

EB234908 PENGOLAHAN CITRA MEDIKA
FINAL PROJECT REPORT



Member of the Group :

Leony Purba (5023211013)
Benedicta Sabdaningtyas Pratita Pratanjana (5023211029)
Farhan Majid Ibrahim (5023211049)
Adelia Safira (5023211061)

Lecturer :

Nada Fitrieyatul Hikmah, S.T., M.T.

BIOMEDICAL ENGINEERING DEPARTMENT
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2024

BAB I

PENDAHULUAN

1.1 Latar Belakang

Eksim, atau yang dikenal juga sebagai dermatitis, adalah kelompok kondisi peradangan pada kulit yang ditandai oleh rasa gatal, kulit kering, ruam, dan bercak bersisik. Kondisi ini memengaruhi jutaan orang di seluruh dunia, dengan gejala yang bisa berkisar dari ringan hingga parah. Salah satu ciri khas eksim adalah rasa gatal yang intens, yang dapat mendorong penderita untuk menggaruk kulitnya sehingga dapat menyebabkan kerusakan lebih lanjut. Eksim tidak hanya berdampak pada kesehatan kulit, tetapi juga pada kualitas hidup penderitanya, karena sering kali rasa gatal dan iritasi membuat aktivitas sehari-hari terganggu.

Secara umum, eksim dapat muncul di berbagai bagian tubuh, meskipun lebih sering ditemukan pada area seperti tangan, kaki, wajah, lipatan siku, dan bagian belakang lutut. Gejala lain yang umum ditemui adalah kulit kering, bersisik, bercak merah atau meradang, serta lepuhan yang dapat pecah atau mengeras. Penampilan eksim juga dapat bervariasi sesuai dengan warna kulit; pada kulit yang lebih terang, eksim biasanya terlihat merah atau merah muda, sementara pada kulit yang lebih gelap dapat tampak sebagai bercak coklat, ungu, atau abu-abu.

Ada beberapa jenis eksim yang masing-masing memiliki pemicu dan gejala yang berbeda. Contohnya, *contact dermatitis* terjadi saat kulit bereaksi terhadap iritan atau alergen, sementara *dyshidrotic eczema* ditandai oleh lepuhan kecil pada tangan dan kaki yang biasanya dipicu oleh stres atau paparan logam tertentu. Jenis lainnya seperti *nummular eczema* muncul sebagai bintik-bintik berbentuk koin yang sangat gatal, sedangkan *seborrheic dermatitis* sering menyerang area kulit yang berminyak seperti kulit kepala dan wajah. Sementara itu, *stasis dermatitis* biasanya terjadi di area kaki bagian bawah akibat sirkulasi yang buruk dan ditandai oleh pembengkakan dan perubahan warna kulit. Meskipun penyebab pasti eksim belum sepenuhnya dipahami, para ahli berpendapat bahwa kondisi ini dipicu oleh kombinasi faktor genetik dan lingkungan. Pemicu eksim yang umum termasuk paparan terhadap iritan seperti sabun atau deterjen, alergen seperti serbuk sari dan bulu hewan peliharaan, serta perubahan suhu dan stres. Fluktuasi hormonal juga bisa menjadi faktor yang memperburuk kondisi ini.

Penanganan eksim tidak bisa menghilangkan penyakit ini sepenuhnya, tetapi beberapa metode dapat membantu mengelola gejala dengan efektif. Penggunaan pelembab secara rutin dapat membantu menjaga kelembaban kulit, sementara krim kortikosteroid digunakan untuk meredakan peradangan selama eksaserbasi. Selain itu, antihistamin dapat digunakan untuk mengurangi rasa gatal, dan obat imunosupresif non-steroid juga dapat membantu mengontrol respon imun. Menghindari pemicu yang sudah diketahui dan menjaga rutinitas perawatan kulit yang konsisten sangat penting untuk mengelola eksim secara efektif.

Dalam proyek ini, kami mengembangkan aplikasi Streamlit untuk membantu mendeteksi gejala eksim melalui analisis gambar kulit. Tujuan proyek ini adalah menyediakan alat bantu yang praktis bagi tenaga medis dan masyarakat untuk mengenali tanda-tanda eksim sejak dini, sehingga perawatan dapat dilakukan lebih cepat dan mencegah kondisi menjadi lebih parah. Di GitHub, kami menyimpan berbagai komponen kode dan model pembelajaran mesin yang dilatih dengan dataset gambar eksim dan kondisi kulit lain. Proses analisis menggunakan teknik deteksi tepi dan segmentasi, membantu aplikasi mengenali area yang berpotensi terkena eksim, seperti bercak merah atau tekstur yang tidak rata, sehingga hasilnya mudah dipahami pengguna. Aplikasi ini memiliki antarmuka sederhana yang memungkinkan pengguna mengunggah gambar dan mendapatkan hasil secara cepat.

1.2 Rumusan Masalah

Rumusan masalah dalam proyek ini adalah bagaimana menyediakan alat bantu deteksi dini eksim yang praktis, mudah digunakan, dan memiliki tingkat akurasi yang baik dalam mengenali karakteristik eksim pada gambar kulit, seperti bercak merah dan tekstur tidak rata, melalui aplikasi berbasis Streamlit. Selain itu, antarmuka yang sederhana dan mudah dipahami perlu dibuat agar dapat diakses oleh pengguna dengan berbagai tingkat pemahaman teknologi.

1.3 Tujuan dan Manfaat

Tujuan dari proyek ini adalah untuk mengembangkan aplikasi deteksi dini eksim berbasis Streamlit yang praktis dan akurat, mampu mengenali karakteristik khas eksim pada gambar kulit dengan tingkat akurasi tinggi, serta mudah digunakan oleh berbagai kalangan. Manfaat yang diharapkan dari proyek ini adalah memberikan solusi yang dapat membantu masyarakat dan tenaga medis dalam melakukan identifikasi eksim dengan cepat.

BAB II

TINJAUAN PUSTAKA

2.1 Eczema Subacute

Eksim atau dermatitis atopik adalah peradangan kulit yang terjadi pada lapisan epidermis dan dermis sebagai respons terhadap pengaruh faktor eksogen dan endogen. Kondisi ini menimbulkan kelainan klinis berupa efloresensi polimorfik, seperti eritema, edema, papul, vesikel, skuama, dan likenifikasi, serta disertai gatal yang menimbulkan ruam. Dermatitis merupakan penyakit kulit yang bisa bersifat akut, subakut, atau kronis, tergantung pada tingkat keparahan peradangan. Prevalensi eksim seumur hidup diperkirakan sekitar 15% hingga 30% pada anak-anak dan 2% hingga 10% pada orang dewasa. Sekitar 60% dari kasus eksim muncul dalam tahun pertama kehidupan.

Eksim lebih sering ditemukan di daerah pedesaan dibandingkan perkotaan, menunjukkan adanya keterkaitan antara gaya hidup, faktor lingkungan, dan perkembangan dermatitis atopik. Diperkirakan sekitar 50% penderita eksim parah juga mengalami asma, sementara 75% dari mereka mengalami rinitis alergi. Ciri khas eksim terlihat pada kondisi kulit yang kering dan gatal dan rentan terhadap infeksi. Selain ruam, pemeriksaan wajah yang menunjukkan garis "Dennie-Morgan" atau kerutan seperti lipatan tepat dibawah kelopak mata bawah dapat menjadi indikasi eksim. Penyakit ini banyak menyerang anak - anak namun dapat juga terlihat pada orang dewasa. Faktor utama penyebab terjadinya eksim merupakan kombinasi faktor genetik dan lingkungan.



Figure 3-11 Acute vesicular eczema has evolved into sub-acute eczema with redness and scaling.

Gambar 2.1 Penyakit eczema pada tangan

Pada penelitian telah dibuktikan bahwa eksim memiliki komponen genetik. Eksim memiliki komponen genetik yang kuat, dengan riwayat keluarga yang terkena eksim, asma, atau alergi yang umum ditemukan pada individu yang terkena. Beberapa gen yang terkait dengan eksim telah diidentifikasi, termasuk yang terlibat dalam fungsi penghalang kulit dan sistem kekebalan tubuh. Salah satu gen yang paling terkenal yang dikaitkan

dengan eksim adalah gen filaggrin (*FLG*). Gen ini memberikan instruksi untuk membuat protein yang disebut filaggrin, yang penting dalam menjaga fungsi penghalang kulit. Gen ini bertanggung jawab untuk menciptakan korneosit yang kuat dan datar yang membentuk lapisan pelindung kulit terluar.

Pada pasien dengan sel kulit normal, korneosit tersusun rapat dan teratur. Pasien dengan mutasi *FLG* akan memiliki penghalang kulit yang tidak berfungsi karena pengaturan sel kulit yang tidak teratur. Disfungsi ini menyebabkan penghalang kulit "bocor", yang menyebabkan hilangnya air dan kurangnya perlindungan dari zat berbahaya. Selain gen *filaggrin* terdapat gen lain yang terlibat dalam fungsi penghalang kulit pada perkembangan eksim. Gen-gen ini termasuk gen yang terlibat dalam sintesis dan transportasi lipid, seperti gen ceramide synthase dan gen *ABCA12*. Orang yang menderita eksim memiliki kadar β -defensin yang lebih rendah di kulit mereka. β -defensin adalah peptida pertahanan penting yang membantu melawan bakteri, virus, dan jamur tertentu. Kekurangan peptida ini membuat kulit lebih rentan terhadap kolonisasi dan infeksi, terutama oleh *Staphylococcus aureus* (S. aureus).

2.2 Image Filtering

Image filtering adalah teknik dalam pemrosesan citra digital yang bertujuan untuk memanipulasi atau memperbaiki citra melalui penerapan filter. Filter gambar dapat digunakan untuk berbagai tujuan, seperti meningkatkan kualitas gambar, menghilangkan noise (gangguan), menajamkan fitur tertentu, atau mengaburkan gambar. Proses ini melibatkan algoritma matematika yang bekerja pada setiap piksel gambar berdasarkan nilai piksel di sekitarnya. Image filtering dibagi menjadi dua jenis yaitu filter linier dan filter nonlinear. Pada filter linear bekerja dengan melakukan operasi konvolusi antara gambar dan kernel dimana nilai setiap piksel hasilnya adalah kombinasi linier dari nilai piksel di sekitarnya. Contoh umum filter linier ialah mean filter dimana filter ini akan menggantikan nilai piksel dengan rata-rata piksel di sekitarnya untuk mengurangi noise meskipun berisiko mengaburkan tepi gambar dan Gaussian filter yang akan memberikan efek smoothing lebih halus dan biasanya filter ini digunakan untuk mengaburkan gambar secara lembut. Selain itu, box blur, varian sederhana dari Gaussian blur, menghasilkan efek blur cepat tetapi kurang halus. Sedangkan filter non-linier memiliki sifat yang tidak mengikuti aturan linier dan bekerja lebih efektif dalam mempertahankan detail seperti tepi dan tekstur gambar. Salah satu filter non-linier yang paling dikenal adalah median filter yang menggantikan nilai piksel dengan median dari piksel di sekitarnya, menjadikannya sangat efektif untuk menghilangkan noise impulsif atau *salt-and-pepper* tanpa mengaburkan tepi. Bilateral filter juga termasuk dalam kategori ini dan bekerja dengan mempertahankan tepi serta tekstur melalui pendekatan yang mempertimbangkan kedekatan spasial dan intensitas warna, menjadikannya pilihan ideal dalam pengolahan citra medis dan fotografi. Selain itu, min/max filter digunakan dalam operasi morfologi untuk meningkatkan kontras lokal pada gambar.

Jenis filter non-linier yang umum digunakan mencakup beberapa metode, seperti Median Filter, yang menggantikan nilai piksel dengan median dari tetangganya, efektif dalam mengatasi noise impulsif seperti salt-and-pepper noise tanpa mengaburkan tepi. Median Filter merupakan metode penyaringan non-linier yang bekerja dengan cara mengganti nilai setiap piksel dengan nilai median dari pixel - pixel di sekitarnya dalam jendela filter tertentu. Karena median filter bersifat robust terhadap outlier, filter ini sangat baik dalam mengurangi noise impulsif atau salt-and-pepper. Noise jenis ini sering muncul akibat gangguan atau cacat piksel dalam gambar digital. Adaptive Median Filter merupakan pengembangan dari median filter, yang menyesuaikan ukuran jendela berdasarkan tingkat noise di sekitarnya, sehingga lebih fleksibel dan akurat pada gambar dengan noise yang lebih bervariasi. Dalam pemrosesan citra, filter ini sering digunakan sebagai tahap pra-pemrosesan sebelum aplikasi algoritma lain, seperti deteksi segmentasi. Proses penghilangan noise ini memungkinkan algoritma lain lanjutan bekerja lebih efisien dan menghasilkan hasil yang lebih bersih dan akurat.

2.3 Image Thresholding

Image thresholding adalah teknik dasar dalam pemrosesan citra yang bertujuan untuk memisahkan objek dari latar belakang dengan mengonversi gambar grayscale menjadi gambar biner. Thresholding bertujuan menghasilkan citra biner yang hanya memiliki dua nilai piksel, umumnya putih untuk objek dan hitam untuk latar belakang, atau sebaliknya. Dalam konteks segmentasi, thresholding dipilih karena metode ini sederhana dan cukup efektif dalam memisahkan objek dengan latar belakang yang berbeda kontras secara signifikan, yang dibahas dalam makalah ini sebagai aplikasi dasar pengolahan citra biner. Pada gambar biner, setiap piksel memiliki nilai 0 (biasanya hitam) untuk latar belakang dan 255 (biasanya putih) untuk objek, atau sebaliknya. Thresholding memerlukan nilai ambang (threshold), yang digunakan sebagai batas untuk menentukan apakah suatu piksel termasuk dalam objek atau latar belakang. Misalnya, jika nilai piksel lebih besar dari threshold, piksel tersebut akan diubah menjadi putih, sementara jika nilainya lebih kecil, akan diubah menjadi hitam. Teknik ini sangat berguna untuk mengidentifikasi dan menyoroti objek yang jelas, seperti teks pada latar belakang putih atau objek dengan kontras tinggi.

Ada beberapa metode thresholding yang dapat diterapkan, tergantung pada distribusi intensitas piksel dalam gambar. Metode global thresholding adalah pendekatan sederhana dalam image thresholding yang menggunakan satu nilai ambang (threshold) yang diterapkan pada seluruh piksel dalam gambar. Nilai ambang ini dipilih berdasarkan distribusi intensitas piksel dalam gambar dan biasanya ditentukan secara manual atau otomatis dengan mempertimbangkan histogram citra. Saat nilai piksel lebih tinggi dari ambang yang ditentukan, piksel tersebut akan diubah menjadi putih untuk menunjukkan objek, dan jika lebih rendah, akan diubah menjadi hitam untuk menandakan latar belakang. Global thresholding sangat efektif pada gambar dengan pencahayaan yang seragam, di mana kontras antara objek dan latar belakangnya jelas. Namun, metode ini

kurang optimal pada gambar dengan variasi intensitas lokal yang signifikan, karena perubahan pencahayaan atau gradasi warna di seluruh gambar dapat menyebabkan beberapa bagian objek atau latar belakang tidak tersegmentasi dengan benar. Metode **global thresholding** menggunakan satu nilai ambang yang diterapkan pada seluruh gambar, yang cocok untuk gambar dengan pencahayaan yang seragam. Sementara itu, **adaptive thresholding** menggunakan nilai ambang yang disesuaikan secara lokal untuk setiap area gambar, membuatnya ideal untuk gambar dengan latar belakang yang bervariasi. **Otsu's thresholding** adalah metode yang lebih kompleks yang dirancang untuk secara otomatis menentukan nilai ambang optimal, terutama ketika distribusi intensitas antara objek dan latar belakang berbeda secara signifikan. Dalam metode ini, nilai ambang dipilih dengan cara meminimalkan varians dalam kelas antara piksel objek dan piksel latar belakang. Otsu's thresholding menganalisis histogram citra untuk menemukan titik ambang dimana perbedaan antara piksel terang dan gelap paling jelas, sehingga mengurangi kesalahan segmentasi yang diakibatkan oleh kesamaan intensitas di dalam satu kelas. Metode ini sangat cocok untuk gambar dengan dua distribusi intensitas yang berbeda, dan sering digunakan ketika informasi mengenai latar belakang dan objek tidak dapat dipisahkan dengan jelas melalui thresholding manual atau global. Terdapat pula **double thresholding**, yang menggunakan dua nilai ambang untuk memisahkan objek dari noise atau detail kecil dalam gambar; teknik ini sering digunakan dalam deteksi tepi.

2.4 Morphological Filters

Morphological filter adalah teknik dalam pemrosesan citra yang mengandalkan operasi matematika berbasis bentuk untuk memodifikasi struktur visual pada gambar biner atau grayscale. Dengan menggunakan elemen struktural (atau kernel) berupa matriks kecil berbentuk persegi atau lingkaran, filter ini memodifikasi piksel berdasarkan pola di sekitarnya. Morphological filter terdiri dari empat operasi dasar diantaranya yaitu dilation, erosion, opening, dan closing. Dilation memperbesar atau memperluas area objek dengan mengubah atau menambah piksel pada tepi menjadi terang, yang membantu mengisi celah atau menghubungkan bagian yang terputus. Sebaliknya, erosion mengecilkan atau merampingkan objek dengan menghilangkan piksel di sekitar tepi objek, efektif untuk menghilangkan noise kecil. Opening menggabungkan erosion dan dilation secara berurutan, sehingga dapat menghilangkan noise kecil tanpa mempengaruhi bentuk asli objek, sementara closing menggabungkan dilation dan erosion secara berurutan untuk mengisi lubang-lubang kecil dan memperhalus tepi objek.

Dalam aplikasi computer vision dan analisis citra skala abu - abu, teknik ini diperluas dengan menggunakan elemen struktur (structuring element) yang memiliki bentuk geometris sederhana, seperti lingkaran atau persegi kecil, untuk menyaring detail berdasarkan ukuran atau bentuk tertentu. Metode ini bermanfaat dalam deteksi tepi dan perataan kontras dengan mempertahankan fitur geometris. Misalnya, *top-hat*

transformation, yang merupakan selisih antara citra asli dengan hasil *opening*, digunakan untuk mendeteksi puncak (*peaks*) dalam citra, sementara transformasi *bottom-hat* mendeteksi lembah. Morphological filter juga banyak digunakan dalam OCR (Optical Character Recognition) untuk memperjelas karakter atau angka yang terhubung, serta dalam analisis gambar medis untuk mengisolasi jaringan atau sel tertentu. Dengan kemampuannya dalam memperbaiki struktur visual, morphological filter memainkan peran penting dalam menghasilkan gambar yang lebih bersih dan siap untuk analisis lebih lanjut. Dalam aplikasi lanjutan, filter morfologi dapat digunakan untuk segmentasi dan pelabelan objek pada citra yang kompleks. Kemampuan *morphological filtering* untuk mempertahankan detail struktural pada objek membuatnya ideal dalam berbagai aplikasi, termasuk dalam pengenalan pola, analisis medis, dan citra satelit, di mana pemahaman terhadap bentuk dan tekstur objek sangat penting. Filter morfologi memberikan pendekatan yang kuat untuk deteksi fitur yang kompleks, seperti tepi, sudut, dan titik-titik penting, serta dalam pelacakan perubahan bentuk objek.

2.5 Object Labeling

Labeling merupakan tahap yang penting pada proses pengolahan citra. Labeling membantu untuk mempelajari ciri dari objek yang diberikan label, mulai dari besar objek, bentuk objek, sampai dengan tekstur objek atau pola objek. Object labeling adalah proses dalam pemrosesan citra dan computer vision yang bertujuan untuk mengidentifikasi dan memberi label pada objek-objek unik dalam sebuah gambar atau video. Teknik ini digunakan untuk mengelompokkan piksel yang berhubungan menjadi satu entitas atau objek tertentu, seperti orang, mobil, pohon, atau objek lainnya dalam sebuah gambar. Setiap objek unik diberi label yang berbeda, memungkinkan komputer untuk memahami jumlah, lokasi, dan bentuk objek dalam gambar. Beberapa metode utama yang sering digunakan dalam *object labeling* antara lain metode rekursif, *Depth First Search* (DFS), dan *Breadth First Search* (BFS). Metode rekursif bekerja dengan cara membanjiri area objek menggunakan *Flood Fill*, di mana piksel-piksel yang terhubung secara bertahap ditandai dengan label unik hingga seluruh objek teridentifikasi. Di sisi lain, metode DFS menggunakan *stack* untuk melacak piksel yang terhubung, sementara metode BFS menggunakan *queue*. Meskipun metode DFS dan BFS lebih cepat, mereka dapat mengalami masalah seperti pengulangan label pada objek yang berbeda atau ketidakkonsistenan label pada objek yang sama, terutama dalam citra yang kompleks.

Proses object labeling biasanya dimulai dengan tahap segmentasi, di mana citra diubah menjadi area-area yang lebih mudah dianalisis. Pada gambar biner, segmentasi ini dilakukan dengan memisahkan piksel objek dari piksel latar belakang. Setelah itu, algoritma labeling, seperti Connected Component Labeling (CCL), digunakan untuk mengidentifikasi dan memberi label pada setiap objek terpisah. Algoritma CCL memeriksa piksel-piksel yang saling terhubung (baik secara vertikal, horizontal, maupun diagonal) untuk menentukan kelompok piksel yang termasuk dalam objek yang sama. Setelah teridentifikasi, setiap kelompok piksel yang saling terhubung ini akan diberikan label unik. Hasil dari object labeling berupa gambar atau data yang menunjukkan jumlah dan posisi setiap objek dalam gambar. Ini sangat berguna dalam aplikasi yang

memerlukan informasi mendetail tentang jumlah dan lokasi objek, seperti analisis objek dalam citra satelit, pengenalan karakter dalam OCR (Optical Character Recognition), analisis mikroskopis dalam biologi, atau deteksi objek di bidang otomotif untuk sistem kendaraan otonom. Dengan object labeling, informasi yang kaya tentang citra dapat diekstraksi, membantu sistem komputer dalam pengambilan keputusan dan analisis data visual lebih lanjut.

BAB III

HASIL DAN PEMBAHASAN

3.1 Program Python

Merealisasikan sebuah program untuk mendeteksi ROI dari dalam gambar penyakit Eczema Subacute dengan menggunakan beberapa metode pengolahan citra, disimpulkan bahwa *eczema* atau eksim dapat dideteksi dan fitur-fitur dapat diekstraksi untuk analisis lebih lanjut. Berikut adalah library yang digunakan pada program. Library yang digunakan termasuk image processing dan ekstraksi fitur.

```
# Deklarasi Library

import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
import numpy as np
import scipy.ndimage as ndi
from skimage import io, color, img_as_ubyte, feature, filters, exposure,
img_as_float, morphology, measure
from pandas import DataFrame
from math import log10
import plotly.express as px
from skimage.measure import label, regionprops, regionprops_table
from skimage.transform import rotate
from skimage.draw import ellipse
import math
import pandas as pd
from datetime import datetime
```

Kode tersebut mengimpor pustaka untuk pemrosesan citra, ekstraksi fitur, dan visualisasi data. Dengan matplotlib dan plotly.express, kode ini mendukung visualisasi statis dan interaktif, sementara numpy, scipy, dan skimage digunakan untuk transformasi gambar, deteksi fitur, dan segmentasi, seperti deteksi tepi dan morfologi. Ekstraksi fitur menggunakan regionprops memungkinkan penghitungan properti area atau bentuk dalam gambar, yang kemudian diorganisir dalam format tabel dengan pandas untuk analisis lebih lanjut. Kode ini cocok untuk analisis gambar mendalam, termasuk aplikasi dalam citra medis.

```
im = io.imread('D:\PCM\ETS\DATASET\eczema-subacute-35-NoWM.jpeg')
print('Data type:', im.dtype)
print('Min. value:', im.min())
print('Max. value:', im.max())
print('shape:', im.shape)
```

```
plt.imshow(im)
plt.axis('off')
plt.show()
```

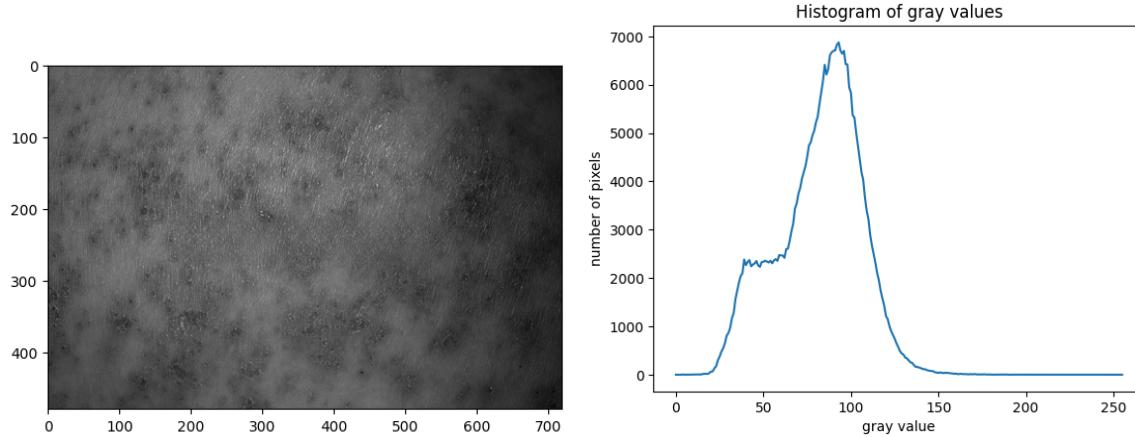
Kode tersebut digunakan untuk memanggil dataset yang akan digunakan yaitu eczema subacute-35. Pada *source code* tersebut juga akan ditampilkan tipe data, *minimal value*, *maximal value*, dan ukuran dari gambar. Berikut gambar yang dihasilkan.



```
weights = np.full((3,3), 1/9)
weights

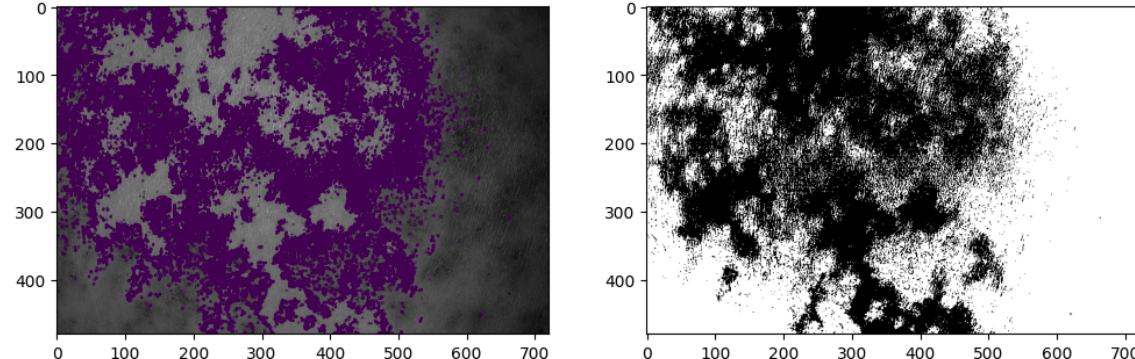
# Transformasi dari RGB ke Gray
img = color.rgb2gray(im)
img = img_as_ubyte(img)
histogram = ndi.histogram(img, min=0, max=255, bins=256)
```

Kode tersebut dimulai dengan inisialisasi filter rata-rata, di mana matriks berukuran 3x3 diisi dengan nilai 1/9 pada setiap elemen menggunakan. Filter ini sering digunakan untuk menghaluskan gambar dan mereduksi noise dengan menghitung rata-rata piksel dalam area lokal 3x3. Selanjutnya, kode melakukan transformasi gambar dari RGB ke grayscale, yang mengubah gambar berwarna menjadi intensitas abu-abu. Gambar grayscale tersebut kemudian dikonversi menjadi format uint8, menjadikannya kompatibel untuk operasi pemrosesan gambar lebih lanjut yang membutuhkan format 8-bit. Da kemudian di plot histogramnya. Berikut output yang dihasilkan.



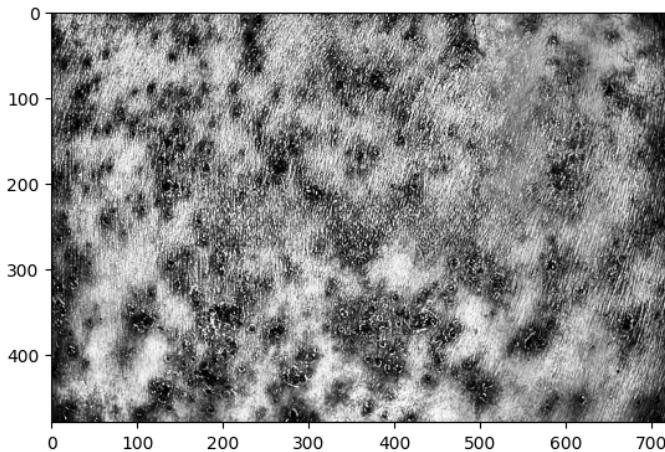
```
threshold = filters.threshold_otsu(im)
threshold
```

Setelah di grayscale, kemudian gambar akan di threshold. Kode tersebut menggunakan metode Otsu untuk menentukan nilai ambang batas (threshold) otomatis pada gambar. Nilai threshold ini dihitung secara otomatis sehingga meminimalkan varians di dalam kelas untuk kedua kelompok tersebut. Teknik ini sangat berguna dalam segmentasi gambar, terutama saat memisahkan objek dari latar belakang tanpa perlu menentukan threshold secara manual. Berikut output yang dihasilkan.



```
img_hieq = exposure.equalize_adapthist(img, clip_limit=0.9) 255
img_hieq = img_hieq.astype('uint8')
```

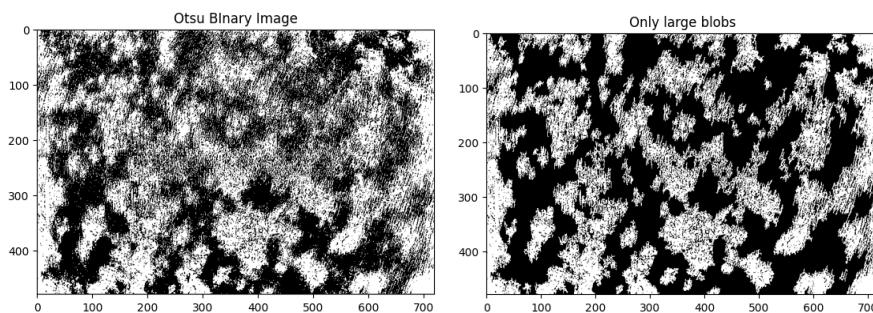
Kemudian, citra tersebut ditajamkan dengan menggunakan *Adaptive Histogram Equalization* supaya kontrasnya lebih tinggi dan gambar tidak blur. Kode tersebut digunakan untuk mengaplikasikan AHE pada gambar yang Sudah di threshold. Berikut output yang dihasilkan.



```
# Otsu thresholding
binary_image = img_hieq < filters.threshold_otsu(img_hieq)
#Remove small objects
only_large_blobs = morphology.remove_small_objects(binary_image,
min_size=100)
#Fill small holes
only_large = np.logical_not(morphology.remove_small_objects(
np.logical_not(only_large_blobs),
min_size=100))

image_segmented = only_large
```

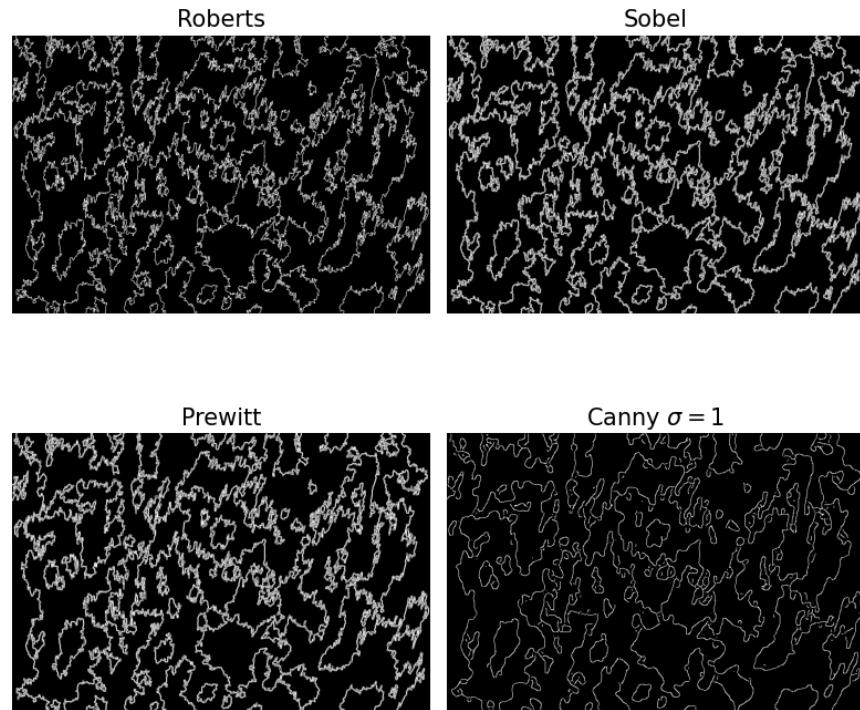
Selanjutnya menerapkan threshold pada gambar yang sudah ditajamkan menggunakan AHE dan disimpan dalam binary image. Putih jika intensitasnya di bawah threshold dan hitam jika intensitas nilai piksel diatasnya. Kemudian gambar diberikan morfologi filter, yaitu *only_large_blobs* untuk menghilangkan objek-objek kecil dari gambar sekitar corpus callosum dan variabel *fill small holes* untuk mengisi lubang kecil di dalam objek besar pada gambar. Adapun kode dan urutan gambar dibawah yaitu *Binary Image*, *Remove small Object*, dan *Fill small holes*. Berikut outputnya yang dihasilkan.



```
roberts = filters.roberts(image_segmented)
sobel = filters.sobel(image_segmented)
prewitt = filters.prewitt(image_segmented)
```

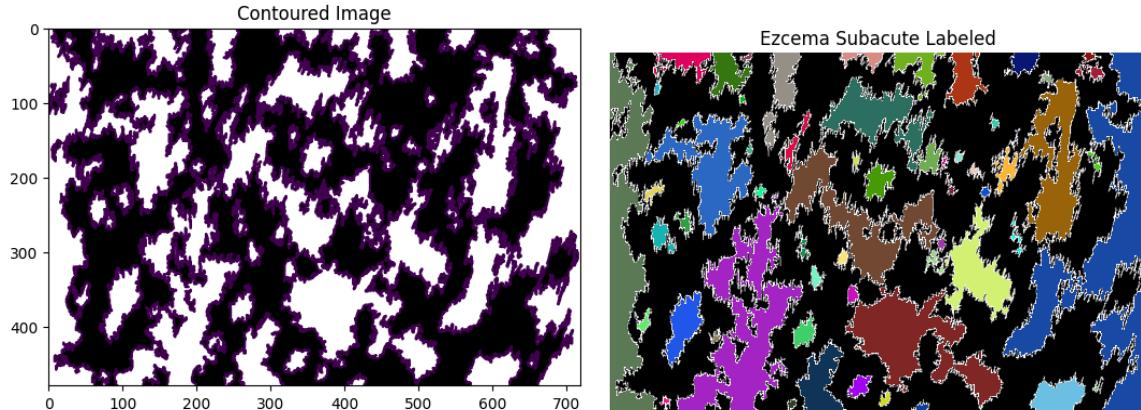
```
canny = feature.canny(image_segmented, sigma=3)
canny_image = canny
```

Pada project ini, kami menambahkan ekstraksi fitur yaitu *edge detection* menggunakan 4 metode untuk membandingkan metode mana yang lebih dalam mendeteksi tepi dari gambar eczema subacute. Berikut perbandingan secara visual untuk keempat metode deteksi tepi tersebut.



```
image_segmented = img_as_ubyte(image_segmented)
lab_image = image_segmented
# Create a random colormap
from matplotlib.colors import ListedColormap
rand_cmap = ListedColormap(np.random.rand(256,3))
labels, nlabels = ndi.label(lab_image)
```

Kode tersebut mengonversi gambar hasil segmentasi ke tipe uint8 untuk kompatibilitas pemrosesan lebih lanjut, lalu diplot dengan pengaktifan kontur. Selanjutnya yaitu membuat colormap acak dengan 256 warna untuk visualisasi yang bervariasi. Setiap objek terpisah dalam gambar dilabeli secara unik, menghasilkan variabel labels yang berisi label objek dan labels sebagai jumlah objek terdeteksi. Ini memungkinkan setiap objek dalam gambar ditampilkan dengan warna berbeda, mempermudah analisis visual dan identifikasi objek. Berikut output yang dihasilkan.



```
# Melakukan labeling pada gambar
boxes = ndi.find_objects(labels)
for label_ind, label_coords in enumerate(boxes):
    if label_coords is None:
        continue # Jika label tidak valid, lewati

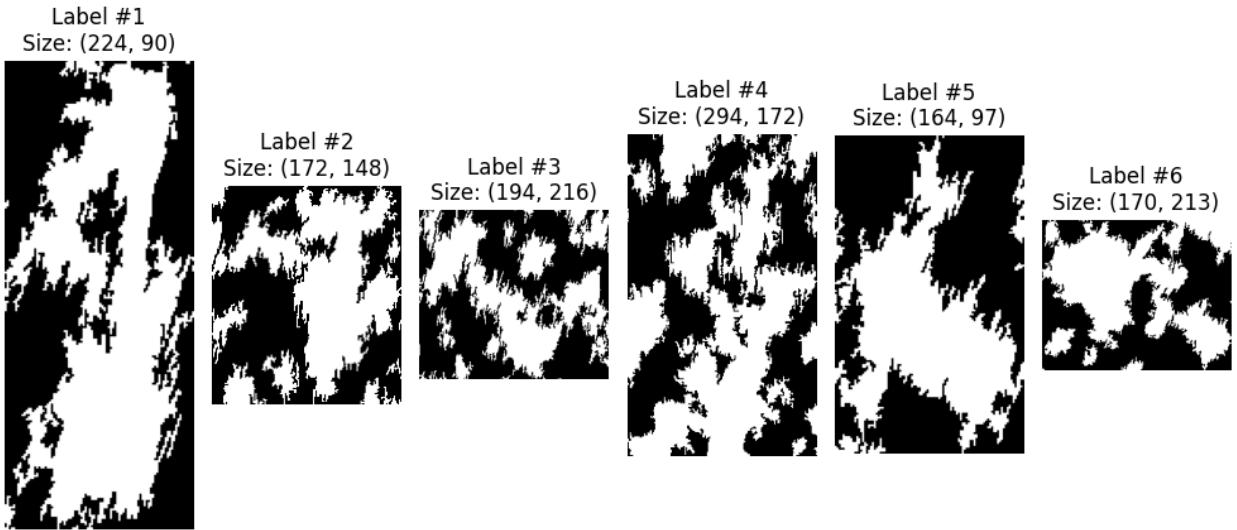
    cell = lab_image[label_coords]

    # Filter objek berdasarkan ukuran
    cell_size = np.prod(cell.shape)

    if cell_size < 5000:
        print(f'Label {label_ind} is too small! Setting to 0.')
        lab_image = np.where(labels == label_ind + 1, 0, lab_image)
```

Selanjutnya yaitu labelling. Pada labelling, sebelumnya divalidasi seberapa banyak objek yang akan di deteksi. Kemudian dari jumlah komponen/objek yang terdeteksi, diambil 6 gambar sebagai sample labeling dan dihitung ukurannya. Berikut output yang dihasilkan.

```
...
Label 57 is too small! Setting to 0.
Label 58 is too small! Setting to 0.
Label 59 is too small! Setting to 0.
There are now 13 separate components / objects detected.
```



```

#Image ROI
image = lab_image
label_img = label(image)
regions = regionprops(label_img)
label_img

fig, ax = plt.subplots()
ax.imshow(image, cmap=plt.cm.gray)

for props in regions:
    y0, x0 = props.centroid
    orientation = props.orientation
    x1 = x0 + math.cos(orientation) * 0.5 * props.minor_axis_length
    y1 = y0 - math.sin(orientation) * 0.5 * props.minor_axis_length
    x2 = x0 - math.sin(orientation) * 0.5 * props.major_axis_length
    y2 = y0 - math.cos(orientation) * 0.5 * props.major_axis_length

    ax.plot((x0, x1), (y0, y1), '-r', linewidth=2.5)
    ax.plot((x0, x2), (y0, y2), '-r', linewidth=2.5)
    ax.plot(x0, y0, '.g', markersize=15)

    minr, minc, maxr, maxc = props.bbox
    bx = (minc, maxc, maxc, minc, minc)
    by = (minr, minr, maxr, maxr, minr)
  
```

```

    ax.plot(bx, by, '-b', linewidth=2.5)

#Feature Extraction
props = regionprops_table(label_img, properties=('centroid',
                                                 'orientation',
                                                 'major_axis_length',
                                                 'minor_axis_length'))

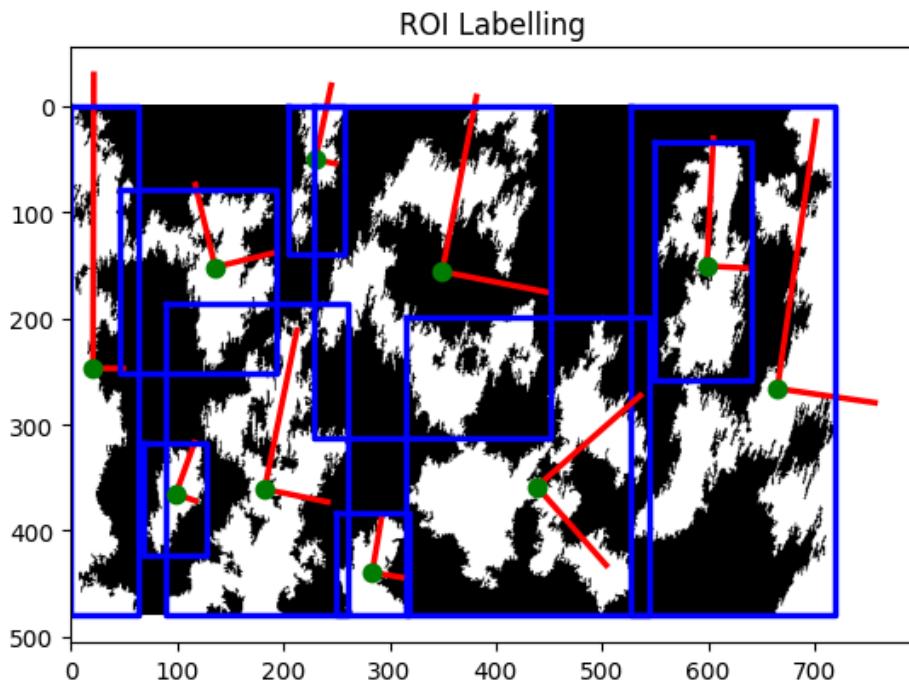
# Convert the properties to a DataFrame
df1 = pd.DataFrame(props)

# Generate a unique filename with timestamp in the Downloads folder
#timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
file_path = rf"D:\PCM\ETS\extract features.xlsx" # 'r' makes it a raw
string

# Save to Excel with a new filename each time
df1.to_excel(file_path, sheet_name='Ekstraksi_fitur_ezcema_1')

```

Langkah terakhir adalah mengekstrak fitur pada Eczema Subacute. Dengan menggunakan *library* properti di skimage, *Centroid*, *orientation*, *major_axis_length* *minor_exis_length* dapat diketahui. Fitur-fitur ini dapat digambar pada gambar dengan menggunakan teori geometri. Fitur-fitur tersebut kemudia diekstraksi dalam bentuk dataframe. Kemudian, area dari setiap objek dapat dimasukkan dengan menambahkan kolom baru. Semua data ini akan disimpan sebagai file CSV dengan nama “Extract Features”. Berikut outputnya.



Features extracted and saved to: [D:\PCM\ETS\extract features.xlsx](#)

Kode tersebut kemudian diintegrasikan ke dalam Streamlit. Desain website Streamlit yang direncanakan diawali dengan *homepage*, pengenalan mengenai eksim, dan ekstraksi eksim sesuai dengan *logic code* yang telah dibuat. Untuk mempermudah pembuatan Streamlit dan membuat seluruh anggota kelompok memiliki akses untuk mengedit *logic code* maka langkah pertama yang diambil adalah membuat Github organization dengan anggota kelompok sebagai *owner*. Langkah selanjutnya adalah membuat repository baru untuk menyimpan kode-kode sehingga bisa langsung saja deploy public app dari Github.

Langkah selanjutnya tentu saja adalah mengubah *logic code* tersebut menjadi kode yang bisa dijalankan di Streamlit. *Logic code* dibuat melalui Jupyter Notebook yang berformat ipynb, sementara kode untuk Streamlit dibuat berformat py. Pertama, tentu saja *library* yang ada harus di *declare* terlebih dahulu, dimulai dari *library* yang digunakan di *logic code* diikuti dengan *library* yang dibutuhkan dalam Streamlit.

Library	Kegunaan
os	Sistem operasi dan file management
pandas	Pengolahan dan analisis data
numpy	Komputasi numerik

matplotlib.pyplot	Visualisasi data dan grafik
streamlit	Aplikasi web Streamlit
streamlit_option_menu	Menu interaktif di Streamlit
skimage	Pengolahan citra digital
BytesIO	Manipulasi file dalam memori
time	Operasi waktu
math	Fungsi matematika dasar
scipy.ndimage (ndi)	Analisis citra digital
matplotlib.colors.ListedColormap	Colormap acak untuk visualisasi label objek
fuzzywuzzy	Pencocokan string pertanyaan

3.2 Program Streamlit

Pada bagian pertama, st.title("Pengolahan Citra Medika") menetapkan judul aplikasi yang ditampilkan di bagian atas antarmuka pengguna, memberikan konteks bahwa aplikasi ini berfokus pada pengolahan citra dalam bidang medis. Selanjutnya, sebuah dictionary bernama page_styles didefinisikan untuk menyimpan gaya CSS yang berbeda untuk setiap halaman di aplikasi.

```
# Judul Aplikasi
st.title("Pengolahan Citra Medika")

# Define background styles for each page
page_styles = {
    "Home": """
        <style>
            [data-testid="stAppViewContainer"] {
                background: linear-gradient(to right, #ff9a9e, #fecfef);
                color: white;
            }
        </style>
    """,
    "More About Eczema": """
        <style>
            [data-testid="stAppViewContainer"] {
                background: linear-gradient(to right, #fbc2eb, #a18cd1);
                color: white;
            }
        </style>
    """
}
```

Setiap entri dalam dictionary ini memiliki kunci yang mewakili nama halaman, seperti "Home", "More About Eczema", "Feature Extraction", dan "Chatbot". Nilai dari setiap kunci adalah sebuah string yang berisi kode CSS yang mengatur gaya tampilan untuk elemen dengan atribut data-testid="stAppViewContainer", yang merupakan kontainer utama di dalam aplikasi Streamlit. Gaya yang diterapkan mencakup latar belakang gradien warna yang bervariasi untuk setiap halaman; misalnya, halaman "Home" memiliki gradien dari merah muda ke putih, sedangkan halaman "More About Eczema" menggunakan gradien ungu muda. Selain itu, semua teks di dalam kontainer tersebut diatur menjadi berwarna putih untuk meningkatkan keterbacaan di atas latar belakang yang berwarna.

```

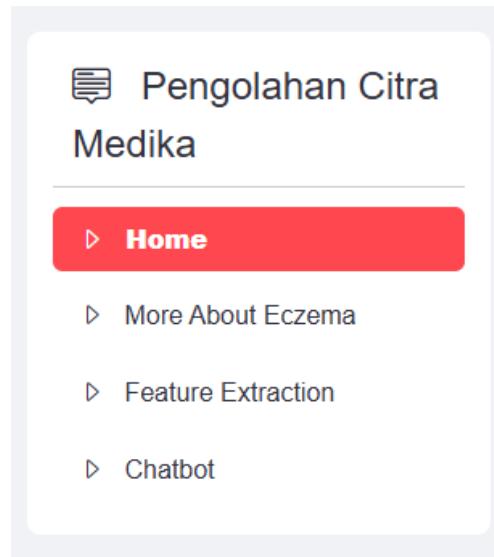
# Sidebar menu using Streamlit's sidebar
with st.sidebar:
    selected = option_menu(
        "Pengolahan Citra Medika",
        ["Home", "More About Eczema", "Feature Extraction", "Chatbot"],
        default_index=0
    )

# Apply the appropriate CSS for each page based on the selected option
st.markdown(page_styles[selected], unsafe_allow_html=True)

# Define members for photo display
members = [
    {"name": "Leony Purba 5023211013", "gif": "Leo.gif"},
    {"name": "Benedicta Sabdaningtyas 5023211029", "gif": "Bene.gif"},
    {"name": "Farhan Majid 5023211049", "gif": "Farhan.gif"},
    {"name": "Adelia Safira 5023211061", "gif": "Dela.gif"},
]

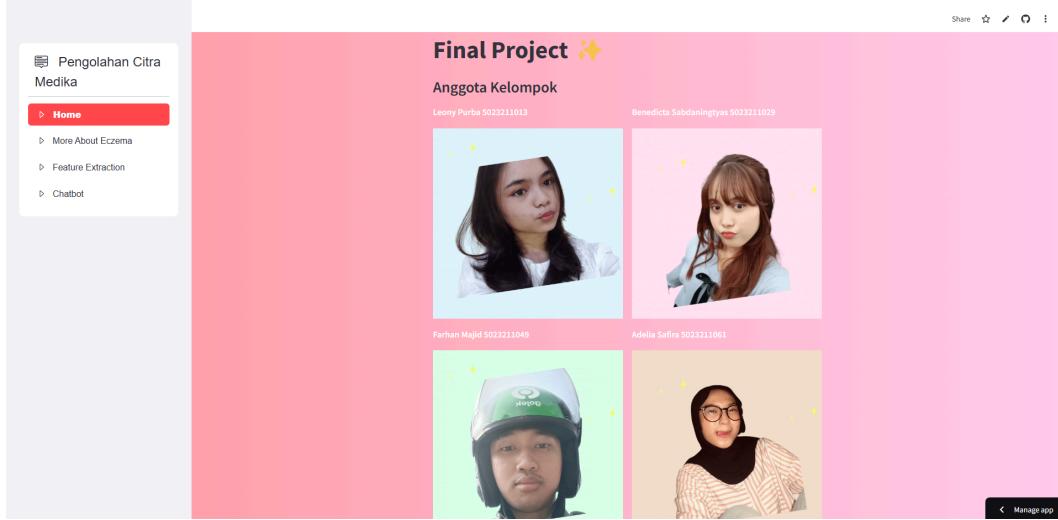
```

Dengan menggunakan `st.sidebar`, aplikasi ini menyediakan opsi menu yang terdiri dari empat halaman: "Home", "More About Eczema", "Feature Extraction", dan "Chatbot". Parameter `default_index=0` menetapkan bahwa halaman pertama, yaitu "Home", akan ditampilkan secara default saat aplikasi dimulai.



Selanjutnya, dengan menggunakan `st.markdown(page_styles[selected], unsafe_allow_html=True)`, aplikasi menerapkan gaya CSS yang sesuai untuk halaman yang dipilih berdasarkan opsi yang dipilih dari sidebar. `unsafe_allow_html=True` memungkinkan penggunaan HTML dan CSS dalam markdown, sehingga gaya yang telah ditentukan dalam

dictionary page_styles dapat diterapkan dengan benar untuk memberikan tampilan yang konsisten dan menarik. Terakhir, bagian kode ini mendefinisikan daftar anggota dalam format dictionary, yang masing-masing berisi informasi tentang nama dan file GIF yang terkait dengan anggota tersebut



```
# Home Page
if selected == "Home":
    st.title("Final Project ✨")
    st.subheader("Anggota Kelompok")

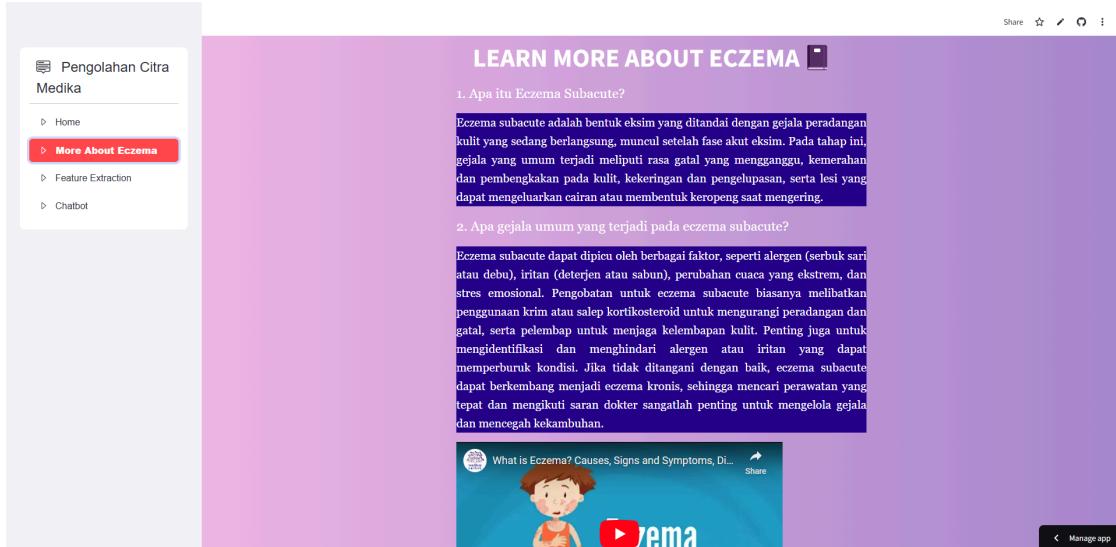
# Display members in a 2x2 grid
cols = st.columns(2)
for idx, member in enumerate(members):
    with cols[idx % 2]: # Alternate between columns
        st.write(f"**{member['name']}**")
        st.image(member["gif"])
```

Pertama, kode memeriksa apakah halaman yang dipilih adalah "Home" dan jika demikian, ia menampilkan judul utama "Final Project ✨" beserta subjudul "Anggota Kelompok", yang memberi konteks bahwa bagian ini berfokus pada tim yang bekerja dalam proyek tersebut. Untuk menampilkan anggota kelompok, aplikasi membuat dua kolom menggunakan st.columns(2), sehingga anggota dapat ditampilkan dalam format grid 2x2. Dengan menggunakan loop enumerate, kode iterasi melalui daftar anggota yang telah didefinisikan sebelumnya dan secara bergantian menempatkan informasi setiap anggota ke dalam

kolom yang sesuai. Nama anggota ditampilkan dengan format bold, diikuti oleh gambar GIF masing-masing anggota, yang menambahkan elemen visual yang menarik.

```
# Eczema Subacute Page
elif selected == "More About Eczema":
    st.markdown("<h1 style='text-align: center; color: white;'>LEARN MORE ABOUT ECZEMA </h1>", unsafe_allow_html=True)
    questions = [
        ("1. Apa itu Eczema Subacute?", "Eczema subacute adalah bentuk eksim yang ditandai dengan gejala peradangan kulit yang sedang berlangsung, muncul setelah fase akut eksim. Pada tahap ini, gejala yang umum terjadi meliputi rasa gatal yang mengganggu, kemerahan dan pembengkakan pada kulit, kekeringan dan pengelepasan, serta lesi yang dapat mengeluarkan cairan atau membentuk keropng saat mengering."),
        ("2. Apa gejala umum yang terjadi pada eczema subacute?", "Eczema subacute dapat dipicu oleh berbagai faktor, seperti alergen (serbusk sari atau debu), iritan (deterjen atau sabun), perubahan cuaca yang ekstrem, dan stres emosional. Pengobatan untuk eczema subacute biasanya melibatkan penggunaan krim atau salep kortikosteroid untuk mengurangi peradangan dan gatal, serta pembersihan untuk menjaga kelembapan kulit. Penting juga untuk mengidentifikasi dan menghindari alergen atau iritan yang dapat memperburuk kondisi. Jika tidak ditangani dengan baik, eczema subacute dapat berkembang menjadi eczema kronis, sehingga mencari perawatan yang tepat dan mengikuti saran dokter sangatlah penting untuk mengelola gejala dan mencegah rekambaran.")
    ]
    for title, description in questions:
        st.markdown(f"<p style='font-family:Georgia; color:white; font-size: 23px; text-align: left;'>{title}</p>", unsafe_allow_html=True)
        st.markdown(f"<p style='font-family:Georgia; color:white; background-color: #00008B; font-size: 20px; text-align: left;'>{description}</p>")
        st.markdown("""<iframe width='560' height='315' src='https://www.youtube.com/embed/fmurdUlmaIg' frameborder='0' allow='accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture' allowfullscreen></iframe>""", unsafe_allow_html=True)
```

Halaman More About Eczema menyediakan informasi tentang eksim subakut. Penjelasan mengenai eksim disajikan dalam format pertanyaan-jawaban menggunakan elemen markdown dan CSS untuk menampilkan dan menata tampilan teks. Di halaman ini kami juga menampilkan video Youtube terkait eksim dengan iframe.



Halaman Feature Extraction menangani pengunggahan dan pengolahan gambar untuk ekstraksi fitur eksim. Halaman ini didasarkan pada *logic code* yang telah dijelaskan sebelumnya, hanya saja dibagi menjadi beberapa bagian. Pada halaman Feature Extraction dalam kode aplikasi ini, pengguna diberikan opsi untuk mengunggah gambar eksim yang kemudian diolah melalui beberapa tahapan pemrosesan citra. Proses pengunggahan gambar dilakukan dengan `st.file_uploader`, di mana pengguna dapat memilih file dengan format .jpg atau .png. Gambar yang diunggah akan dibaca menggunakan `io.imread` dari `skimage`, lalu dikonversi menjadi grayscale dengan `color.rgb2gray` dan ke format `uint8` menggunakan `img_as_ubyte` agar kompatibel dengan algoritma pemrosesan yang lain. Setelah itu, pengguna dapat memilih beberapa tahap pengolahan gambar melalui menu navigasi horizontal yang dibuat menggunakan

option_menu, yang mencakup opsi Image, Image Processing, Edge Detection, Image Segmentation, dan Data.

```
# Eczema Feature Extraction Page
elif selected == "Feature Extraction":
    st.title("Eczema Image Feature Extraction 🧬")

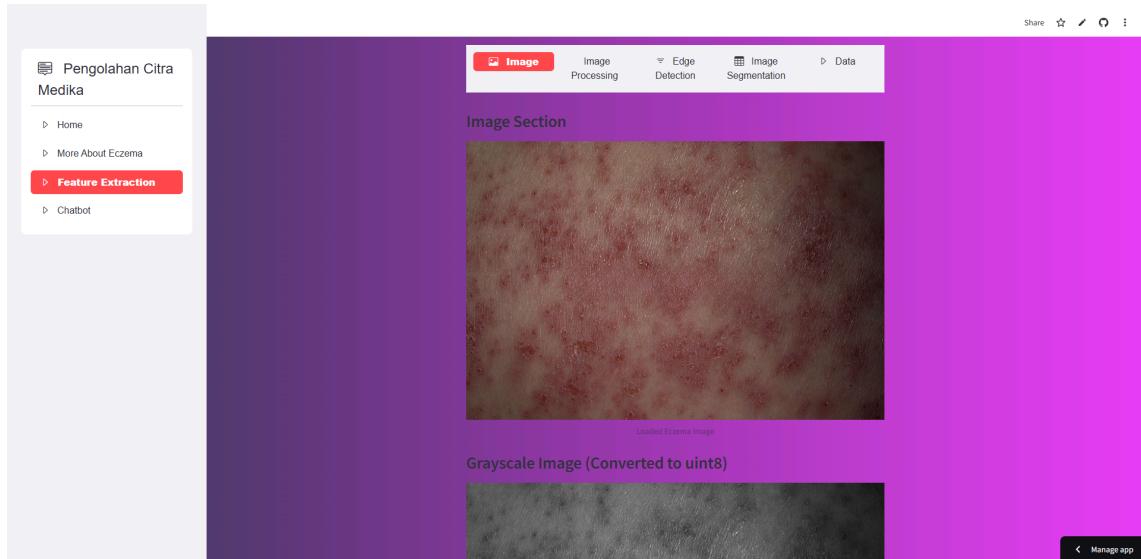
# Unggah file gambar
uploaded_file = st.file_uploader("Unggah file gambar eczema (format .jpg atau .png)", type=["jpg", "jpeg", "png"])

# Mengecek apakah file telah diunggah
if uploaded_file is not None:
    # Membaca gambar dan mengonversinya ke grayscale
    im = io.imread(BytesIO(uploaded_file.read()))
    img = color.rgb2gray(im) # Convert image to grayscale
    img = img_as_ubyte(img) # Convert to uint8

    selected2 = option_menu(
        None,
        ["Image", "Image Processing", "Edge Detection", "Image Segmentation", "Data"],
        icons=['image', 'adjust', 'filter', 'table'],
        menu_icon="cast",
        default_index=0,
        orientation="horizontal"
    )

    # Image Section
    if selected2 == "Image":
        st.subheader("Image Section")
        st.image(im, caption="Loaded Eczema Image", use_column_width=True)
        st.subheader("Grayscale Image (Converted to uint8)")
        st.image(img, caption="Grayscale Image", use_column_width=True)
```

Setelah memeriksa apakah file telah diunggah dengan if uploaded_file is not None, kode ini melanjutkan untuk membaca gambar yang diunggah dan mengonversinya menjadi format grayscale, yang sering digunakan dalam pengolahan citra karena mengurangi kompleksitas data tanpa kehilangan informasi penting yang diperlukan untuk analisis lebih lanjut. Untuk mengonversi gambar, kode menggunakan io.imread(BytesIO(uploaded_file.read())), yang memungkinkan pembacaan data gambar langsung dari objek file yang diunggah. Selanjutnya, fungsi color.rgb2gray(im) mengubah gambar menjadi grayscale, dan img_as_ubyte(img) mengonversi gambar grayscale menjadi format uint8, yang merupakan tipe data yang umum digunakan dalam pengolahan citra.



Jika pengguna memilih bagian Image, aplikasi akan menampilkan gambar asli yang diunggah dan versi grayscale-nya dengan menggunakan st.image. Pada bagian Image Processing, gambar mengalami beberapa tahapan pemrosesan citra, termasuk thresholding menggunakan metode Otsu (filters.threshold_otsu) untuk mendapatkan nilai threshold yang optimal, serta Adaptive Histogram Equalization (AHE) dengan exposure.equalize_adapthist untuk meningkatkan kontras gambar. Pada bagian ini, pengguna juga dapat melihat hasil dari proses penghapusan objek kecil dan pengisian lubang di dalam objek menggunakan morphology.remove_small_objects, yang ditampilkan menggunakan st.pyplot.

```

# Image Processing Section
elif selected2 == "Image Processing":
    st.subheader("Pre-Processing")

    # Calculate Otsu's Threshold
    threshold = filters.threshold_otsu(img)
    st.write(f"Otsu's threshold value: {threshold}")

    # Apply Adaptive Histogram Equalization (AHE)
    img_hieq = exposure.equalize_adapthist(img, clip_limit=0.9) * 255
    img_hieq = img_hieq.astype('uint8')

    # Display original image with Otsu's contour and binary thresholded image
    fig, ax = plt.subplots(ncols=2, figsize=(12, 6))
    ax[0].imshow(img, cmap='gray')
    ax[0].contour(img, levels=[threshold], colors='red')
    ax[0].set_title('Original Image with Otsu Contour')
    ax[1].imshow(img < threshold, cmap='gray')
    ax[1].set_title('Binary Image (Otsu Threshold Applied)')
    st.pyplot(fig)

    # Display AHE result
    st.subheader("Adaptive Histogram Equalization")
    fig, ax = plt.subplots(figsize=(6, 6))
    ax.imshow(img_hieq, cmap='gray')
    ax.set_title('Adaptive Histogram Equalization')
    st.pyplot(fig)

```

Langkah pertama dalam kode ini adalah menampilkan nilai ambang batas (threshold) Otsu yang dihitung secara otomatis untuk memisahkan objek dari latar belakang. Metode Otsu ini menentukan nilai threshold optimal berdasarkan distribusi piksel dalam gambar grayscale, sehingga objek yang relevan lebih mudah terlihat setelah diubah menjadi gambar biner. Kode kemudian menampilkan gambar asli bersama kontur hasil threshold Otsu, yang memperjelas batas-batas objek di dalam gambar. Selanjutnya, kode menerapkan Adaptive Histogram Equalization (AHE) untuk meningkatkan kontras pada gambar grayscale. Teknik AHE membantu menonjolkan detail yang tidak terlihat jelas pada gambar asli dengan mengatur kembali distribusi piksel dalam skala warna yang lebih dinamis. Hasil dari AHE ini ditampilkan dalam mode grayscale yang lebih kontras,

```

# Otsu Thresholding on AHE result
st.subheader("Otsu Thresholding on AHE Result")
binary_image = img_hieq < filters.threshold_otsu(img_hieq)
fig, ax = plt.subplots(figsize=(6, 6))
ax.imshow(binary_image, cmap='gray')
ax.set_title('Binary Image (Otsu Threshold on AHE)')
st.pyplot(fig)

# Remove small objects
st.subheader("Remove Small Objects")
only_large_blobs = morphology.remove_small_objects(binary_image, min_size=100)
fig, ax = plt.subplots(figsize=(6, 6))
ax.imshow(only_large_blobs, cmap='gray')
ax.set_title('Binary Image with Small Objects Removed')
st.pyplot(fig)

# Fill small holes
st.subheader("Fill Small Holes")
only_large = np.logical_not(morphology.remove_small_objects(np.logical_not(only_large_blobs), min_size=100))
image_segmented = only_large # Save result for later use

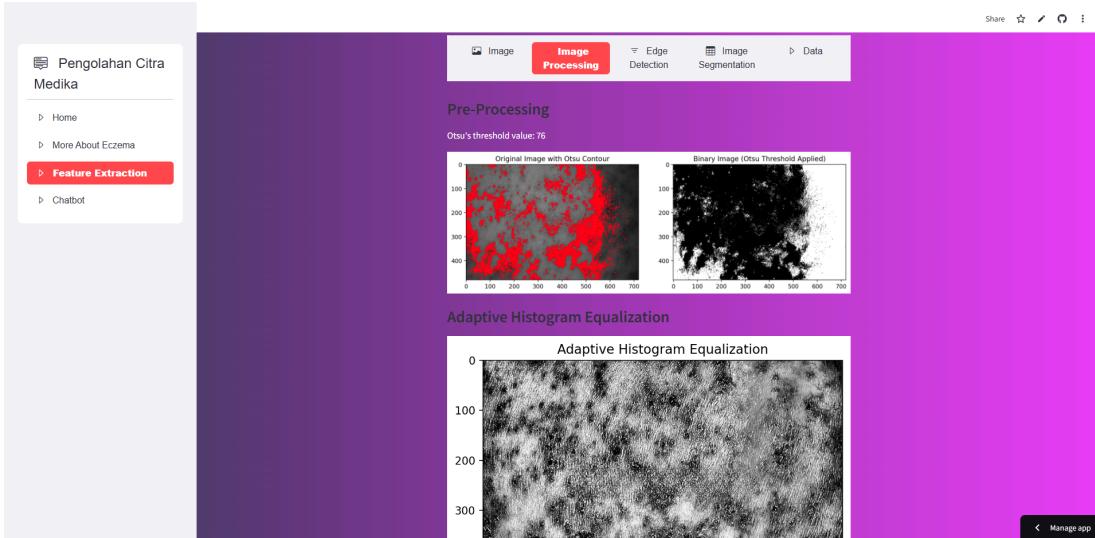
fig, ax = plt.subplots(figsize=(6, 6))
ax.imshow(image_segmented, cmap='gray')
ax.set_title('Binary Image with Small Holes Filled')
st.pyplot(fig)

```

Setelah AHE diterapkan untuk meningkatkan kontras, proses berikutnya adalah mengaplikasikan Otsu Thresholding pada hasil AHE tersebut. Dengan langkah ini, kode mengubah gambar hasil AHE menjadi gambar biner menggunakan metode Otsu untuk menentukan nilai threshold yang optimal. Hasilnya ditampilkan dalam bentuk gambar biner, di mana area yang melebihi nilai ambang batas menjadi putih dan area lainnya tetap hitam, memudahkan pemisahan objek utama dari latar belakang.

Tahap berikutnya adalah menghilangkan objek kecil dalam gambar biner untuk mengurangi noise atau komponen kecil yang tidak relevan, menggunakan fungsi `morphology.remove_small_objects()` dengan parameter `min_size=100`, yang hanya mempertahankan objek yang lebih besar dari ukuran tertentu. Ini bertujuan untuk mempertahankan objek yang signifikan secara medis atau relevan untuk analisis lebih lanjut. Setelah itu, kode melanjutkan dengan proses mengisi lubang kecil dalam objek yang tersisa, menggunakan pendekatan logika `np.logical_not()` untuk memastikan bahwa area berlubang di dalam objek diisi secara otomatis.

Hasil dari tiap tahap ditampilkan menggunakan `st.pyplot(fig)`



Selanjutnya, di bagian Edge Detection, beberapa filter deteksi tepi diterapkan pada gambar hasil segmentasi. Filter seperti Roberts, Sobel, Prewitt, dan Canny diterapkan untuk menyoroti batas-batas area dalam gambar yang kemungkinan menunjukkan ciri khas eksim. Hasil deteksi tepi dari setiap filter ditampilkan dalam grid 2x2 menggunakan matplotlib untuk membantu pengguna memahami perbedaan hasil dari tiap metode. Di bagian Image Segmentation, gambar tersegmentasi diberi label dengan `ndi.label` untuk mendeteksi dan mengidentifikasi objek dalam gambar, yang kemudian ditampilkan dengan warna acak dari ListedColormap agar setiap objek lebih mudah dibedakan. Labeling ini juga mencakup penambahan kotak pembatas, centroid, dan garis orientasi untuk setiap objek yang terdeteksi.

```

# Edge Detection Section
elif selected2 == "Edge Detection":
    st.subheader("Edge Detection Filters")

    # Re-apply segmentation if needed
    img_hieq = exposure.equalize_adapthist(img, clip_limit=0.9) * 255
    binary_image = img_hieq < filters.threshold_otsu(img_hieq)
    only_large_blobs = morphology.remove_small_objects(binary_image, min_size=100)
    only_large = np.logical_not(morphology.remove_small_objects(np.logical_not(only_large_blobs), min_size=100))
    image_segmented = only_large

    # Apply edge detection filters
    roberts = filters.roberts(image_segmented)
    sobel = filters.sobel(image_segmented)
    prewitt = filters.prewitt(image_segmented)
    canny = feature.canny(image_segmented, sigma=1)

    # Display edge detection results in a 2x2 grid
    fig, ax = plt.subplots(2, 2, figsize=(8, 8))
    ax[0, 0].imshow(roberts, cmap='gray')
    ax[0, 0].set_title('Roberts')
    ax[0, 1].imshow(sobel, cmap='gray')
    ax[0, 1].set_title('Sobel')
    ax[1, 0].imshow(prewitt, cmap='gray')
    ax[1, 0].set_title('Prewitt')
    ax[1, 1].imshow(canny, cmap='gray')
    ax[1, 1].set_title(r'Canny $\sigma=1$')

    for a in ax.flat:
        a.axis('off')

    st.pyplot(fig)

```

Bagian kode ini melakukan deteksi tepi pada gambar medis yang telah diproses sebelumnya, menggunakan empat metode berbeda: Roberts, Sobel, Prewitt, dan Canny. Setelah segmentasi gambar menggunakan Adaptive Histogram Equalization (AHE) dan Otsu Thresholding, deteksi tepi diterapkan untuk menyoroti batas atau kontur objek dalam gambar. Setiap metode memiliki karakteristik unik: Roberts dan Sobel menghitung perubahan intensitas untuk mengidentifikasi tepi, dengan Sobel yang lebih tahan terhadap noise; Prewitt memiliki fungsi serupa dengan sedikit perbedaan dalam operator; dan Canny adalah metode kompleks yang menghasilkan garis tepi yang lebih tajam dengan beberapa tahap, termasuk smoothing dan thresholding. Hasil dari keempat filter ditampilkan secara berdampingan dalam grid 2x2, memberikan pengguna visualisasi komparatif untuk memilih metode terbaik dalam analisis gambar medis.

Pengolahan Citra Medika

- Home
- More About Eczema
- Feature Extraction**
- Chatbot

Edge Detection Filters

Filter	Description
Roberts	Roberts filter, a discrete differentiation operator, computing the approximate absolute value of the gradient of a grayscale image.
Sobel	Sobel edge detection filter, a discrete differentiation operator, computing the approximate absolute value of the gradient of a grayscale image.
Prewitt	Prewitt edge detection filter, a discrete differentiation operator, computing the approximate absolute value of the gradient of a grayscale image.
Canny $\sigma = 1$	Canny edge detection filter, a multi-stage edge detector that uses a Gaussian derivative operator to find the edges in an image.

< Manage app

```

# Image Segmentation Section
elif selected2 == "Image Segmentation":
    st.subheader("Contour Image")

    # Import necessary libraries
    from skimage import exposure, filters, morphology
    from skimage.measure import label, regionprops
    import numpy as np
    import matplotlib.pyplot as plt
    from matplotlib.colors import ListedColormap
    from scipy import ndimage as ndi
    import math

    # Step 1: Image preprocessing and segmentation
    img_hieq = exposure.equalize_adapthist(img, clip_limit=0.9) * 255
    binary_image = img_hieq < filters.threshold_otsu(img_hieq)
    only_large_blobs = morphology.remove_small_objects(binary_image, min_size=100)
    only_large = np.logical_not(morphology.remove_small_objects(np.logical_not(only_large_blobs), min_size=100))
    image_segmented = only_large

    # Step 2: Display the segmented image with contour
    fig, ax = plt.subplots()
    ax.imshow(image_segmented, cmap='gray')
    ax.contour(image_segmented, levels=[0.5])
    st.pyplot(fig)

    # Step 3: Label image with random colors
    lab_image = image_segmented
    rand_cmap = ListedColormap(np.random.rand(256, 3)) # Create a random colormap
    labels, nlabels = ndi.label(image_segmented)
    labels_for_display = np.where(labels > 0, labels, np.nan)

    fig2, ax2 = plt.subplots()
    ax2.imshow(image_segmented, cmap='gray')
    ax2.imshow(labels_for_display, cmap=rand_cmap)
    ax2.axis('off')
    ax2.set_title(f'Eczema Subacute Labeled ({nlabels} labels)')
    st.pyplot(fig2)

```

Bagian ini melakukan segmentasi pada gambar medis dengan langkah-langkah preprocessing, seperti peningkatan kontras menggunakan Adaptive Histogram Equalization (AHE) dan Otsu Thresholding untuk membuat gambar biner. Setelah menghilangkan objek kecil yang tidak relevan, kontur area utama ditampilkan. Gambar yang tersegmentasi kemudian diberi label warna acak menggunakan `ndi.label()` dan `ListedColormap`, sehingga setiap objek memiliki warna unik. Visualisasi ini memudahkan identifikasi dan analisis area penting dalam gambar, khususnya untuk kondisi seperti eczema.

Pengolahan Citra Medika

- Home
- More About Eczema
- Feature Extraction**
- Chatbot

Image Processing Edge Detection **Image Segmentation** Data

Contour Image

Eczema Subacute Labeled (58 labels)

Manage app

```

# Step 4: Filtering objects and display with size labels
boxes = ndi.find_objects(labels)

# Filter objects based on size and update labels
for label_ind, label_coords in enumerate(boxes):
    if label_coords is None:
        continue # Skip if the label is invalid

    cell = lab_image[label_coords]
    cell_size = np.prod(cell.shape)

    # Remove objects that are smaller than the threshold
    if cell_size < 5000:
        lab_image = np.where(labels == label_ind + 1, 0, lab_image)

# Regenerate labels after filtering
labels, nlabels = ndi.label(lab_image)
st.write(f'Terdapat {nlabels} komponen / objek yang terdeteksi setelah filtering.')

# Step 5: Display the filtered objects with size labels
fig3, axes = plt.subplots(nrows=1, ncols=6, figsize=(10, 6))
for ii, obj_indices in enumerate(ndi.find_objects(labels)[5:11]):
    if obj_indices is not None:
        cell = image_segmented[obj_indices]
        axes[ii].imshow(cell, cmap='gray')
        axes[ii].axis('off')
        axes[ii].set_title(f'Label #{ii+1}\nSize: {cell.shape}')

plt.tight_layout()
st.pyplot(fig3)

```

Bagian kode ini menyaring objek yang tersegmentasi berdasarkan ukuran untuk mempertahankan hanya objek yang signifikan, lalu menampilkan objek yang disaring bersama label ukurannya. Pertama, kode menemukan objek dalam gambar menggunakan `ndi.find_objects(labels)`, lalu memfilter objek-objek kecil (berukuran kurang dari 5000 piksel), menghapusnya dari gambar dengan mengganti label objek tersebut menjadi nol. Setelah penyaringan, label objek diperbarui, dan jumlah total objek yang terdeteksi ditampilkan.

Pada tahap selanjutnya, kode menampilkan hingga enam objek yang tersaring dengan label ukuran, menggunakan grid 1x6. Setiap objek ditampilkan dalam grayscale, dengan judul yang menunjukkan label dan ukurannya. Visualisasi ini memungkinkan pengguna untuk melihat

secara rinci objek-objek yang signifikan setelah penyaringan berdasarkan ukuran, membantu dalam analisis lebih lanjut pada gambar medis.

```
# Step 6: Analyze and display centroid, orientation, and bounding boxes
image = lab_image
label_img = label(image)
regions = regionprops(label_img)

fig4, ax4 = plt.subplots()
ax4.imshow(image, cmap=plt.cm.gray)

for props in regions:
    y0, x0 = props.centroid
    orientation = props.orientation
    x1 = x0 + math.cos(orientation) * 0.5 * props.minor_axis_length
    y1 = y0 - math.sin(orientation) * 0.5 * props.minor_axis_length
    x2 = x0 - math.sin(orientation) * 0.5 * props.major_axis_length
    y2 = y0 - math.cos(orientation) * 0.5 * props.major_axis_length

    # Plot centroid and orientation lines
    ax4.plot((x0, x1), (y0, y1), '-r', linewidth=2.5)
    ax4.plot((x0, x2), (y0, y2), '-r', linewidth=2.5)
    ax4.plot(x0, y0, '.g', markersize=15)

    # Plot bounding box
    minr, minc, maxr, maxc = props.bbox
    bx = (minc, maxc, maxc, minc)
    by = (minr, minr, maxr, maxr)
    ax4.plot(bx, by, '-b', linewidth=2.5)

# Show the final plot with orientation and bounding boxes
ax4.set_title("Centroid, Orientation, and Bounding Boxes of Labeled Regions")
st.pyplot(fig4)
```

Kode ini menganalisis objek tersegmentasi dengan menghitung centroid, orientasi, dan bounding box untuk setiap objek dalam gambar. Menggunakan `regionprops`, kode mendapatkan koordinat pusat (centroid), orientasi, serta panjang sumbu mayor dan minor dari tiap objek. Garis orientasi digambar untuk menunjukkan arah objek, dan centroid ditandai dengan titik hijau. Selain itu, bounding box untuk setiap objek digambar dengan garis biru untuk menyoroti area objek yang terdeteksi. Hasil ini divisualisasikan pada gambar, membantu dalam memahami posisi, ukuran, dan orientasi objek yang relevan.

```

#Overlay Colormap with RGB
img_hieq = exposure.equalize_adapthist(img, clip_limit=0.9) * 255
binary_image = img_hieq < filters.threshold_otsu(img_hieq)
only_large_blobs = morphology.remove_small_objects(binary_image, min_size=100)
only_large = np.logical_not(morphology.remove_small_objects(np.logical_not(only_large_blobs), min_size=100))
image_segmented = only_large

# Label the segmented image
labels, nlabels = ndi.label(image_segmented)

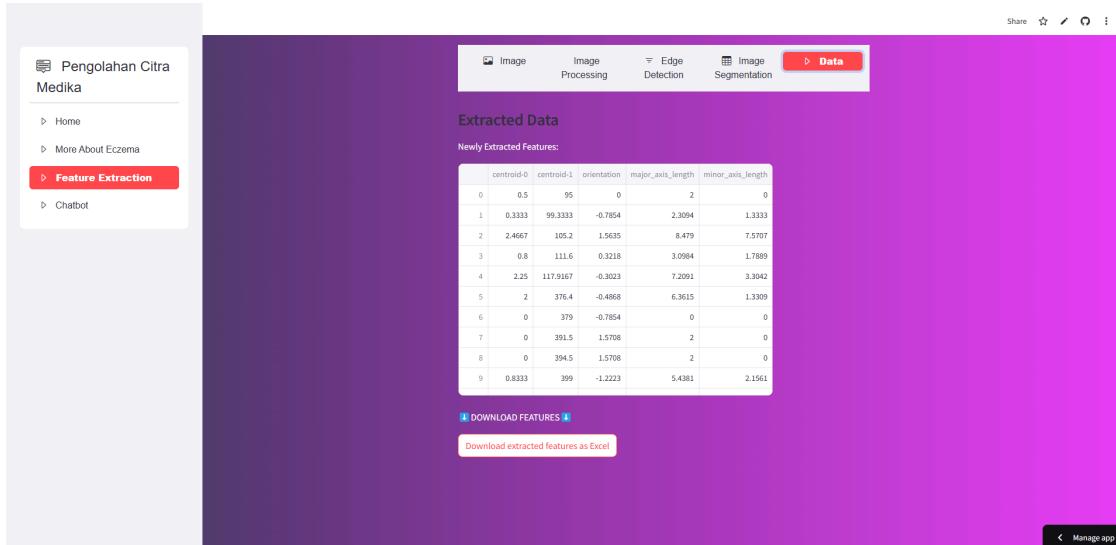
# Create a random colormap
rand_cmap = ListedColormap(np.random.rand(256, 3))

# Display the grayscale image with overlay
fig, ax = plt.subplots()
ax.imshow(img, cmap='gray') # Display grayscale image
ax.imshow(np.where(labels > 0, labels, np.nan), cmap='Paired', alpha=0.6) # Overlay colormap
ax.axis('off')
ax.set_title(f'Overlaid Eczema Subacute ({nlabels} regions)')
st.pyplot(fig)

```

Bagian kode ini menampilkan hasil akhir berupa gambar yang menunjukkan centroid, orientasi, dan bounding box dari setiap objek tersegmentasi, dilengkapi judul “Centroid, Orientation, and Bounding Boxes of Labeled Regions.” Setelah itu, kode membuat overlay pada gambar grayscale dengan warna acak untuk setiap area tersegmentasi, menggunakan colormap acak yang membantu membedakan setiap objek. Dengan `alpha=0.6`, overlay colormap diterapkan di atas gambar asli, memberikan visualisasi transparan yang menyoroti objek-objek utama dengan warna berbeda. Hasil ini memudahkan identifikasi visual terhadap area-area penting dalam gambar medis, khususnya untuk kondisi seperti eczema.

Bagian terakhir adalah Data Extraction, di mana fitur numerik dari gambar yang sudah diproses diekstraksi menggunakan regionprops_table. Fitur yang diekstraksi meliputi centroid, orientation, major_axis_length, dan minor_axis_length untuk setiap objek dalam gambar tersegmentasi. Hasil ekstraksi ini kemudian ditampilkan dalam bentuk tabel dengan st.write, dan pengguna diberikan opsi untuk mengunduh data tersebut dalam format Excel menggunakan st.download_button. Halaman Feature Extraction ini, secara keseluruhan, menyediakan alat yang lengkap bagi pengguna untuk melakukan analisis gambar eksim mulai dari pengolahan dasar hingga ekstraksi fitur yang dapat dianalisis lebih lanjut.



```
# Data Extraction Section
elif selected2 == "Data":
    st.subheader("Extracted Data")

    # Hitung properti menggunakan regionprops
    label_img = measure.label(img < filters.threshold_otsu(img))
    props = regionprops_table(label_img, properties=('centroid', 'orientation', 'major_axis_length', 'minor_axis_length'))
    df_new = pd.DataFrame(props)
    st.write("Newly Extracted Features:")
    st.write(df_new)

    # Path untuk menyimpan file Excel baru
    excel_path = r"extract_features.xlsx" # Simpan di direktori kerja saat ini

    # Simpan data baru ke Excel (mode 'w' untuk membuat file baru)
    with pd.ExcelWriter(excel_path, mode='w', engine="openpyxl") as writer:
        df_new.to_excel(writer, index=False, sheet_name='New Features')

    # Tombol unduh untuk file Excel yang baru dibuat
    st.write("⬇️ DOWNLOAD FEATURES ⬇️")
    with open(excel_path, 'rb') as file:
        st.download_button(
            label="Download extracted features as Excel",
            data=file.read(),
            file_name="extract_features.xlsx", # Nama file untuk unduhan
            mime="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"
        )

```

Bagian kode ini mengekstrak fitur dari objek tersegmentasi dalam gambar, seperti centroid, orientasi, serta panjang sumbu mayor dan minor, menggunakan `regionprops_table`. Fitur-fitur ini disimpan dalam format DataFrame menggunakan `pandas` dan ditampilkan di aplikasi sebagai tabel yang mudah dibaca. Data yang telah diekstrak kemudian disimpan dalam file Excel bernama `extract_features.xlsx`, memungkinkan pengguna untuk mengunduhnya langsung melalui tombol unduhan di antarmuka Streamlit. Hal ini memberi pengguna akses mudah terhadap data terperinci yang dapat digunakan untuk analisis lanjutan atau dokumentasi.

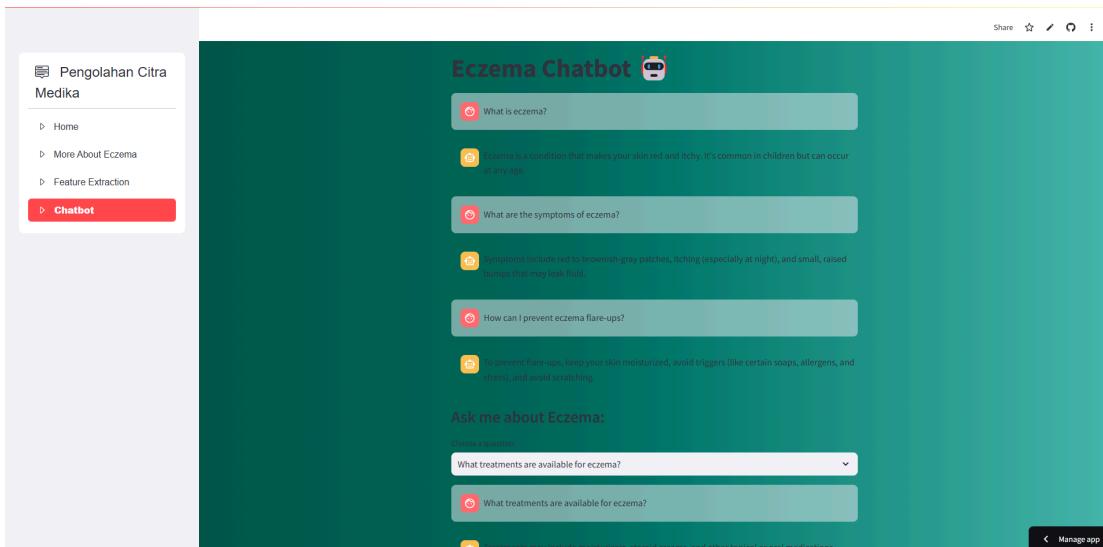
```

# Eczema Chatbot Page
elif selected == "Chatbot":
    st.title("Eczema Chatbot 🤖")

if "messages" not in st.session_state:
    st.session_state["messages"] = []

```

Bagian kode ini menginisialisasi halaman Chatbot Eczema pada aplikasi, di mana pengguna dapat mengakses informasi seputar eczema melalui interaksi chatbot. Halaman ini memanfaatkan `st.session_state` untuk menyimpan riwayat pesan selama sesi berlangsung, memastikan bahwa percakapan pengguna tidak hilang saat halaman diperbarui. Selanjutnya, `qa_pairs` berisi kumpulan pasangan pertanyaan dan jawaban (Q&A) tentang eczema yang sering ditanyakan, dengan tujuan memberikan respons otomatis yang relevan setiap kali pengguna memilih atau memasukkan pertanyaan yang ada dalam database. Chatbot ini memungkinkan pengguna untuk belajar lebih lanjut tentang eczema dengan cara yang interaktif dan terstruktur.



Pada halaman Chatbot dalam kode aplikasi ini, pengguna dapat berinteraksi dengan chatbot yang dirancang untuk menjawab pertanyaan seputar eksim. Kode dimulai dengan menetapkan `st.session_state["messages"]` untuk menyimpan riwayat percakapan, sehingga percakapan tetap terlihat meskipun halaman dimuat ulang. Daftar pertanyaan dan jawaban umum terkait eksim disimpan dalam `qa_pairs`, yang berfungsi sebagai basis pengetahuan bagi chatbot. Setiap kali pengguna memilih atau mengetikkan pertanyaan, chatbot akan mencari jawaban di `qa_pairs` dan menampilkannya dalam bentuk percakapan interaktif. Kode ini menggunakan elemen `st.chat_message` untuk mengelola tampilan pesan, di mana setiap pesan yang dikirim oleh pengguna dan jawaban dari chatbot ditampilkan dengan format percakapan yang mudah dibaca. Selain itu, terdapat fungsi `getSuggestions` yang dimaksudkan untuk memberikan saran pertanyaan serupa kepada pengguna, meskipun di sini fungsinya belum diimplementasikan.

secara penuh. Pengguna dapat memilih pertanyaan dari dropdown st.selectbox atau mengetik pertanyaan secara langsung. Jika pertanyaan ditemukan di question and answer_pairs, chatbot akan menampilkan jawabannya; jika tidak, akan muncul pesan bahwa chatbot tidak memiliki jawaban untuk pertanyaan tersebut.

```
# Function to get similar questions - only defined on Chatbot page
def get_suggestions(query, choices, limit=5):
    # Get the closest matches
    suggestions = prc.extract(query, choices, limit=limit)
    return [suggestion[0] for suggestion in suggestions]

# Display chat messages from history on app rerun
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.markdown(message["content"])

# Display predefined questions for user to choose
st.write("### Ask me about Eczema:")
user_question = st.selectbox("Choose a question:", [""] + list(qa_pairs.keys()))

# If user selects a question, display answer
if user_question:
    # Display user question in chat
    with st.chat_message("user"):
        st.markdown(user_question)
    # Add user question to chat history
    st.session_state.messages.append({"role": "user", "content": user_question})

    # Display bot response
    bot_response = qa_pairs.get(user_question, "I'm sorry, I don't have an answer for that question.")
    with st.chat_message("assistant"):
        st.markdown(bot_response)
    # Add bot response to chat history
    st.session_state.messages.append({"role": "assistant", "content": bot_response})
```

Kode ini menambahkan fungsionalitas pada chatbot eczema dengan fitur pemilihan pertanyaan dan riwayat percakapan. Fungsi `get_suggestions` menggunakan pustaka pencocokan teks untuk memberi saran pertanyaan yang mirip dengan input pengguna, memudahkan mereka menemukan jawaban yang relevan jika tidak ada pertanyaan yang persis sama. Riwayat percakapan disimpan di `st.session_state["messages"]`, sehingga pengguna dapat melihat kembali pesan sebelumnya meskipun halaman diperbarui. Pengguna dapat memilih pertanyaan dari dropdown `selectbox`, yang berisi pertanyaan-pertanyaan yang umum tentang eczema dari `qa_pairs`. Jika pengguna memilih pertanyaan, pertanyaan tersebut ditampilkan di antarmuka, dan chatbot membalsas dengan jawaban yang sudah ada di `qa_pairs`. Jika jawaban untuk pertanyaan tersebut tidak tersedia, bot memberikan pesan respons default. Riwayat pertanyaan dan jawaban disimpan secara berurutan, menciptakan pengalaman interaksi yang lancar dan responsif.

BAB IV

KESIMPULAN

Kesimpulannya, Proses Ekstraksi Fitur dari citra Eczema Subacute melibatkan *filtering* gambar diawal menggunakan filter median, penerapan otsu threshold untuk memisahkan objek dari latar belakang, penghilangan lubang kecil dalam objek besar untuk mendapatkan segmentasi yang bersih dari Eczema Subacute sehingga setiap fitur dapat diekstrak dan disimpan dalam bentuk cvs atau excel dalam mempermudah perhitungan dan penggunaan data lebih lanjut. Aplikasi Streamlit ini mencakup beberapa halaman yang dirancang untuk membantu pengguna dalam memahami dan menganalisis eksim secara komprehensif. Halaman Home menampilkan anggota tim dengan desain grid sederhana, sementara More About Eczema memberikan informasi mendalam tentang eksim melalui teks dan video edukatif. Halaman Feature Extraction memungkinkan pengguna mengunggah gambar eksim untuk diolah lebih lanjut menggunakan teknik pemrosesan citra seperti konversi grayscale, thresholding, deteksi tepi, dan segmentasi, sehingga fitur eksim dapat diidentifikasi dengan lebih jelas. Terakhir, Chatbot menawarkan fitur interaktif untuk menjawab pertanyaan terkait eksim melalui daftar Q&A, memberikan akses cepat pada informasi tambahan atau pertanyaan yang mungkin muncul. Kombinasi berbagai halaman ini menjadikan aplikasi sebagai alat yang praktis, edukatif, dan interaktif untuk membantu pengguna mengenali karakteristik eksim dan meningkatkan pemahaman pengguna.

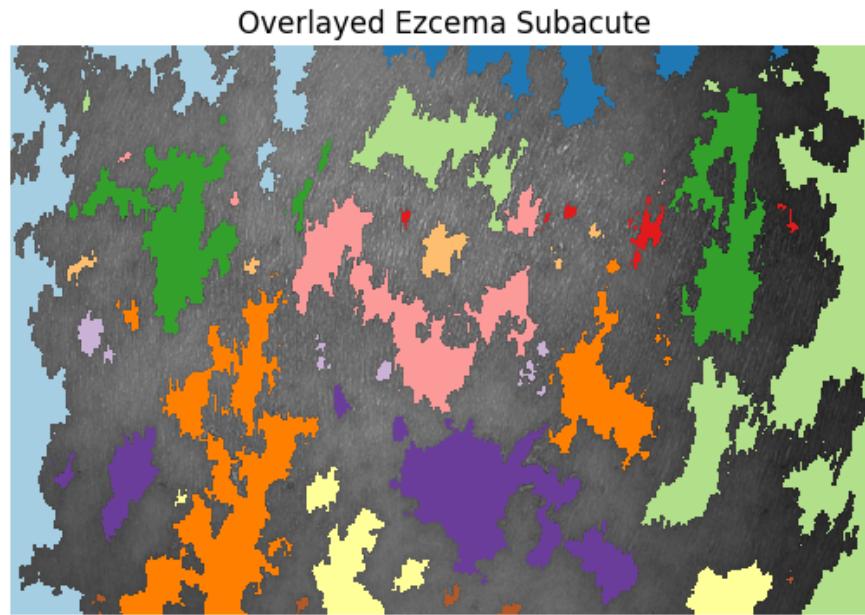
LAMPIRAN

1. Pembagian Jobdesk

Nama	Jobdesk	
	Pre-Demo	Laporan
Leony Purba	Program	Hasil dan Pembahasan Program
Benedicta Sabdaningtyas Pratita Pratanjana	Streamlit dan Github	Pendahuluan, Hasil dan Pembahasan
Farhan Majid Ibrahim	Streamlit dan Github	Hasil dan Pembahasan, Tinjauan Pustaka
Adelia Safira	Streamlit	Hasil dan Pembahasan, Kesimpulan

2. Demo-Day

Pada 31 oktober, dilaksanakan demo di ruang AJ-203. Setelah selesai mempresentasikan program dan streamlit, bu Nada meminta untuk overlay gambar yang sudah di grayscale menggunakan random color oleh library ListedColormap . Namun, secara real time output yang kita dapatkan salah karena kode yang diprogram yaitu gambar asli yang hanya di grayscale. Sehingga tekstur dan bentuk setiap object atau nodul pada citra eksim tidak ter-overlay secara sempurna. seharusnya gambar yang diproses adalah setelah melakukan segmentasi gambar sehingga tepi, bentuk, dan tekstur dari setiap gambar nodul dapat terdeteksi dan diberi warna oleh random color. Berikut output yang sudah benar.



Adapun *source code* untuk overlay citra grayscale yaitu :

```
lab_image = image_segmented

# Create a random colormap
from matplotlib.colors import ListedColormap
rand_cmap = ListedColormap(np.random.rand(256,3))
labels, nlabels = ndi.label(lab_image)

labels_for_display = np.where(labels > 0, labels, np.nan)
plt.imshow(img, cmap='gray')
plt.imshow(labels_for_display, cmap='Paired')
plt.axis('off')
plt.title(' Overlaid Ezcema Subacute'.format(nlabels))
plt.show()
```

3. Fitur yang sudah di ekstrak dan disimpan di excel dapat dilihat pada link berikut :

extract features.xlsx