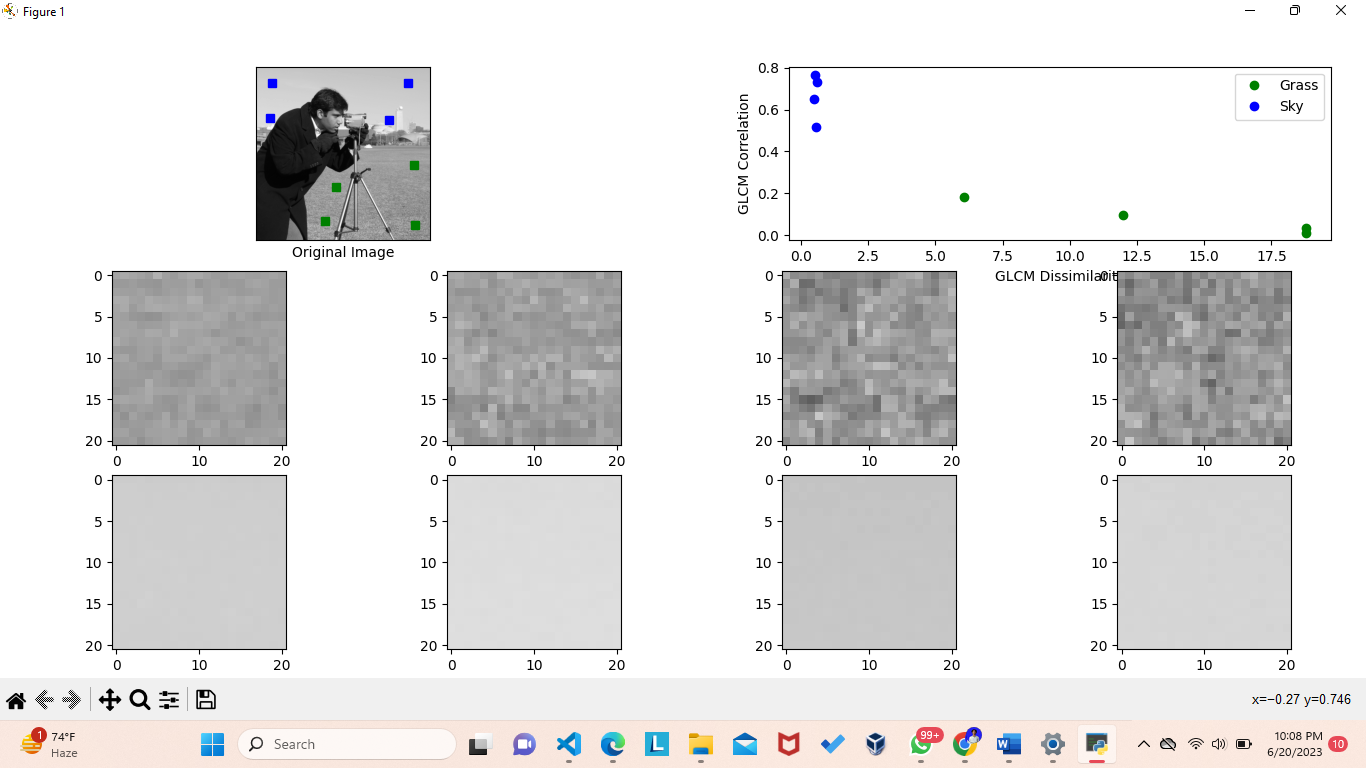
1. **Praktikum 10.2 – Ekstraksi Fitur Berdasarkan Warna**



* + Fungsi makeItZero digunakan untuk menginisialisasi array dengan nilai 0.
  + Program menghitung histogram dari komponen warna (merah, hijau, biru) dan menampilkan histogram RGB menggunakan Matplotlib.
  + Program juga menggunakan fungsi cv2.calcHist dari OpenCV untuk menghitung histogram warna dan menampilkannya.

1. **Ekstraksi Fitur GLCM**



Ekstraksi Fitur GLCM (Gray-Level Co-occurrence Matrix):

* + Program ini menggunakan library Matplotlib dan scikit-image (skimage) untuk memanipulasi dan menampilkan gambar.
  + Program menggunakan gambar "camera" dari library scikit-image sebagai input.
  + Program ini mengambil beberapa area di gambar (misalnya rumput dan langit) dan menghitung GLCM untuk setiap area. GLCM dihitung untuk setiap patch menggunakan fungsi graycomatrix dari scikit-image.
  + Fitur-fitur GLCM seperti dissimilarity dan correlation dihitung menggunakan fungsi graycoprops dari scikit-image.
  + Hasilnya ditampilkan dalam bentuk scatter plot, dengan sumbu x dan y masing-masing merupakan dissimilarity dan correlation dari setiap patch.

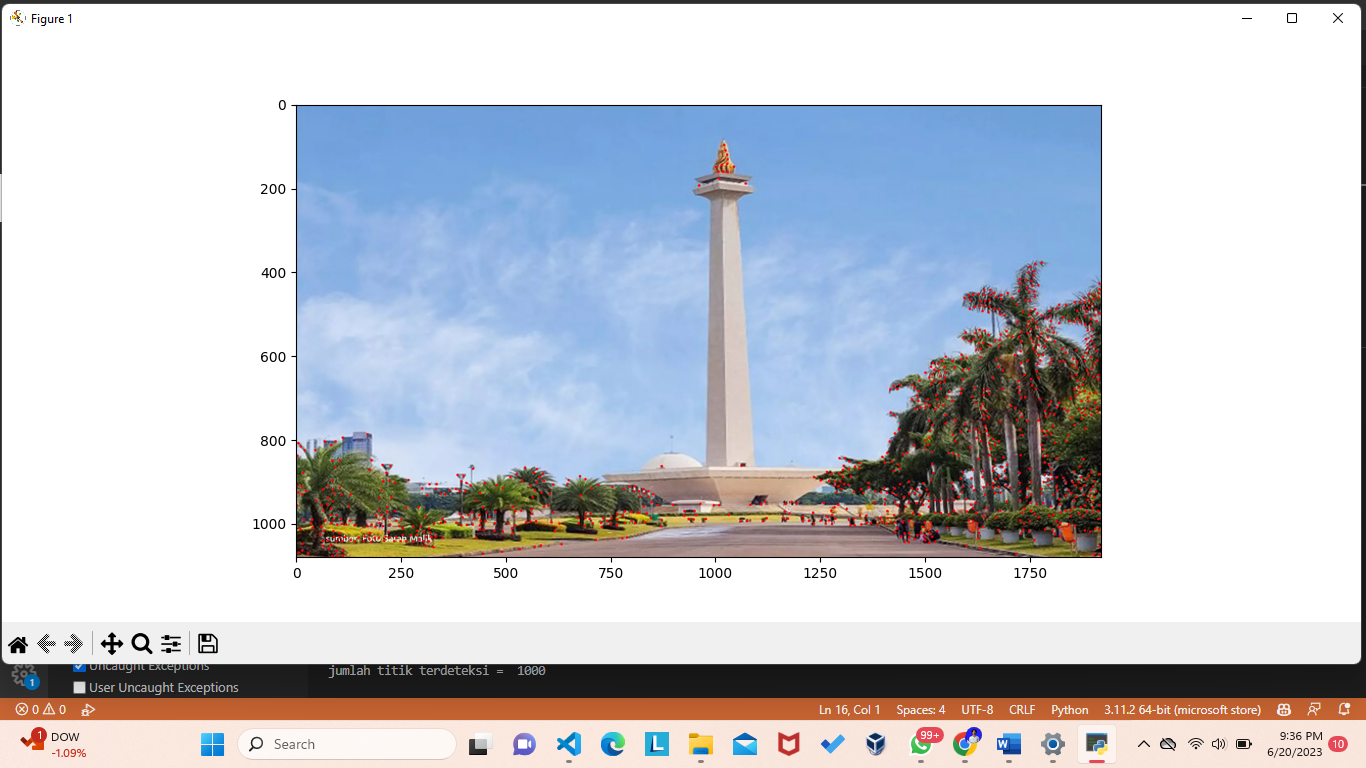
**Kesimpulan Perbandingan Ekstraksi Fitur LBP, Berdasarkan Warna, dan GLCM :**

Terdapat perbedaan pada ketiga cara program tersebut dalam mengekstraksi fitur. LBP baik untuk analisis tekstur atau menggambarkan pola tekstur dalam gambar, sedangkan program warna dan GLCM mengekstraksi fitur berdasarkan distribusi warna dalam gambar. Metode ekstraksi fitur yang sesuai tergantung pada jenis analisis yang ingin dilakukan dan karakteristik gambar yang akan diproses.

Kebutuhan analisis dan jenis gambar yang akan diolah menentukan pemilihan program. Tidak satu pun dari program-program ini memiliki tujuan yang sama dan dapat menyediakan data yang sama.

Pemilihan program yang tepat tergantung pada kebutuhan analisis dan sifat gambar yang akan diolah. Semua program tersebut memiliki kegunaan yang berbeda dan dapat memberikan informasi yang berbeda pula tergantung pada jenis data yang diproses.

1. **Praktikum 10.4 – Corner Detector**



Dalam program di atas, algoritma Shi-Tomasi Good Features to Track (GFTT) digunakan untuk mendeteksi sudut pada gambar grayscale. Fungsi cv2.goodFeaturesToTrack() pada OpenCV digunakan untuk menjalankan metode Shi-Tomasi GFTT, yang mendeteksi sudut pada gambar grayscale. Argumen pertama, yang diwakili oleh warna hitam, adalah sumber gambar; argumen kedua, yang diwakili oleh nilai 1000, adalah jumlah sudut terbesar yang diinginkan untuk dideteksi; argumen ketiga, yang diwakili oleh nilai 0,01, adalah kualitas sudut yang diinginkan, dan argumen keempat, yang diwakili oleh jarak terkecil antara sudut yang diinginkan. Fungsi NumPy np.int0() mengkonversi tipe data array ke tipe data integer.

Fungsi melakukan iterasi untuk setiap sudut yang terdeteksi pada for i in corners. Untuk mengubah dimensi array sudut menjadi variabel x dan y, gunakan fungsi cv2.circle() pada OpenCV untuk menggambar lingkaran pada gambar. lingkaran dengan pusat (x, y) dan radius tiga pixel, warna putih (255), dan ketebalan -1 untuk mengisi lingkaran.

Oleh karena itu, program akan membaca gambar monas.jpg, menemukan sudut menggunakan metode Shi-Tomasi GFTT, menghitung jumlah titik yang ditemukan dengan jumlah titik sebanyak 1000, dan menampilkan gambar dengan sudut yang ditandai menggunakan lingkaran.

1. **Praktikum 10.5 – Feature Detection and Matching**

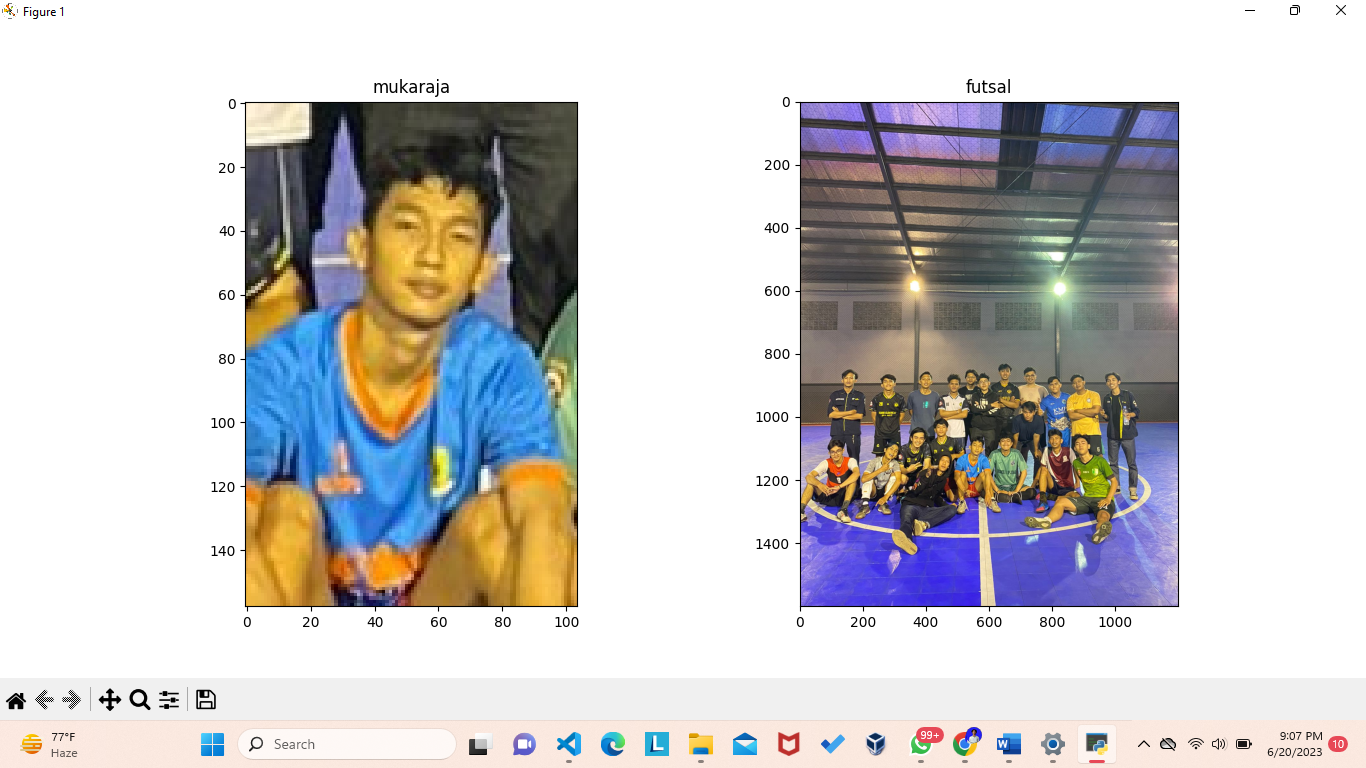


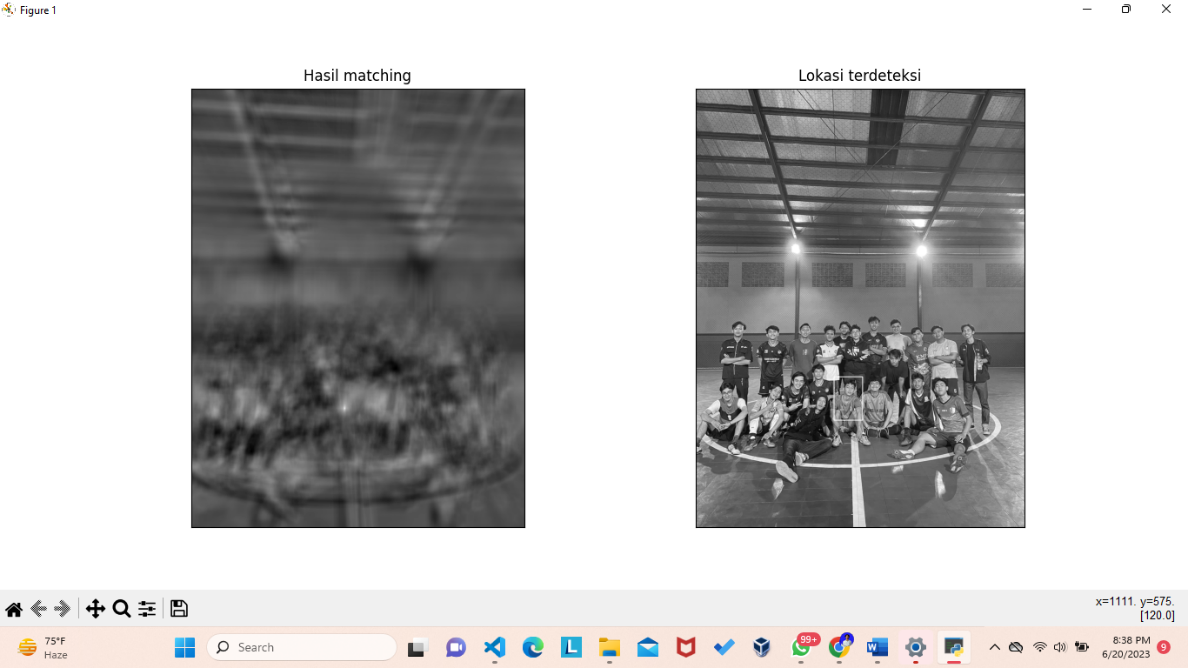
Script yang disebutkan di atas menggunakan algoritma SIFT dan Brute-Force Matcher untuk mendeteksi dan menyamakan fitur antara dua gambar. Metode cv2.SIFT\_create dapat digunakan untuk membuat objek detektor SIFT dalam OpenCV. Dengan metode SIFT, objek ini mendeteksi poin kunci dan menghitung deskriptor. Deskriptor adalah representasi numerik yang menggambarkan fitur poin kunci dalam citra. Kita juga dapat menggunakan fungsi sift.detectAndCompute() untuk melakukan deteksi poin kunci dan menghitung deskriptor. Argumen pertama menyatakan bahwa citra grayscale akan digunakan, sedangkan argumen kedua menyatakan bahwa mask opsional dapat digunakan untuk membatasi area di mana deteksi poin penting akan dilakukan. Kita dapat melakukan analisis dan pengolahan gambar menggunakan metode SIFT OpenCV dengan menggunakan objek detektor SIFT dan fungsi detectAndCompute().

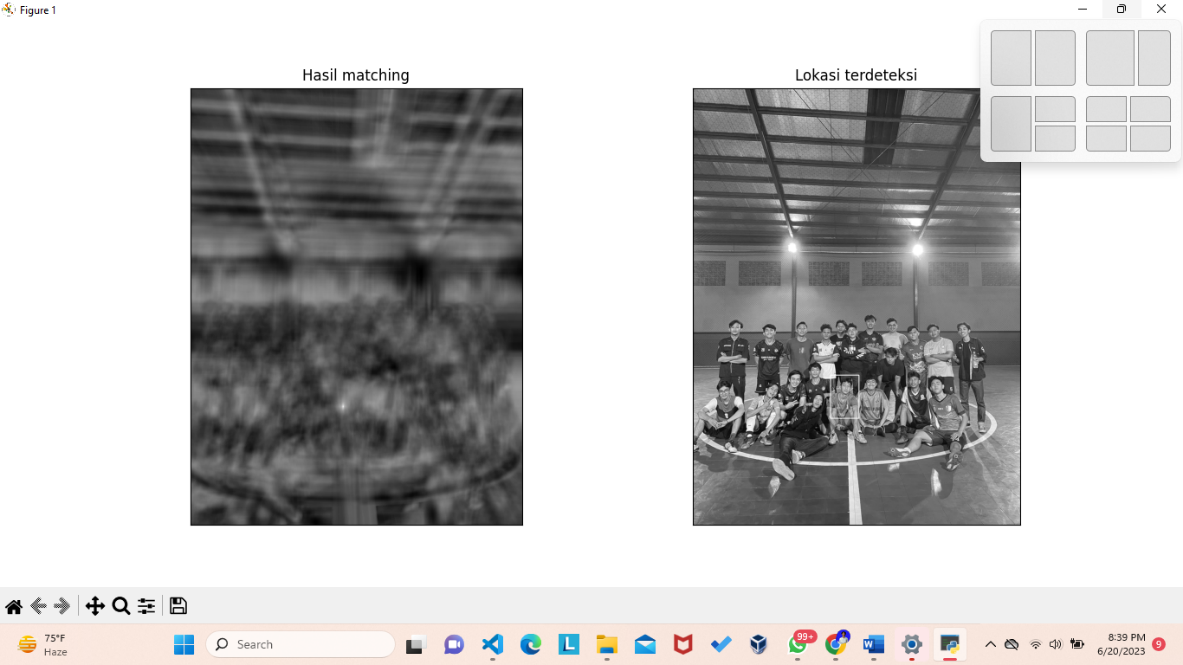
Selama proses pencarian, kita dapat menggunakan objek matcher Brute-Force dengan menggunakan metode cv2.BFMatcher() dari OpenCV. Metode k-NN (k-Nearest Neighbors) digunakan untuk mencocokkan fitur. Kita memberikan dua argumen ke fungsi bf.knnMatch(). Argumen pertama menunjukkan deskriptor fitur dari dua gambar, di sini des1 dan des2, dan argumen kedua menentukan jumlah tetangga terdekat yang akan dicocokkan, k=2. Kami dapat melakukan pencocokan fitur antara dua gambar dengan metode k-NN dengan menggunakan objek matcher Brute-Force dan fungsi knnMatch().

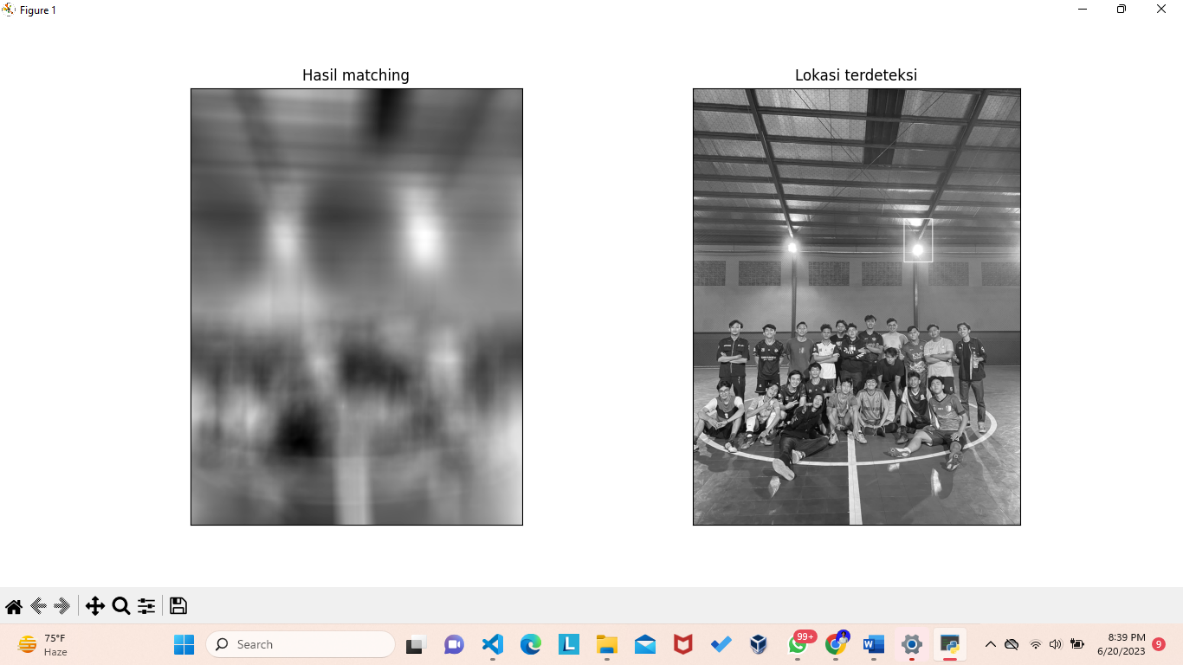
Fungsi cv2.drawMatchesKnn() dapat digunakan untuk menggambar hasil matching pada gambar baru (img3). Fungsi ini menghasilkan gambar yang menunjukkan hasil matching antara dua gambar dengan keypoint yang sesuai. untuk menunjukkan hasilnya pada gambar pertama, yang menampilkan banyak garis-garis berwarna sebagai sinyal bahwa gambar kedua sesuai. Ketika gambar wajah diganti, program tetap mendeteksi dan menyesuaikan, tetapi tidak cukup menyesuaikan, sehingga garis-garis berwarna tidak sebanding dengan gambar sebelumnya..

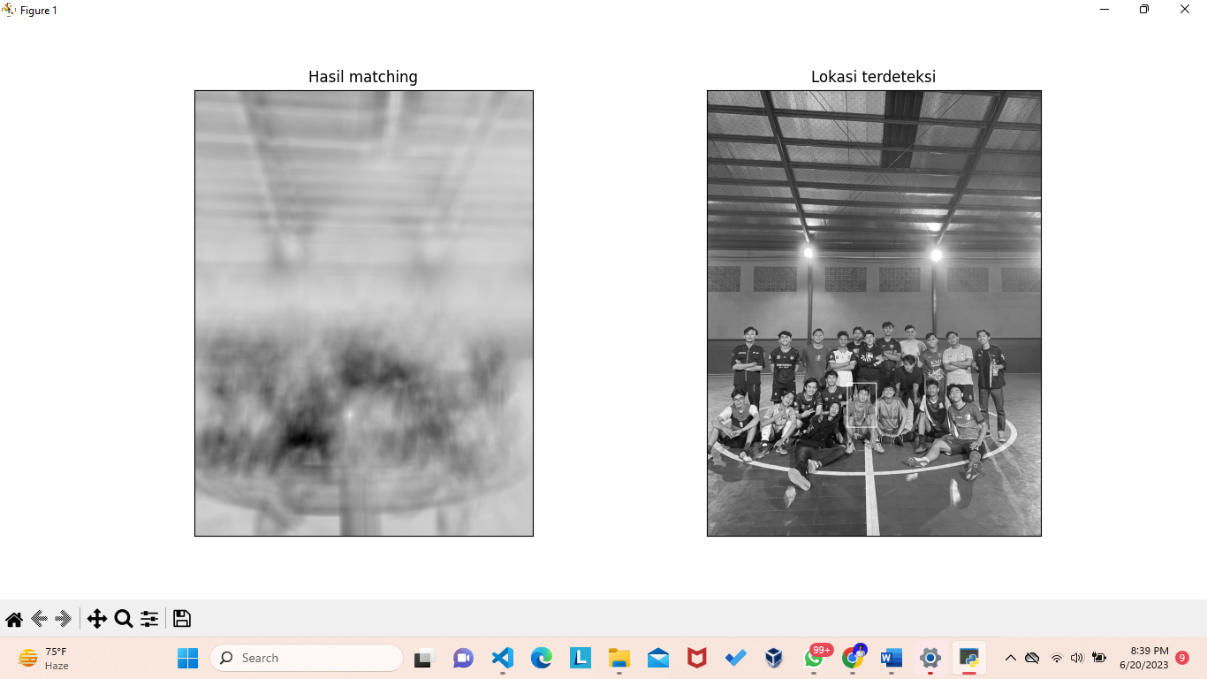
1. **Praktikum 10.6 – Template Matching**

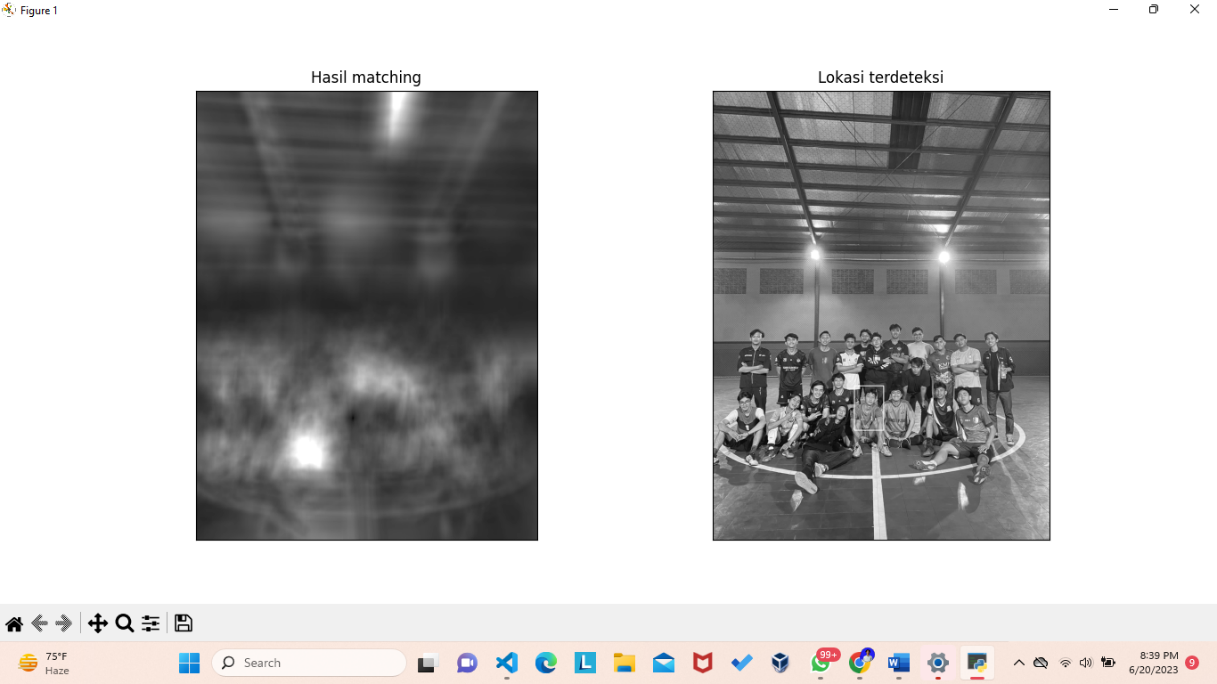
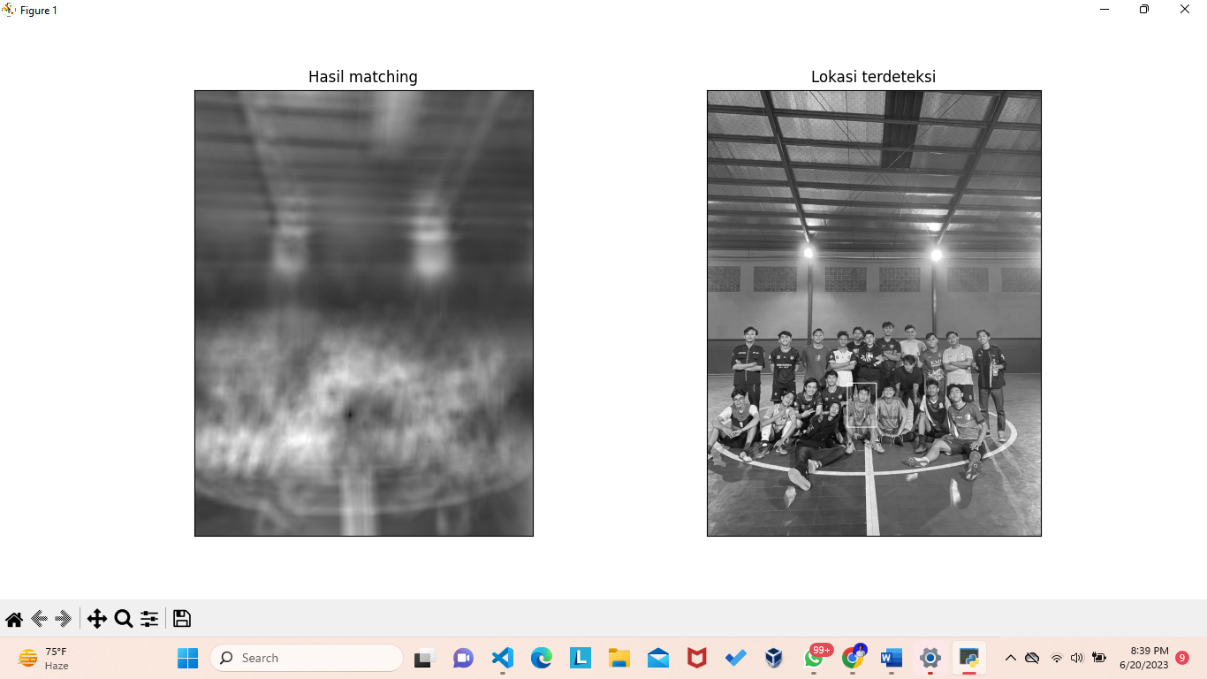












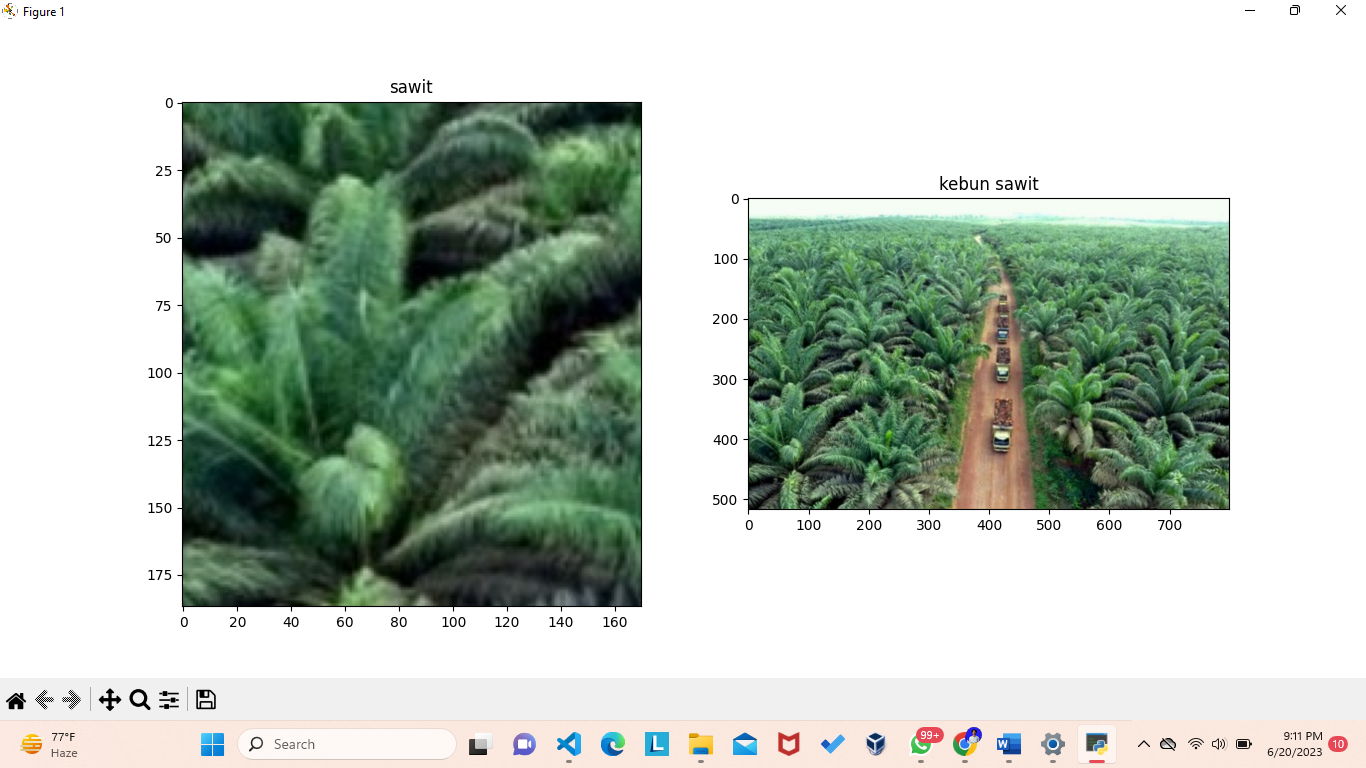
Dalam program ini, metode template matching digunakan untuk mengidentifikasi kemiripan antara gambar template dan gambar input. Fungsi cv2.imread digunakan untuk membaca gambar template dan gambar input, dan metode template matching digunakan untuk mengcocokkan gambar template dengan gambar input.

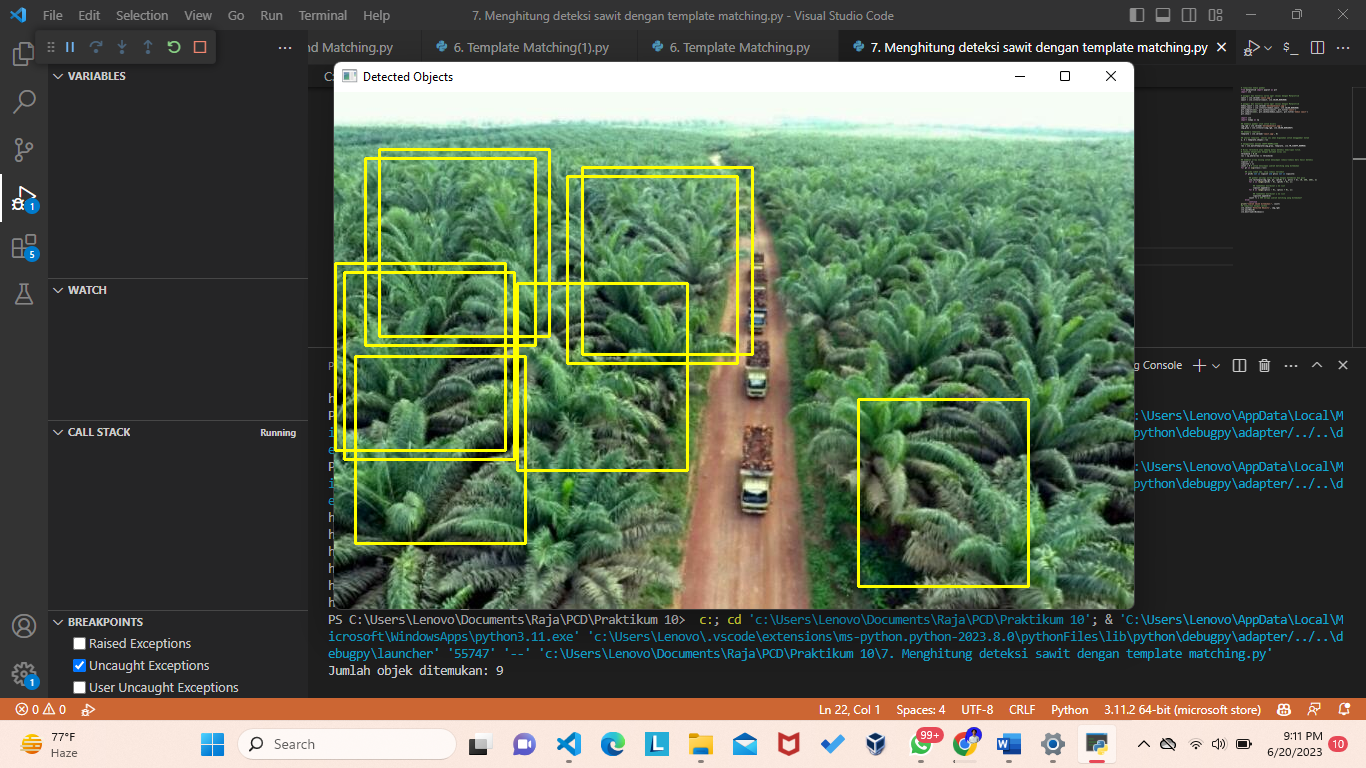
Program menggunakan enam metode yang berbeda untuk perbandingan, yang terdiri dari cv2.TM\_CCOEFF, cv2.TM\_CCOEFF\_NORMED, cv2.TM\_CCORR, cv2.TM\_CCORR\_NORMED, cv2.TM\_SQDIFF, dan cv2.TM\_SQDIFF\_NORMED. Metode-metode ini akan diterapkan secara berurutan pada citra input. Selanjutnya, program memperbesar ukuran tampilan hasil plotting menggunakan plt.rcParams["figure.figsize"].

Pada setiap iterasi, citra input disalin menjadi citra baru untuk diproses dengan metode template matching yang sedang digunakan. Hasil template matching disimpan dalam variabel res. Selain itu, program mencari ukuran citra template untuk menggambar kotak persegi pada lokasi yang ditemukan. Jika metode cv2.TM\_SQDIFF dan cv2.TM\_SQDIFF\_NORMED ditemukan, maka nilai top\_left diambil dari variabel res.Selanjutnya, program membuat persegi pada lokasi yang ditemukan menggunakan fungsi cv2.rectangle dengan ketebalan garis 2. Program mencetak hasil metode-template matching yang digunakan. Terakhir, program menampilkan citra hasil template matching dan citra dengan persegi yang menunjukkan lokasi terdeteksi menggunakan matplotlib.

Jika tidak mengunakan gambar yang sama atau foto yang tidak terkait pada gambar yang dicari maka tidak akan menunjukkan hasil apa apa, tetapi program akan mendeteksi latar,warna dan bentuk yang serupa pada gambar.

1. **Praktikum 10.7 – Menghitung deteksi sawit dengan Template Matching**





Program di atas terdiri dari dua bagian utama. Bagian pertama menggunakan library matplotlib untuk menampilkan kedua gambar, dan bagian kedua melakukan deteksi objek menggunakan metode template matching pada gambar.

Bagian pertama program menggunakan library matplotlib untuk menampilkan dua gambar, masing-masing "sawit.jpg" dan "kelompoksawit.jpg". Gambar pertama ditampilkan pada subplot 1 dan diberi judul "sawit", sedangkan gambar kedua ditampilkan pada subplot 2 dan diberi judul "kebun sawit". Semua ini dilakukan dengan menggunakan cv2.imread dan kemudian diubah ke format warna yang sesuai dengan matplotlib.

Untuk memulai bagian kedua program, gambar "kelompoksawit.jpg" dibaca secara keseluruhan dan diubah menjadi skala abu-abu menggunakan cv2.cvtColor. Kemudian, gambar template "sawit.jpg" juga dibaca dan diubah menjadi skala abu-abu. Setelah itu, ukuran template diambil dan digunakan untuk menggambar kotak persegi di lokasi yang ditemukan. Dengan menggunakan cv2.matchTemplate, program melakukan matching template dengan metode cv2.TM\_CCOEFF\_NORMED. Ambang batas deteksi kemiripan adalah nilai 0.15. Dengan menggunakan np.where, program mencari lokasi dengan nilai kemiripan di atas atau sama dengan ambang batas.

Setiap tempat yang ditemukan, program melakukan iterasi dan melakukan beberapa tindakan. Pertama, program menggunakan cv2.rectangle untuk menggambar persegi kuning di tempat tersebut. Kemudian, program menyimpan koordinat x dan y dalam list lspoint dan lspoint2 dengan rentang (+/- 9) dari titik tengah persegi. Ini dilakukan untuk mencegah duplikasi deteksi yang berdekatan. Terakhir, program menggunakan cv2.imshow untuk menampilkan gambar dengan objek yang terdeteksi. Kemudian, program menggunakan cv2.waitKey dan cv2.destroyAllWindows untuk menunggu tombol keyboard yang ditekan sebelum menutup jendela tampilan.

Program ini menggunakan metode template matching untuk menemukan objek yang serupa dalam gambar "sawit.jpg" dan menampilkan dua gambar secara berurutan menggunakan matplotlib. Untuk mengidentifikasi objek, kotak persegi digambarkan di tempat yang memiliki kemiripan di atas atau sama dengan batas yang ditentukan. Dicetak juga jumlah objek yang ditemukan dan hasil deteksi objek sebanyak 9.