# BAB III

# ANALISIS DAN PERANCANGAN SISTEM

Bab ini akan bertujuan untuk membahas mengenai analisis dan perancangan sistem untuk klasifikasi kebakaran hutan di pulau Sumatera. Tahapan analisis dan perancangan sistem terdiri dari pengumpulan data citra satelit, …., pengolahan citra satelit, dan pembuatan serta pelatihan model CNN untuk klasifikasi kebakaran hutan.

## Pengeksplorasian dan Pengunduhan Data Citra Satelit

Pengeksplorasian dan pengunduhan data citra satelit akan dilakukan dengan menggunakan Python API *package* yang bernama ee. Pengeksplorasian dan pengunduhan data dilakukan berdasarkan tanggal terjadinya peristiwa kebakaran hutan melalu *dataset* yang bernama *data-total.csv* yang diperoleh dari **(jangan lupa isi ini)**. Terdapat beberapa fungsi dalam ee *package* yang digunakan untuk mengembalikan citra satelit yakni (“API Reference  |  Google Earth Engine  |  Google Developers” n.d.):

* + - 1. ee.Authenticate()

Fungsi di atas bertujuan untuk melakukan autentikasi pada *user* sebelum menggunakan *server* dari Google Earth Engine. Proses autentikasi memerlukan akun G-mail dan kemudian Google akan memberikan sebuah *authentication token*.

* + - 1. ee.Initialize()

Fungsi di atas bertujuan untuk menginisialisasi API dari Google Earth Engine untuk digunakan.

* + - 1. ee.Date(ComputedObject|Date|Number|String **date**, String|optional **tz**)

Fungsi di atas bertujuan untuk membuat sebuah Date Object untuk menentukan tanggal dari citra satelit yang ingin dilihat atau untuk keperluan lainnya. *Parameter* **date** menerima nilai tanggal dalam bentuk *string*, ISO Date *string*, JavaScript Date atau ComputedObject. *Parameter* **tz** menerima nilai berupa *timezone*.

* + - 1. ee.ImageCollection(ComputedObject|Image|List **args**)

Fungsi di atas bertujuan untuk mengembalikan kumpulan citra satelit berdasarkan *argument* yang dimasukan ke dalam **args**. Parameter ini bisa menerima argumen berupa *string* yang akan dianggap sebagai nama koleksi citra satelit, kumpulan gambar yang nantinya digunakan untuk menghasilkan gambar baru, sebuah gambar, dan *computed object* yang akan diinterpretasikan sebagai sebuah koleksi.

* + - 1. ee.Geometry.Rectangle(List **coords**, Projection**proj**, Boolean **geodesic**, Boolean **evenOdd**)

Fungsi di atas bertujuan untuk membuat sebuah ee.Geometry yang mendeskripsikan sebuah persegi. Parameter **coords** akan menerima sebuah argument berjenis List dengan urutan xMin, yMin, xMax, dan yMax. Parameter lainnya bersifat opsional dan tidak digunakan dalam kasus ini.

* + - 1. ee.Date.advance(Float **delta**, String **unit**, String **timeZone**)

Fungsi di atas bertujuan untuk membuat Date Object baru dengan menambahkan sebuah nilai yang diberikan. Parameter **delta** menerima nilai negatif atau positif yang akan ditambahkan pada tanggal. Parameter **unit** menerima nilai *string* yang menunjukkan satuan yang digunakan, misalnya 'year', 'month' 'week', 'day', 'hour', 'minute', atau 'second'. *Parameter* **timeZone** menerima nilai berupa *timezone* dalam bentuk *String*.

* + - 1. ee.ImageCollection.FilterDate(Date|Number|String **start**, Date|Number|String|optional **end**)

Fungsi di atas bertujuan untuk melakukan *filter* pada sebuah koleksi citra satelit sesuai dengan batas tanggal yang dimasukan. *Parameter* **start** merupakan tanggal awal sedangkan *parameter* **end** merupakan tanggal akhir.

* + - 1. ee.ImageCollection.FilterBounds(ComputedObject|FeatureCollection|Geometry **geometry**)

Fungsi di atas hanya bertujuan untuk melakukan *filter* pada sebuah koleksi citra satelit sesuai dengan koordinat (*longitude* dan *latitude*) batas yang sudah ditentukan. Parameter **geometry** menerima sebuah argument berupa nilai batasan koordinat dari sebuah gambar citra satelit yang ingin diambil.

* + - 1. ee.ImageCollection.map(Function **algorithm**, Boolean **dropNulls**)

Fungsi di atas bertujuan untuk menjalankan sebuah algoritma atau fungsi yang dimasukan terhadap keseluruhan koleksi citra satelit. Parameter **algorithm** akan menerima sebuah *Function* yang nantinya akan diimplementasikan kepada keseluruhan koleksi citra satelit.

* + - 1. ee.ImageCollection.median()

Fungsi di atas bertujuan untuk melakukan reduksi pada sekumpulan koleksi citra satelit menjadi satu gambar dengan menghitung median dari *pixel* sekumpulan koleksi berdasarkan nama *band*.

* + - 1. Image.normalizedDifference(List **bandNames**)

Fungsi di atas bertujuan untuk melakukan kalkulasi perhitungan *normalized difference* di antara kedua *band* dari citra satelit. Rumus perhitungannya adalah (nilai\_pertama – nilai\_kedua) / (nilai\_pertama + nilai\_kedua). Parameter **bandNames** akan menerima sebuah nilai berjenis *List* dalam bentuk seperti ini [nilai \_pertama, nilai\_kedua].

* + - 1. ee.Image.subtract(Image **image2**)

Fungsi di atas bertujuan untuk mengurangi nilai dari setiap bands yang sama pada citra satelit yang menggunakan fungsi tersebut dan **image2**. Parameter **image2** akan menerima nilai jenis data ee.Image untuk dikurangi dnegan ee.Image yang menggunakan fungsi *subtract()*.

* + - 1. ee.Image.rename(List **var\_args**)

Fungsi di atas hanya bertujuan untuk mengubah nama *band* yang terdapat pada citra satelit. Parameter **var\_args** akan menerima nilai berupa sekumpulan nama *band* yang baru di mana nama itu akan digunakan untuk menggantikan nama *band* lama yang terdapat pada citra satelit.

* + - 1. ee.Image.updateMask(Image **mask**)

Fungsi di atas bertujuan untuk nemperbarui *masik* dari citra satelit di segala posisi di mana *mask* yang ada tidak nol nilainya. Parameter **mask** akan berupa *mask* baru untuk citra satelit biasanya berupa *floating-point* dengan nilai berada di antara 0 sampai dengan 1.

* + - 1. ee.Image.addBands(Image **srcImg**, List **names**, Boolean **overwrite**)

Fungsi di atas bertujuan untuk menambahkan *band* pada citra satelit. Parameter **srcImg** akan menerima sebuah citra satelit yang terdiri dari kumpulan *band* yang akan di*­-copy*. Parameter *names* bertujuan untuk memilih nama *band* apa saja yang akan di-*copy*. Parameter **overwrite** akan menerima nilai *Boolean* antara *true* atau *false*. Jika *true*, *band* dari **srcImg** akan menimpa *band* dengan nama yang sama pada citra satelit yang menggunakan fungsi ini. Jika *false*, *band* baru akan diganti namanya dengan sufiks numerik (foo ke foo\_1 kecuali foo\_1 ada, lalu foo\_2 kecuali jika ada, dan lain-lain).

Terdapat beberapa fungsi dalam folium *package* yang digunakan untuk menampilkan citra satelit yang dikembalikan oleh Google Earth Engine, yakni sebagai berikut:

1. folium.raster\_layers.TileLayer(**tiles**='OpenStreetMap', **min\_zoom**=0, **max\_zoom**=18, **max\_native\_zoom**=None, **attr**=None, **detect\_retina**=False, **name**=None, **overlay**=False, **control**=True, **show**=True, **no\_wrap**=False, **subdomains**='abc', **tms**=False, **opacity**=1, **\*\*kwargs**)

Fungsi di atas bertujuan untuk membuat *tile layer* yang kemudian akan ditambahkan atau dimasukan ke dalam sebuah Map. Parameter **tiles** memiliki nilai awal 'OpenStreetMap' (*built-in tiles*) yang bisa diubah dengan memasukan ­*costum tileset* dalam bentuk Uniform Resource Locator (URL). Parameter **attr** akan menerima sebuah data *String* untuk memberikan informasi tambahan mengenai *tile layer* yang biasanya berupa URL. Parameter **name** merepresentasikan nama dari layer ditampilkan. Parameter **overlay** menerima nilai *Boolean* untuk menentukan apakah layer ini akan dijadikan sebagai lapisan utama(*base layer*) atau lapisan opsional (*optional layer*).

1. folium.folium.Map(**location**=None, **width**='100%', **height**='100%', **left**='0%', **top**='0%', **position**='relative', **tiles**='OpenStreetMap', **attr**=None, **min\_zoom**=0, **max\_zoom**=18, **zoom\_start**=10, **min\_lat**=- 90, **max\_lat**=90, **min\_lon**=- 180, **max\_lon**=180, **max\_bounds**=False, **crs**='EPSG3857', **control\_scale**=False, **prefer\_canvas**=False, **no\_touch**=False, **disable\_3d**=False, **png\_enabled**=False, **zoom\_control**=True, **\*\*kwargs**)

Fungsi di atas bertujuan untuk menampilkan sebuah peta dengan Folium() dan Leaflet.js (*open source project* untuk pembuatan aplikasi pemetaan *web*site). Parameter **location** akan menerima sebuah List yang berupa nilai *latitude* dan *longitude*. Parameter **zoom\_start** hanya berfungsi untuk menunjukan tingkat pembesaran awal yang di mana semakin besar maka semakin dekat. Parameter lainnya bersifat opsional namun dalam kasus ini tidak akan digunakan.

1. folium.map.Marker(**location**=None, **popup**=None, **tooltip**=None, **icon**=None, **draggable**=False, **\*\*kwargs**)

Fungsi di atas bertujuan untuk menampilkan tanda (*marker*) pada peta. Parameter **location** akan menerima sebuah List yang berupa nilai *latitude* dan *longitude*. Parameter **popup** akan menerima sebuah argument berjenis *String*, HTML, dan sebagainya sebagai label.

Kemudian untuk pengunduhan citra satelit dapat dilakukan dengan menggunakan fungsi ee.Image.getDownloadURL(Object **params**, Function callback) yang bertujuan untuk mengembalik sebuah *Object* atau *String* berupa *link* untuk mendownload data citra satelit. Parameter **params** akan menerima sebuah argument bertipe *Object* yang dalam kasus ini adalah berupa *dictionary* dengan masing-masing nilai, yakni:

1. **region** yang menunjukkan wilayah atau daerah yang perlu diunduh biasanya dalam bentuk koordinat atau dalam kasus ini menggunakan ee.Geometry.Rectangle().
2. **bands** yang menunjukkan *band* apa saja yang perlu diunduh.
3. **scale** yang menunjukkan skala dari citra satelit yang diunduh.
4. **format** yang menunjukkan format dari *file* yang akan diunduh, seperti “ZIPPED\_GEO\_TIFF”, “GEO\_TIFF”, “NPY” dan sebagainya.

## Perhitungan *Normalized Burned Ratio* (NBR)

Perhitungan NBR melibatkan *band* yang merepresentasikan NIR dan SWIR dari citra satelit yang dikumpulkan. Rumus perhitungan dari NBR seperti yang dijelaskan dalam BAB II adalah sebagai berikut:

Implementasi dari komputasi telah tersedia dalam fungsi yang terkandung dalam ee *package*, yakni ee.Image.normalizedDifference(List **bandNames**) di mana fungsi ini bertujuan untuk melakukan kalkulasi perhitungan *normalized difference* di antara kedua *band* dari citra satelit. Rumus perhitungannya adalah (nilai\_pertama – nilai\_kedua) / (nilai\_pertama + nilai\_kedua). Parameter **bandNames** akan menerima sebuah nilai berjenis *List* dalam bentuk seperti ini [nilai \_pertama, nilai\_kedua].

Penggunaan Google Earth Engine API untuk bahasa pemrogramana Python difasilitasi dengan menggunakan *package* ee. Penggunaan Google Earth Engine API memudahkan penarikan data citra satelit dari satelit yang mengorbit dunia. Terdapat

Untuk mencantumkan kode program, untuk membedakan teks dengan bagian tulisan atau paragraf yang lain, gunakan font yang bertipe **monotype** (misal: **Courier New**, **MS Gothic**, atau **Consolas**, ukuran font 10pt).

Untuk menyebut nama identifier (misal: variabel, konstanta, method, function, class, dsb.) pada suatu paragraf, gunakan font yang bertipe **monotype dengan cetak tebal (bold).**

Contoh:



Berikut fungsi OpenCV yang akan digunakan pada tahap *enhancement*:

1. *Grayscale Image*

Mat **imread** (const String& **filename**, int **flags**=IMREAD\_COLOR)

**filename** merupakan nama gambar yang akan di-*input;* **flags =** CV\_LOAD\_IMAGE\_GRAYSCALE, untuk selalu mengkonversi citra *input* ke dalam *grayscale.*

1. *Mean Filter*

void **blur**( InputArray **src**, OutputArray **dst**, Size **ksize**, Point **anchor** = Point(-1,-1), int **borderType** = BORDER\_DEFAULT)

**src** adalah *input* gambar; **dst** adalah *output* gambar; **ksize** adalah ukuran *kernel* *blur*; **anchor adalah titik pusat kernel (*default* adalah Point (-1,-1); borderType adalah metode ektrapolasi piksel (*default* adalah** BORDER\_DEFAULT)**.**

1. ***Gaussian Filter***

void **GaussianBlur**(InputArray **src**, OutputArray **dst**, Size **ksize**, double **sigmaX**, double **sigmaY**=0, int **borderType**= BORDER\_DEFAULT)

**src** adalah *input* gambar; **dst** adalah *output* gambar; **ksize** adalah ukuran *kernel gaussian*; **sigmaX** adalah standar deviasi *kernel* arah X (default adalah 0); **sigmaY** adalah standar deviasi *kernel* arah Y (default adalah 0); **borderType** adalah metode ektrapolasi piksel (*default* adalah BORDER\_DEFAULT).

1. *Median Filter*

void **medianBlur**(InputArray **src**, OutputArray **dst,** int **ksize**)

**src** adalah *input* gambar; **dst** adalah *output* gambar; **ksize** adalah ukuran *kernel median*.

1. *Bilateral Filter*

void **bilateralFilter**(InputArray **src**, OutputArray **dst**, int **d**, double **sigmaColor,** double **sigmaSpace**, int **borderType** = BORDER\_DEFAULT)

**src** adalah *input gambar;* **dst** adalah *output* gambar; **d** adalah diameter dari pixel pada gambar yang digunakan selama *filtering,* jika nilai itu negatif, maka dihitung dari **sigmaSpace**; **sigmaColor** adalah *filter sigma* dalam dalam ruang warna. Sebuah nilai yang lebih besar dari parameter berarti bahwa warna jauh dalam lingkungan pixel (lihat sigmaSpace) akan diolah bersama - sama, sehingga wilayah yang lebih luas dari *semi-equal color*. **sigmaSpace** adalah *filter sigma* dalam ruang koordinat. Sebuah nilai yang lebih besar dari parameter berarti bahwa piksel jauh akan mempengaruhi satu sama lain selama warna mereka cukup dekat (lihat **sigmaColor**). Ketika **d** > 0, itu menentukan ukuran lingkungan tanpa **sigmaSpace**. Jika tidak, **d** sebanding dengan **sigmaSpace**.

1. *Histogram Equalization*

void **equalizeHist**(InputArray **src**, OutputArray **dst**)

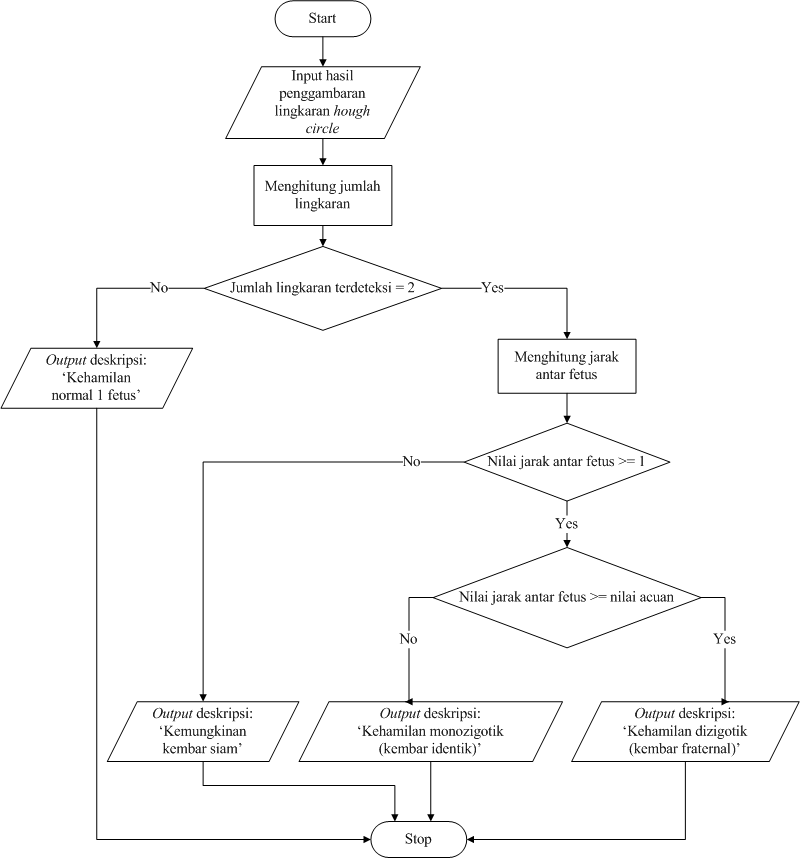
**src** adalah *input* gambar; **dst** adalah *output* gambar.

## Cara penulisan algoritma yang dilengkapi dengan flowchart

*Contoh:*

Perancangan algoritma dalam mengklarifikasi jenis dan kondisi dari kehamilan multifetus adalah:

1. Menghitung jumlah lingkaran yang dibuat dari hasil pendeteksian *hough circle.*
2. Ketika lingkaran yang dideteksi berjumlah 1 maka aplikasi akan menampilkan deskripsi “kehamilan normal 1 fetus”.
3. Ketika lingkaran yang dideteksi berjumlah 2 maka aplikasi akan menghitung jarak antara kedua fetus;
4. Ketika jarak antar kedua fetus bernilai kurang dari 1 (bernilai nol atau negatif) yang berarti posisi dari kedua fetus saling berdempetan, maka aplikasi akan menampilkan deskripsi “kemungkinan kembar siam”.
5. Ketika jarak antar fetus lebih dari sama dengan 1, maka aplikasi akan menghitung perbandingan jarak antara fetus dengan nilai acuan yang telah diperhitungkan dalam penelitian ini (hasil dari total jumlah jari – jari kedua lingkaran);
6. Ketika jarak antara kedua fetus memiliki nilai lebih besar sama dengan dari nilai acuan, maka aplikasi akan menampilkan deskripsi “Kehamilan kembar dizigotik (*fraternal*)”.
7. Ketika jarak antara kedua fetus memiliki nilai lebih kecil dari nilai acuan, maka aplikasi akan menampilkan deskrispi “Kehamilan kembar monozigotik (identik)”.

Gambar 3.11 adalah *flowchart* dari algoritma klasifikasi dari kehamilan multifetus.

Gambar 3.11 mvfgfdglfdgd

Gambar 3.11 *Flowchart* algoritma klasifikasi kehamilan multifetus

# BAB IV

# IMPLEMENTASI DAN PENGUJIAN



## Implementasi Aplikasi *(hanya contoh, sesuaikan dengan sistematika penulisan yang sesuai untuk metodologi penelitian anda)*

*Contoh:*

Implementasi aplikasi dibagi menjadi …. bagian, yaitu tahap ….., tahap ……, dan tahap ….. Berikut penjabaran mengenai tahapan implementasi aplikasi “……judul/nama aplikasi…..”.

### Tahap Pemrosesan Awal

*Contoh:*

Implementasi tahap pemrosesan awal dibagi menjadi … tahap, yaitu …, …, …, …, …, …, dan ….

#### Enhancement (hanya contoh)

Implementasi tahap *enhancement* dibagi menjadi 3 tahapan. Tahapan pertama, yaitu mengubah gambar asli menjadi gambar *grayscale.* Tahapan kedua, yaitu mengaplikasikan *filter. Filter* yang digunakan terdiri dari *mean filter, median filter, gaussian filter,* dan *bilateral filter.* Dengan ukuran *kernel* yang dapat dipilih 3x3, 5x5, 7x7, 9x9, dan 11x11. Untuk mengaplikasikan ketiga tahapan tersebut menggunakan fungsi yang telah disediakan oleh OpenCV yang dijalankan pada piranti lunak *Visual Studio*.

Berikut adalah penjabaran tentang kode yang diimplementasikan dalam aplikasi:

*Contoh cara mencantumkan potongan kode program dengan mempertahankan format penulisan kode pada layar editing dari IDE yang digunakan*.

1. Mengkonversi citra kedalam *grayscale*

Gambar 4.1 Kode mengubah input citra kedalam *grayscale*

//Input image grayscale

Mat img = imread("USG/twins8 (7w).jpg", CV\_LOAD\_IMAGE\_GRAYSCALE);

Pada Gambar 4.1, fungsi dari imread akan mengambil citra *input* dari lokasi sesuai dengan nama citra, dengan penambahan kode **CV\_LOAD\_IMAGE\_GRAYSCALE** maka citra *input* akan selalu dikonversi kedalam 8 *bit image grayscale.*

1. Mengaplikasikan *filtering*

Gambar 4.2 Kode tahap *filtering*

Mat image = img.clone();

///Filtering Image

while (1)

{

switch (filterInt)

{

case 0:

blur(img, out, Size(kernelInt, kernelInt), Point(-1, -1), BORDER\_DEFAULT);

out1 = out.clone();

break;

case 1:

GaussianBlur(img, out, Size(kernelInt, kernelInt), 0, 0);

out1 = out.clone();

break;

case 2:

medianBlur(img, out, kernelInt);

out1 = out.clone();

break;

case 3:

bilateralFilter(img, out, kernelInt, kernelInt \* 2, kernelInt / 1);

out1 = out.clone();

break;

}

*Contoh cara mencantumkan penjelasan terhadap potongan kode program yang digunakan*:

Berikut penjelasan dari kode pada Gambar 4.2:

* 1. Mat image = img.clone();

Kode ini digunakan untuk menduplikasi gambar *image*.

* 1. blur(img, out, Size(kernelInt, kernelInt), Point(-1, -1), BORDER\_DEFAULT);

Kode ini digunakan untuk mengaplikasikan *mean filter.*

* 1. GaussianBlur(img, out, Size(kernelInt, kernelInt), 0, 0);

Kode ini digunakan untuk mengaplikasikan *gaussian blur.*

* 1. medianBlur(img, out, kernelInt);

Kode ini digunakan untuk mengaplikasikan *median blur*

* 1. bilateralFilter(img, out, kernelInt, kernelInt \* 2, kernelInt / 1);

Kode pada gambar 4.3 digunakan untuk mengaplikasikan *bilateral filter.*

switch (kernelInt)

{

…………….

…………….

}

Gambar 4.3 Kode pergantian fitur *kernel* pada *filter*

#### Morphological Image Processing (contoh lain)

Proses selanjutnya adalah tahap memodifikasi informasi dari bentuk struktur obyek pada citra untuk menghilangkan gangguan atau *noise* di sekitar area rahim pada citra serta mendapatkan hasil area atau sebuah bentuk citra yang lebih jelas. Dalam implementasi dari tahap ini, aplikasi dirancang dengan dilatasi dan erosi.

/// Create Dilation Trackbar

createTrackbar("Kernel size:\n 2n +1", "Dilation Demo", &dilation\_size, max\_kernel\_size, Dilation);

Mat element\_d1 = getStructuringElement(MORPH\_ELLIPSE,

Size(2 \* dilation\_size + 1, 2 \* dilation\_size + 1),

Point(dilation\_size, dilation\_size));

/// Apply the dilation operation

dilate(img\_thresh, dilation\_dst, element\_d1);

imshow("Dilation Demo", dilation\_dst);

/// Create Erosion Trackbar

createTrackbar("Kernel size:\n 2n +1", "Erosion Demo",

&erosion\_size, max\_kernel\_size, Erosion);

Mat element\_e = getStructuringElement(MORPH\_ELLIPSE,

Size(2 \* erosion\_size + 1, 2 \* erosion\_size + 1),

Point(erosion\_size, erosion\_size));

/// Apply the erosion operation

erode(dilation\_dst, erosion\_dst, element\_e);

imshow("Erosion Demo", erosion\_dst);

createTrackbar("Kernel size:\n 2n +1", "Dilation Demo 2",

&dilation2\_size, max\_kernel\_size, Dilation2);

Mat element\_d2 = getStructuringElement(MORPH\_ELLIPSE,

Size(2 \* dilation2\_size + 1, 2 \* dilation2\_size + 1),

Point(dilation2\_size, dilation2\_size));

/// Apply the dilation operation

dilate(erosion\_dst, dilation2\_dst, element\_d2);

imshow("Dilation Demo 2", dilation2\_dst);

/// Default start

Dilation(0, 0);

Erosion(0, 0);

Dilation2(0, 0);

Gambar 4.7 Kode tahap *morphological image processing*

Dalam proses ini, citra akan terlebih dahulu dilatasi, kemudian dierosi, dan terakhir dilatasi kembali. Pengguna dapat mengatur seberapa banyak proses dilatasi dan erosi pada citra dengan menggunakan *trackbar* yang diterapkan dalam tahap ini. Dalam pengaplikasian fungsi pada penelitian ini digunakan fungsi yang telah disediakan oleh OpenCV.

Berikut adalah penjabaran dari kode *morphological image processing* pada Gambar 4.7:

1. …..
2. ………..
3. …………….

## Pengujian Aplikasi

*Contoh cara penyajian hasil pengujian:*

Pengujian aplikasi dibagi menjadi ….. bagian besar, yaitu …….., …….., dan ………... Pada penelitian ini pengujian dilakukan dengan menggunakan ………. yang didapatkan dari …………. sebanyak …… dengan jenis format …………

Dalam penelitian ini, untuk memberi gambaran mengenai hasil pengujian fungsional dari aplikasi, dipilih 1 sampel dari 20 sampel yang ada karena tidak memungkinkan untuk melampirkan semua gambar. Untuk pengujian akurasi dalam megklarifikasi dalam mendapatkan area rahim dan mengklarifikasi jenis kehamilan, penelitian ini menampilkan seluruh hasil dari setiap sampel.

Tabel 4.1 Sampel citra USG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Jenis Kehamilan | Nama Sampel | Sumber Sampel | Gambar |
| 1. | Normal 1 Fetus | Sampel 1 | <https://images.radiopaedia.org/images/928694/4a4d1b246b1d29d6f4870192e023c3_big_gallery.jpg> |  |
| 2. | Sampel 2 | <http://www.totalpregnancycare.com/wp-content/uploads/2014/10/OUS.jpg> |  |
| 6. | Kembar Siam | Sampel 6 | <http://momomiracle.com/wp-content/uploads/2014/06/two-300x247.jpg> |  |
| 7. | Sampel 7 | <https://image.slidesharecdn.com/bai13-siumtrongathai-160816153323/95/bai-13-siu-m-trong-a-thai-17-638.jpg?cb=1471361640> |  |
| 11. | Kembar Identik | Sampel 11 | <https://iame.com/online/twin/figure2t.jpg> |  |
| 12. | Sampel 12 | <http://2.bp.blogspot.com/-1qSrUcIGWIw/TjsvowGJMDI/AAAAAAAAEMU/1Xuu_NNt-QA/s320/the%2Bsingle%2Bchorionic%2Bcavity%252C%2Bwhich%2Bcontains%2Btwo%2Bamniotic%2Bcavities.png> |  |
| 16. | Kembar Fraternal | Sampel 16 | <http://ibuhamil.com/attachments/diskusi-umum/14057d1435113886-alhamdulillah-calon-bayiku-kembar-hasilusg8w.jpg> |  |
| 17. | Sampel 17 | <http://www.twin-pregnancy-and-beyond.com/images/6-weeks-pregnant-with-twins-21533536.jpg> |  |

### Pengujian ….

*Contoh:*

Aplikasi tidak mengalami *error* saat melakukan kompilasi, sebagai pengujian fungsional utama. Pengujian fungsional aplikasi dibagi menjadi sembilan bagian, yaitu pengujian fungsional tahap *enhancement,* tahap segmentasi, *morphological image processing,* …. Penjabaran mengenai bagian – bagian dari pengujian fungsional akan dibahas pada subbab – subbab berikut.

#### Pengujian …… Tahap …… (hanya contoh)

Pengujian …. tahap *…..* akan dilakukan pada ……... Fungsi – fungsi yang akan diuji adalah fungsi untuk ….., ……., ……..*.* Pada Gambar 4.19 merupakan hasil dari pengaplikasian fungsi …..di mana fungsi tersebut dapat dijalankan tanpa *error*.

[Tampilkan data hasil pengujian dan analisanya]

#### Pengujian …

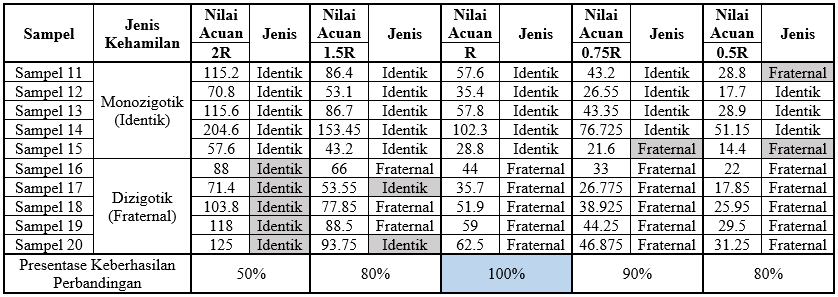
Pengujian …. tahap *…..* akan dilakukan pada ……... Fungsi – fungsi yang akan diuji adalah fungsi untuk ….., ……., ……..*.*

[Tampilkan data hasil pengujian dan analisanya]

#### Pengujian …

Pengujian …. tahap *…..* akan dilakukan pada ……... Fungsi – fungsi yang akan diuji adalah fungsi untuk ….., ……., ……..*.*

[Tampilkan data hasil pengujian dan analisanya]



Tabel 4.4 Hasil percobaan perhitungan nilai acuan dalam klasifikasi kehamilan multifetus

*(contoh penulisan judul tabel jika tabel terdiri dari banyak kolom, sehingga harus ditampilkan secara landscape)*

*Contoh analisa tabel:*

Dari hasil percobaan pada Tabel 4.4 nilai acuan terbaik dengan menggunakan perhitungan dari jumlah total dari nilai jari – jari kedua lingkaran terdeteksi, yaitu dilambangkan dengan R, dengan nilai keberhasilan pengklasifikasian yaitu 100% dengan melakukan perbandingan nilai acuan dengan jarak antara kedua fetus/janin terdeteksi untuk membedakan kehamilan monozigotik (identik) dengan kehamilan dizigotik (fraternal).