BAB III

METODE PENELITIAN

3.1 Alat dan Bahan Penelitian

Pada penelitian ini digunakan alat dan bahan sebagai berikut

a. 33 file photo
b. Perangkat Keras

   Perangkat keras yang digunakan antara lain:

   - Processor Intel I3
   - Harddisk 500 GB
   - RAM 4 GB
   - Kamera Digital

c. Perangkat Lunak

   Perangkat lunak yang digunakan antara lain :

   - Sistem Operasi Windows 7
   - MATLAB 7.8 (R2009a)

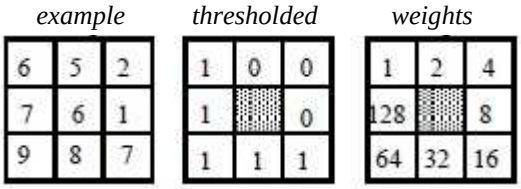
3.2 Pengertian *Local Binary Pattern (LBP)*

Operator *LBP* adalah salah satu deskriptor tekstur terbaik dan telah banyak digunakan dalam berbagai aplikasi. *LBP* telah terbukti sangat diskriminatif dan keuntungan utamanya, yaitu variasi untuk perubahan tingkat abu-abu monoton dan efisiensi komputasi, membuatnya cocok untuk tugas gambar menuntut analisis. Ide untuk menggunakan *LBP* untuk deskripsi kualitas sayur didukung oleh fakta kualitas sayur dapat dilihat sebagai komposisi pola mikro yang dapat dijelaskan oleh sebuah operator.

Local Binary Pattern (LBP) didefinisikan sebagai ukuran tekstur *gray-scale invarian*, berasal dari definisi umum tekstur di daerah sekitar. Operator *LBP* dapat dilihat sebagai pendekatan kesatuan dengan model statistik dan struktur tradisional berbeda dari analisis tekstur.

Secara sederhana, *LBP* adalah sebuah kode biner yang menggambarkan pola tekstur lokal. Hal ini dibangun dengan lingkungan batas dengan nilai abu-abu dari pusatnya (Hadid, & Pietikainen, 2004, pp. 2-3).

Contoh komputasi *LBP* pada 3x3 *pixel*



Pattern = 11110001
LBP = 1 + 16 + 32 + 64 + 128 = 241

Gambar 3.1 Proses perhitungan *pixel LBP*

Setiap *pixel* memiliki nilai hasil *grayscale*, kemudian dilakukan threshold berpusat pada titik tengah. *Pixel* yang memiliki nilai sama atau lebih dibandingkan dengan titik tengah diberi nilai 1 selain itu diberi nilai 0. Kemudian nilai *LBP* didapat dari penjumlahan dua pangkat nilai angka yang bernilai satu.

$$LBP_{P,R}(x_c, x_y) = \sum_{p=U}^{P-1} s(g_p - g_c)2^p \qquad s(x) = \begin{cases} 1, \textit{if } x \geq 0; \\ 0, \textit{otherwise}. \end{cases}$$

Gambar 3.2 Rumus komputasi *LBP*

Dimana :

x_c dan y_c = koordinat pusat *pixel* tetangga,

P = banyaknya *sampling points* atau *pixel* tetangga,

R = *radius* antara *pixel* titik pusat dan *pixel* tetangga,

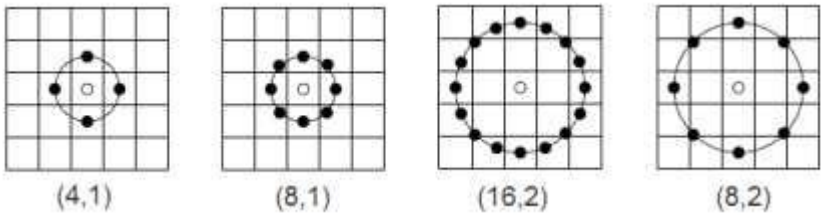
p = *circular sampling points*,

g_p = nilai keabuan dari *p*,

g_c = nilai *pixel* pusat,

s = *sign* (kode biner).

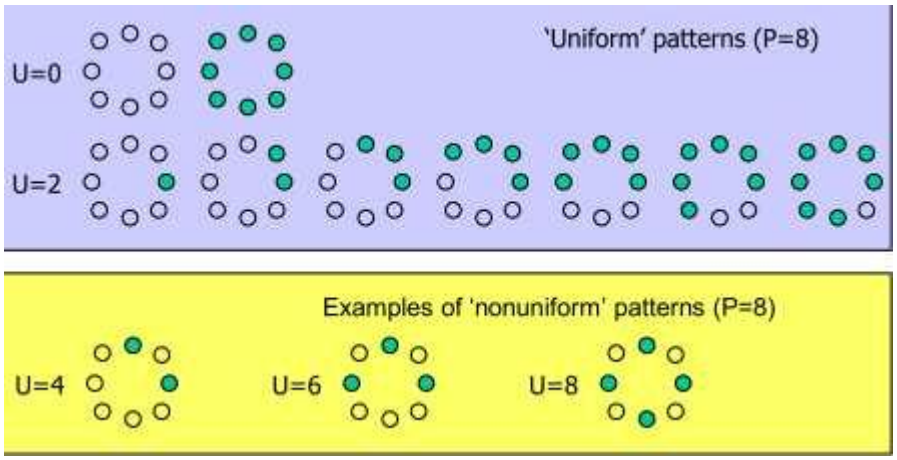
Operator pada *LBP* memiliki label yang ditandai dengan P dan R. P mewakili jumlah *pixel* tetangga yang digunakan dalam komputasi sementara R adalah radius antara *pixel* titik pusat dan *pixel* tetangga (Prasvita, 2016).



Gambar 3.3 Varian *LBP*

Dalam aplikasi analisis tekstur banyak diinginkan untuk memiliki fitur yang *invarian* atau kuat untuk rotasi gambar *input*. Sebagai *LBP*, pola P, R diperoleh dengan sirkuler sampel sekitar *pixel* pusat, rotasi gambar *input* memiliki dua efek: setiap lingkungan lokal diputar ke lokasi *pixel* lainnya, dan dalam masing-masing lingkungan, titik sampling pada lingkaran yang mengelilingi titik pusat diputar ke orientasi yang berbeda.

Perpanjangan pada operator aslinya menggunakan *Uniform Pattern*. Untuk ini, ukuran keseragaman pola yang digunakan: U ("pola") adalah jumlah bitwise transisi dari 0 ke 1 atau sebaliknya ketika pola bit dianggap melingkar. Pola biner lokal disebut seragam jika ukuran keseragaman adalah 2. Misalnya, pola 00000000 (0 transisi), 01110000 (2 transisi) dan 11001111 (2 transisi) adalah seragam sedangkan pola 11001001 (4 transisi) dan 01010011 (6 transisi) tidak. Dalam pemetaan LBP seragam ada yang terpisah *output* label untuk setiap pola seragam dan semua non-seragam pola ditugaskan ke label tunggal. Dengan demikian, jumlah label *output* yang berbeda untuk pemetaan pola-pola.

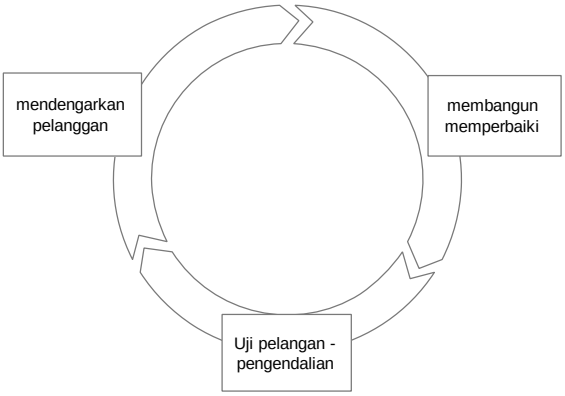


Gambar 3.4 *Uniform Patterns*

3.3 Model Pengembangan Perangkat Lunak

Pengembangan perangkat lunak mengunakan *prototyping model* dalam berbagai situasi dapat menawarkan pendekatan yang terbaik, hal ini didasari oleh pendapat pressman (2002, p.39) yang menyatakan bahwa *prototyping model* merupakan metode yang efektif dalam merancang perangkat lunak.

Pressman (2002, p.40) menjelaskan bahwa *prototyping paradigm* dimulai dengan mengumpulkan kebutuhan. Pengembang bertemu dan pelanggan bertemu dan mendefinisikan object keseluruhan dari perangkat lunak, mengidentifikasi segala kebutuhan yang diketahui dan kemudian melakukan "perancangan kilat". Perancangan kilat berfokus pada penyajian dari aspek-aspek perangkat lunak tersebut yang akan nampak bagi pelanggan atau pemakai (contohnya pendekatan *input* dan format *output*). Perancangan kilat membawa kepada kontruksi sebuah *prototype*. *Prototype* tersebut dievaluasi oleh pelanggan atau pelanggan dan dipakai untuk menyaring kebutuhan pengembangan perangkat lunak.



Gambar 3.5 *Prototype* paradigma

Prototype model juga dapat didefinisikan sebagai proses pengembangan suatu prototype secara cepat untuk digunakan terlebih dahulu dan ditingkatkan terus menerus sampai didapatkan sistem yang utuh. *Prototype model* merupakan proses yang digunakan untuk membantu pengembang perangkat lunak dalam membentuk *prototype* dari perangkat lunak yang harus dibuat.

Proses pada model *prototyping* dapat dijelaskan sebagai berikut :

1. Pengumpulan kebutuhan : *developer* dan klien bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya. Detil kebutuhan mungkin tidak dibicarakan disini, pada awal pengumpulan kebutuhan . Data yang digunakan dalam penelitian ini adalah data dari petani sayur sawi dan kubis.
2. Perancangan : perancangan dilakukan cepat dan rancangan mewakili semua aspek perangkat lunak yang diketahui, dan rancangan ini menjadi dasar pembuatan *prototype*.
3. Evaluasi *prototype* : klien mengevaluasi *prototype* yang dibuat dan digunakan untuk memperjelas kebutuhan perangat lunak.

Perulangan ketiga proses ini terus berlangsung hingga semua kebutuhan terpenuhi. *Prototype-prototype* dibuat untuk memuaskan kebutuhan klien dan

untuk membangun perangkat lunak lebih cepat, namun tidak semua *prototype* bisa dimanfaatkan. Demi kebutuhan klien lebih baik p*rototype* yang dibuat diusahakan dapat dimanfaatkan.

3.4 GUIDE Matlab

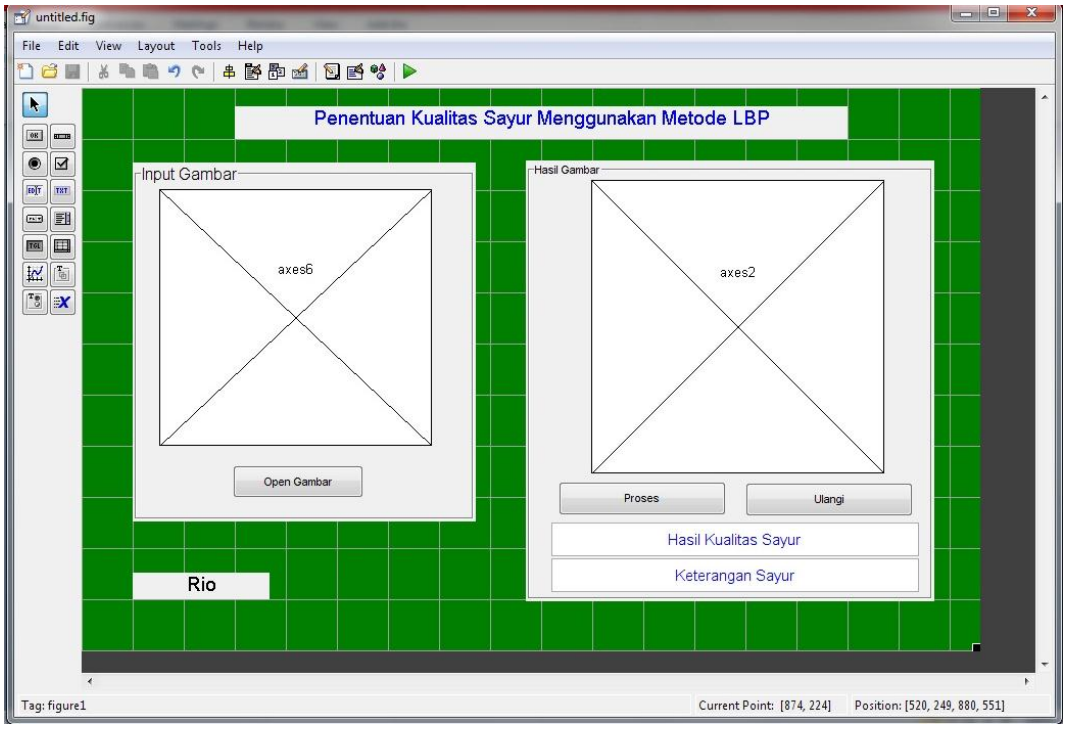
GUIDE atau GUI builder merupakan sebuah graphical user interface (GUI) yang dibangun dengan obyek grafik seperti tombol (button), kotak teks, slider, menu dan lain-lain. Aplikasi yang menggunakan GUI umumnya lebih mudah dipelajari dan digunakan menggunakan GUI umumnya lebih mudah dipelajari dan digunakan karena orang yang menjalankannya tidak perlu mengetahui perintah yang ada dan bagaimana kerjanya.

Kelebihan GUIDE Matlab

1) GUIDE Matlab banyak digunakan dan cocok untuk aplikasi-aplikasi berorientasi sains, sehingga banyak peneliti dan mahasiswa menggunakan GUIDE Matlab untuk menyelesaikan riset atau tugas akhirnya.
2) GUIDE Matlab mempunyai fungsi built-in yang siap digunakan dan pemakai tidak perlu repot membuatnya sendiri.
3) Ukuran file, baik FIG-file maupun M-file, yang dihasilkan relatif kecil.
4) Kemampuan grafisnya cukup andal dan tidak kalah dibandingkan dengan bahasa pemrograman lainnya.

3.5 Rancangan Sistem Program

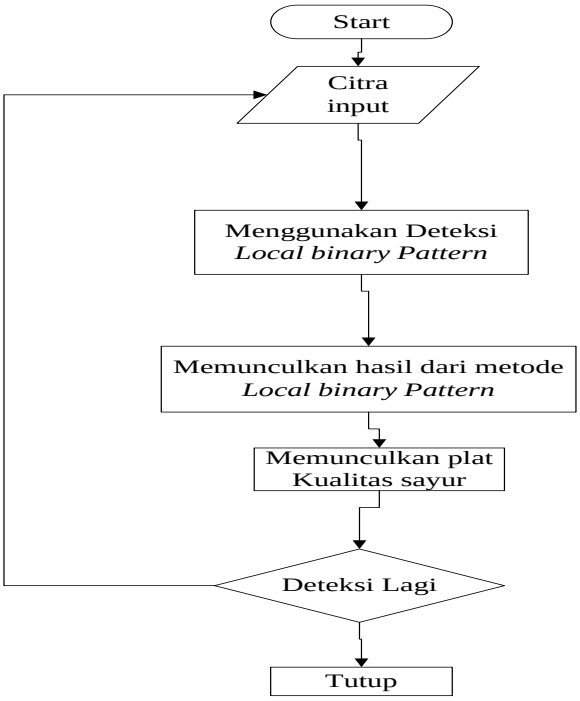
Disini adalah contoh rancangan program yang akan dibangun yang digambarkan dalam bentuk gambar. Ada 2 menu *button* yang tersedia open dan proses. Untuk lebih jelasnya dapat dilihat pada gambar 3.6 dibawah ini:



Gambar 3.6 Tampilan Rancangan Aplikasi

3.5.1 Diagram Blok Sistem Aplikasi

Gambar di bawah ini adalah digram blok tentang program yang akan dibangun. Program berjalan dan *user* memasukan foto dan men*capture* objek sayur yang diinginkan, kemudian foto tersebut di deteksi menggunakan metode *Local Binary Pattern*. Dapat di liat pada gambar 3.7 di bawah ini :



Gambar 3. 7 Diagram Blok Sistem Aplikasi

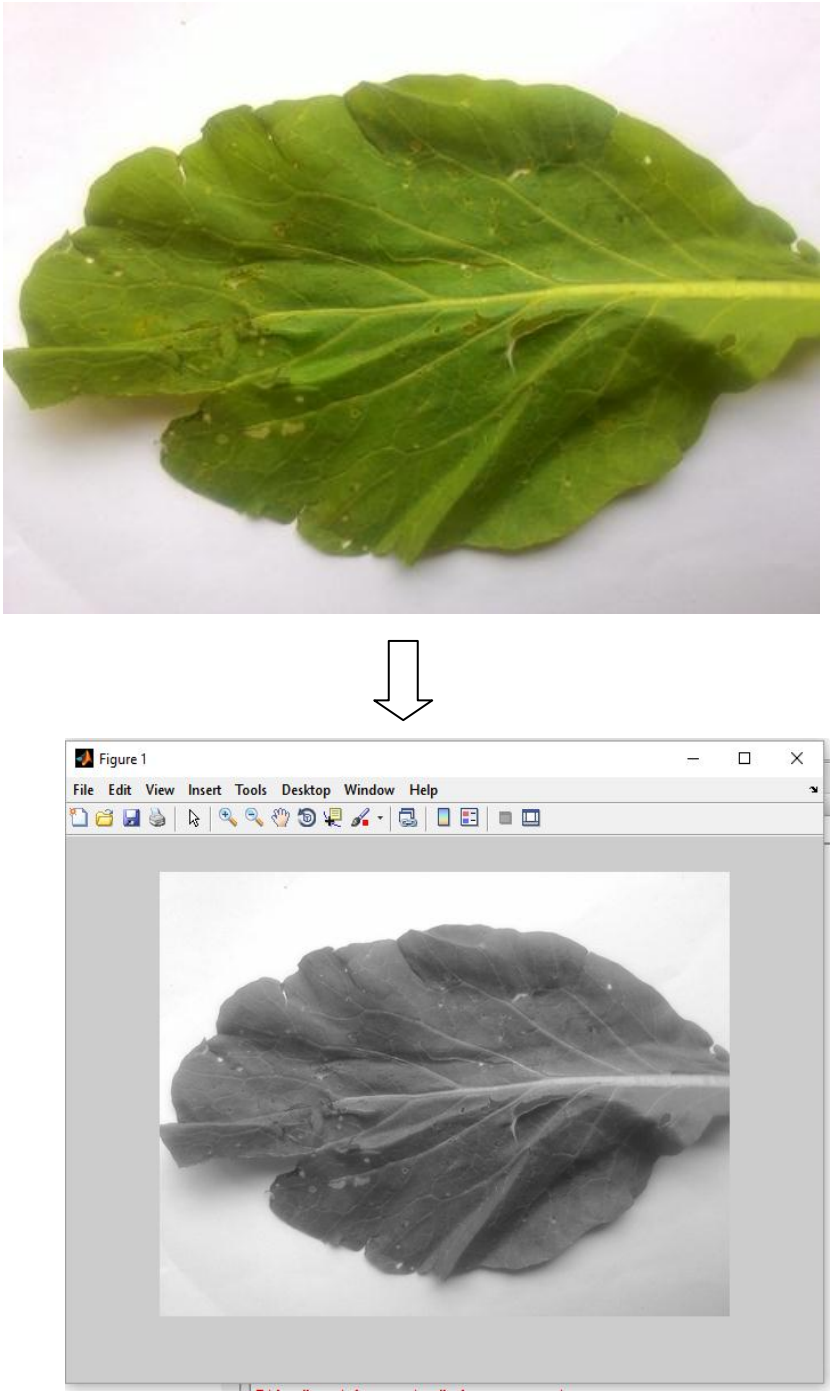
a. Meng*input*kan Citra

Citra *input* berupa citra sayur yang didapatkan dari hasil pengambilan menggunakan kamera, citra ini berupa citra RGB dengan format JPEG.

b. Mengubah Citra RGB Menjadi *Grayscale*

Proses ini termasuk dalam tahapan *preprocessing.* Citra RGB diubah warnanya menjadi abu-abu. Hal ini dimaksudkan untuk mempermudah perhitungan nilai *pixel*. Pada citra RGB setiap *pixel* mempunyai tiga nilai, masing-masing untuk nilai warna merah *(red),* hijau *(green),* dan biru *(blue).* Sedangkan pada citra keabuan tiap *pixel* hanya memiliki satu nilai yang mewakili skala keabuannya. Nilai intensitas citra *grayscale* (keabuan) dihitung dengan nilai intensitas citra RGB.

Citra gambar dapat dilakukan perubahan dari RGB ke *gray* dengan terlebih dahulu mengetahui *pixel region* dari citra buah tersebut. Perintah pada matlab untuk mengetahui *pixel region* tersebut adalah im*pixelregion*. Contoh perubahan citra RGB diubah ke *grayscale* adalah sebagai berikut:

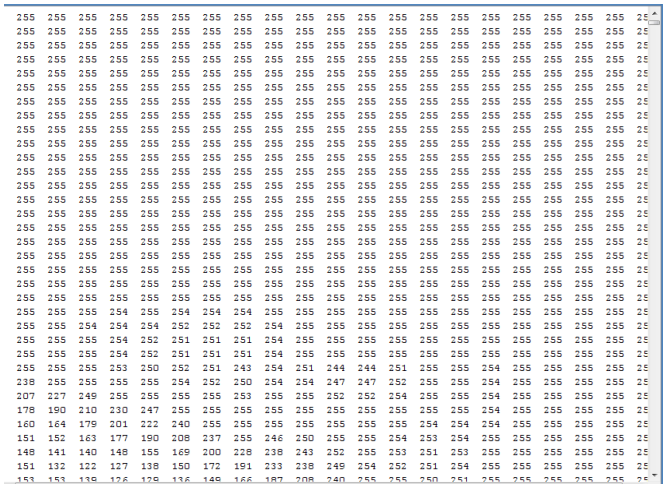


Gambar 3.8 Proses perubahan citra rgb ke *grayscale*

c. Memperoses Citra Dengan Mengunakan Metode *Local Binary Pattern*

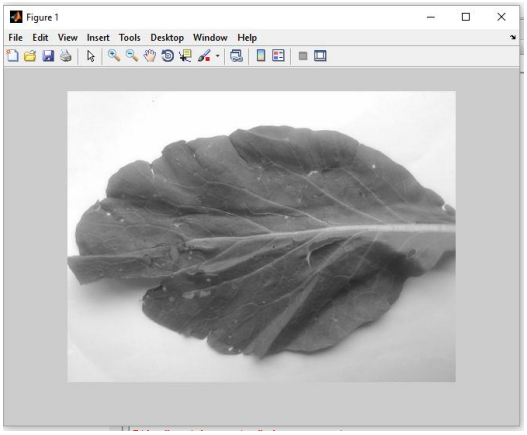
Proses ini adalah fungsi dari program ini. Fungsi untuk mendeteksi pola pada gambar. Program menghitung jumlah pola yang di *input*kan dan nantinya akan memunculkan hasil dari jumlah pola tersebut dengan pola tersebut. Sebelum melakukan pendeteksian yang harus dilakukan terlebih

dahulu melakukan proses mencari matrik citra dengan menggunakan perintah a = imread ('1-1.jpg').



Gambar 3.9 Proses menampilkan matrik sebuah citra

Setelah perhitungan matrik sebuah citra secara otomatis sudah diketahui, maka selanjutnya merubah citra menjadi *grayscale* gray=rgb2gray(a).

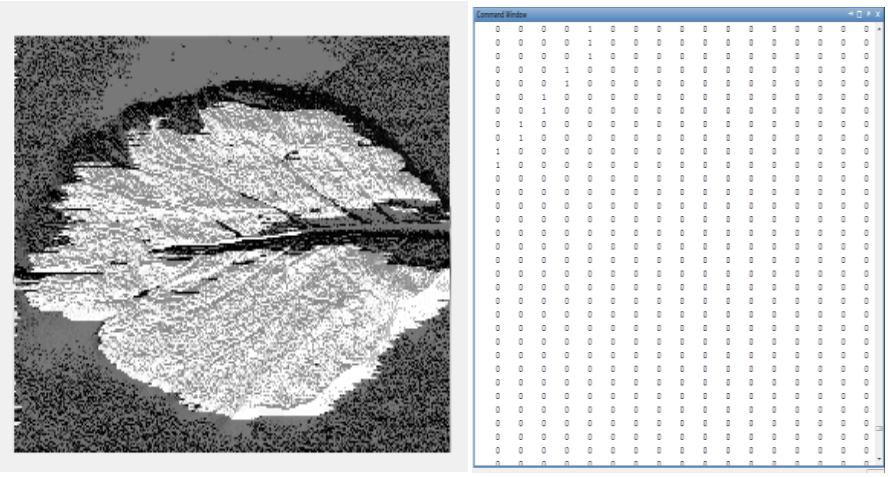


Gambar 3.10 Proses perubahan *grayscale*

Kemudian melakukan pendeteksian dengan menggunakan metode *Local Binary Pattern* dengan perintah

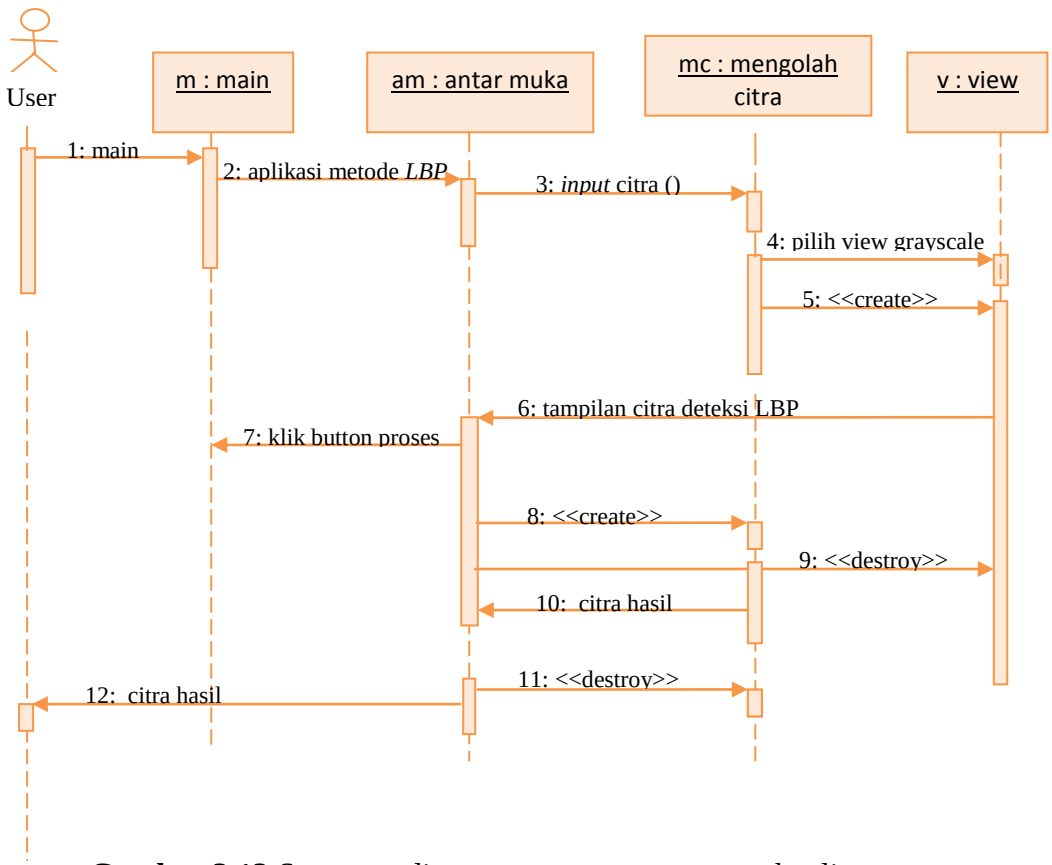
```
 [w h]=size(J);
for i=2:w-1
   for j=2:h-1
      val=J(i,j); scale=2.^[0 1 2;7 -inf 3;6 5 4];
     mat=[J(i-1,j-1) J(i-1,j) J(i-1,j+1);J(i,j-1) J(i,j) J(i,j+1);J(i+1,j-1) J(i+1,j) J(i+1,j+1)];
       mat=mat>=val; fin=mat.*scale; J(i,j)=uint8(sum(sum(fin)));
   end
```

Selanjutnya pengambilan sebuah citra yang akan di cari besarnya nilai citra dengan menggunakan perintah imshow (BW). Untuk citra sayur sawi segar mempunyai angka *pixel* 27000000. Dapat di lihat pada gambar di bawah ini



Gambar 3.11 Proses pengambilan *LBP*

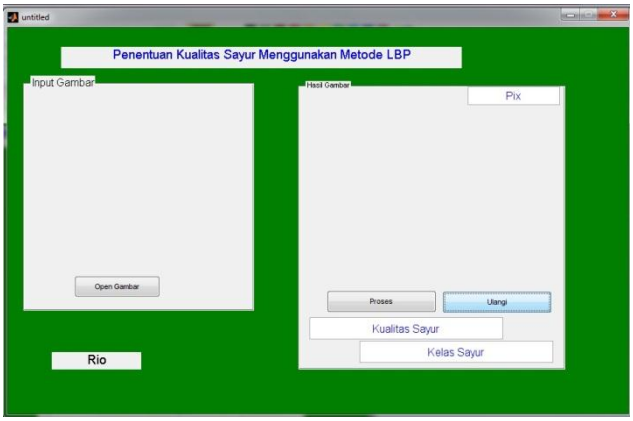
3.5.2 *Sequence* Diagram



Gambar 3.12 *Sequence* diagram program penentuan kualitas sayur

3.5.3 Rancangan Tampilan Antar Muka

Gambar berikut ini adalah merupakan rancangan tampilan antar muka program penentuan kualitas sayur dengan metode *Local Binary Pattern*. Bisa di liat pada gambar 3.13 berikut :



Gambar 3.13 Rancangan antar muka aplikasi penentuan kualitas sayur