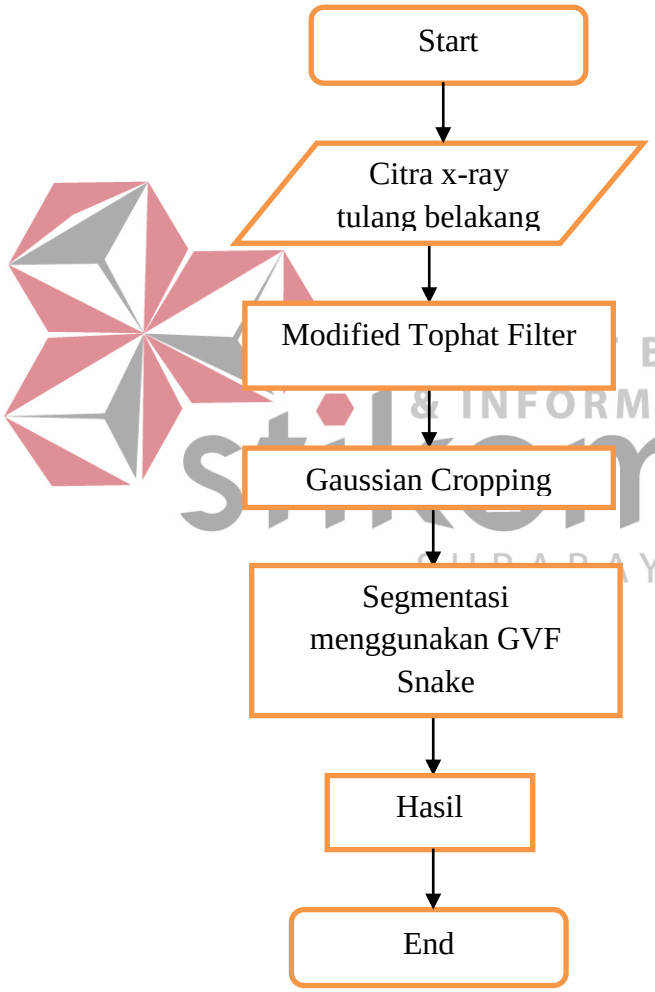
BAB III

METODE PENELITIAN

3.1. Perancangan Sistem dan Blok Diagram Sistem

Model penelitian yang akan dilakukan adalah model penelitian pengembangan. Untuk mempermudah dalam memahami sistem yang akan dibuat dapat dijelaskan melalui blok diagram pada Gambar 3.1.



Gambar 3.1 Blok Diagram

Pada Gambar 3.1 terdapat 2 buah bagian utama, yaitu *pre-processing* dan segmentasi menggunakan GVF Snake.

1. *Pre-processing*

   Pada proses *pre-processing* dilakukan *filtering*. *Filtering* digunakan untuk memisahkan bagian yang dibutuhkan dari *background* sebuah citra sehingga setelah proses ini selesai, didapat sebuah citra baru yang berisi bagian-bagian yang diperlukan saja.

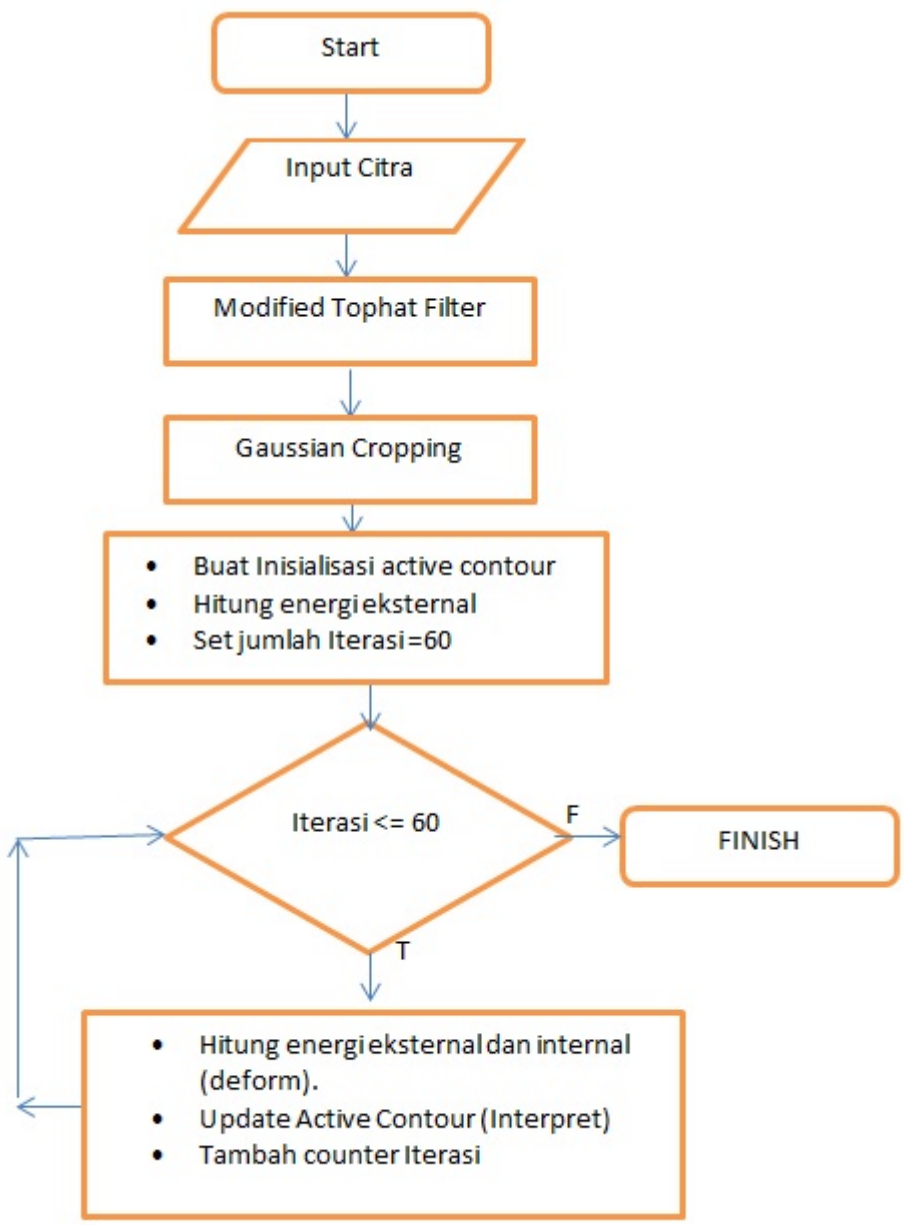
2. Segmentasi menggunakan GVF Snake.

   Setelah proses *filtering*, didapat citra biner yang berbentuk tulang belakang tanpa objek yang lain, barulah proses pembentukan imitasi siluet tulang belakang melalui titik-titik yang telah diinisialisasikan sambil menjaga kelenturan dan bentuk sesuai citra dengan memanfaatkan energi internal dan eksternal.

3.2. Perancangan Perangkat Lunak

Dalam perancangan GVF Snake, *compiler* yang digunakan adalah Microsoft Visual C++ 2008. Untuk *library* yang digunakan pada pengolahan citra yaitu *library* OpenCV v1.1. yang menyediakan fungsi untuk pengolahan citra.

Dalam penulisannya atau dalam pembuatan program, akan meliputi bagian-bagian penting dalam setiap langkah-langkah per bagian sesuai dengan algoritma atau logika sekuensial dari awal sampai output. Berikut adalah *flowchart* program secara global :



Gambar 3.2 Flowchart Program

Deskripsi *flowchart* :

1. Dimulai dari input citra yang akan diproses.
2. Dilanjutkan *pre-processing* dengan menggunakan filter *modified tophat* kemudian menggunakan Gaussian cropping untuk mengekstrak *edge map*, mereduksi *noise* dan bagian-bagian dari citra sinar x yang tidak diperlukan.

3. Selanjutnya proses inisialisasi membuat titik-titik yang nanti akan bergerak menuju tepi citra.
4. Selanjutnya kalkulasikan medan gaya GVF *field* dan simpan pada variabel array dan set iterasi berjumlah 60 kali.
5. Hitung nilai energi eksternal dari citra yang telah terfilter dan energi internal dari *active contour* yang telah diinisialisasi.
6. Lakukan proses perhitungan gaya yang melibatkan energi internal dari GVF Snake dengan energi eksternal dari citra agar titik-titik yang telah diinisialisasi bergerak mendekati tepi citra dan menambah counter iterasi.
7. Ulangi langkah kelima sampai memenuhi jumlah iterasi yang di tentukan.

3.2.1. Perancangan Modified Tophat Filter

Tahap pertama dari proses *pre-processing* adalah proses filtering untuk memperjelas citra tulang belakang dan mengurangi noise di sekitar tulang belakang menggunakan Modified Tophat Filter.

Rumus modified tophat filter adalah :

$$\text{modTH} = \frac{f \circ b - f}{f \bullet b} \times \gamma \quad \text{.....(21)}$$

Dimana : f = citra gambar

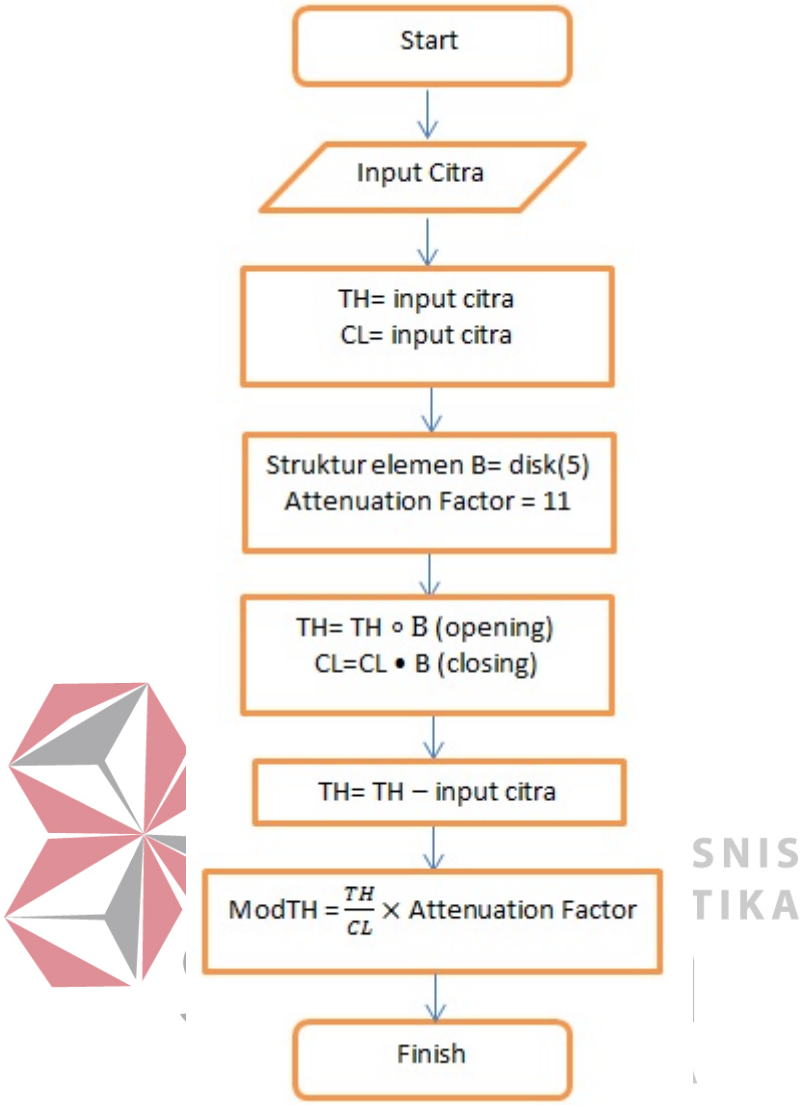
b = struktur elemen

$\circ$ = operator morfologi *closing*

$\bullet$ = operator morfologi *opening*

γ = *attenuation factor*

Alur proses modified tophat bisa dilihat pada flowchart dibawah ini :

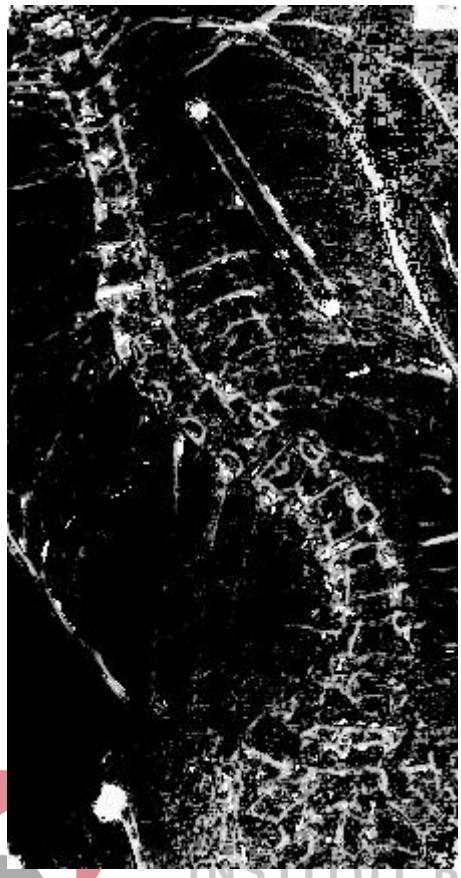


Gambar 3.3 Flowchart Modified Tophat Filter

Jika diprogram menggunakan rumus *modified tophat filter*, maka dapat ditulis *code* program sebagai berikut :

```
int disk = 5;
IplConvKernel *strel =cvCreateStructuringElementEx(2*disk+1,
2*disk+1, disk, disk, CV_SHAPE_ELLIPSE, 0);
norMat(edge, f32Image, 1);
cvMorphologyEx(f32Image,TH,tempMat,strel,CV_MOP_OPEN,1);
cvSub(TH,f32Image,TH);
cvMorphologyEx(f32Image,CL,tempMat,strel,CV_MOP_CLOSE,1);
cvDiv(TH, CL, modTH,11);
norMat(modTH,modTH);
cvCopyImage(modTH,edge);
```

Hasil yang diperoleh ditunjukan pada gambar 3.4 :



Gambar 3.4 Hasil modified tophat filter.

Perancangan modified tophat filter dimulai dengan melakukan operasi *opening* , yaitu proses erosi dilanjutkan dengan dilasi pada citra gambar dengan struktur elemen bertipe *disk* dengan ukuran 5. Hasil dari proses *opening* kemudian dikurangkan dengan citra awal dari citra gambar sehingga menyisakan tepi dari objek saja. Selanjutnya dilakukan proses *closing* menggunakan citra awal, yaitu proses dilasi yang dilanjutkan dengan erosi dengan menggunakan struktur elemen berukuran sama. Proses opening menyebabkan citra menjadi lebih mengembang, sebaliknya proses closing menyebabkan citra menyusut.Kemudian hasil dari *opening* dikurangi citra awal dibagi dengan hasil *closing* untuk menghilangkan sisa *noise* selanjutnya dikalikan dengan *attenuation factor* untuk memperjelas bentuk tepi.

3.2.2. Perancangan Gaussian Cropping

Tahap kedua dari proses pre-precessing adalah dengan menghilangkan untuk mereduksi *noise* dan bagian-bagian dari citra sinar x yang tidak diperlukan khususnya bagian kanan dan kiri dari citra tulang belakang menggunakan Gaussian cropping.

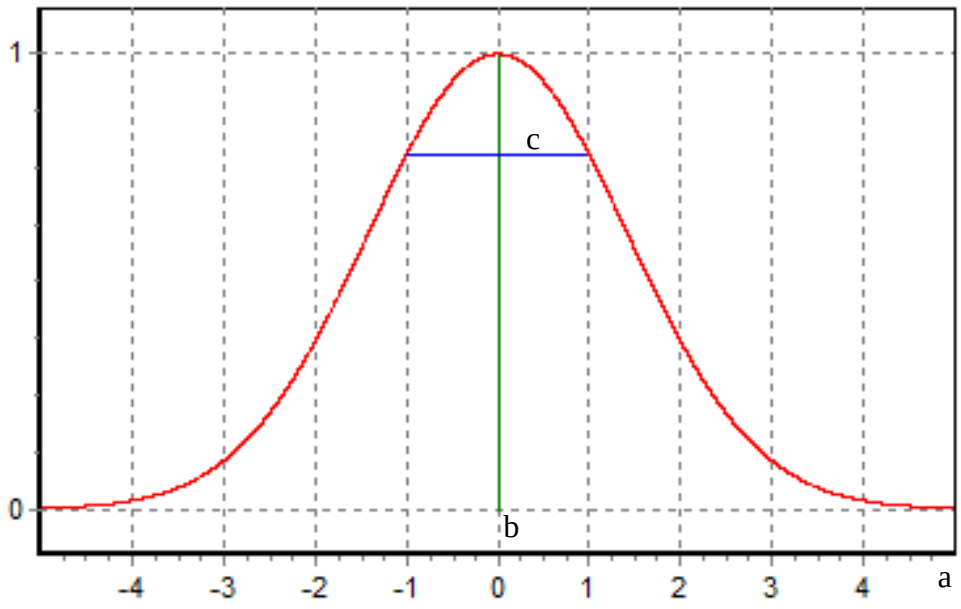
Rumus Gaussian Cropping adalah :

$$Gaussian = \exp^{-\left(\frac{(b-a)^2}{(2c)^2}\right)} \quad \text{.....(22)}$$

Keterangan : a = koordinat piksel pada citra

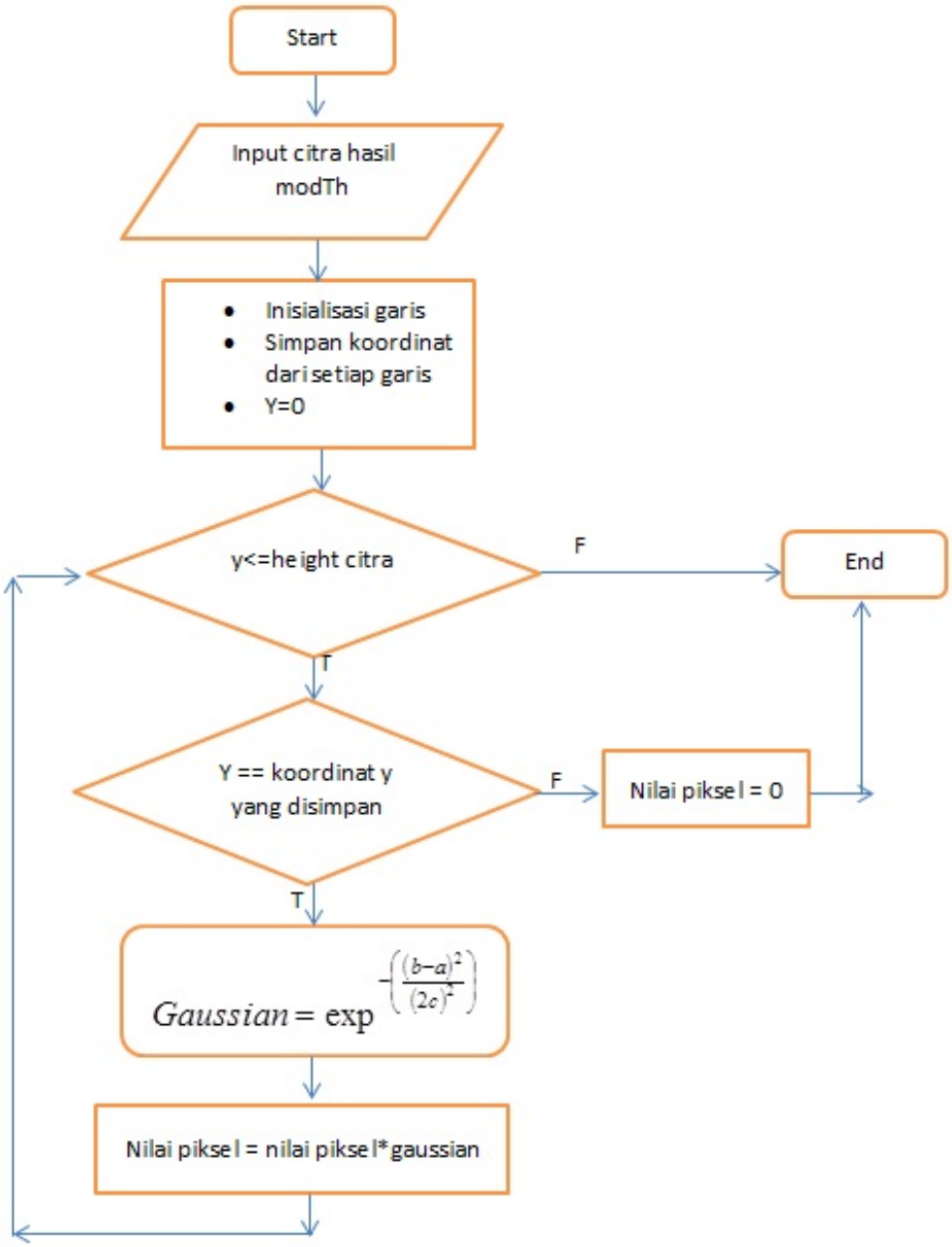
b = koordinat center of peak dari fungsi Gaussian

c = standar deviasi.



Gambar 3.5 Kurva fungsi Gaussian

Alur dari penggunaan gaussian cropping bisa dilihat pada gambar 3.6 :



Gambar 3.6 Flowchart Gaussian cropping

Jika diprogram menggunakan rumus *gaussian cropping*, maka dapat ditulis *code* program sebagai berikut :

```
int counter=0;
int jmlh_baru=jmlh;

      //baris atas
      for (int i=1;i<=koordinat[1][2];i++)
            for (int j=0;j<=edge->width;j++)
                  edge->imageData[edge->widthStep * i + j*
                  edge->nChannels]=0;
```

```
//baris tengah
int index=1;
for (int y=koordinat[1][2];y<=koordinat[jmlh_baru][2];y++)
{
      if (y==koordinat[index][2])
      {
            y=koordinat[index][2];
                  float nilai=0;
                  for (int x=0;x<=edge->width;x++)
                  {
                        double gaussian=0;
                        gaussian=((koordinat[index][1]-
                        x)*(koordinat[index][1]-x))
                        / ((2*10)*(2*10));
                        gaussian=exp(-(gaussian));
                        nilai=(uchar)edge -> imageData
                        [edge->widthStep*y +
                        x*edge->nChannels];
                        edge->imageData[edge->widthStep * y
                        + x* edge->nChannels]=nilai*
                        gaussian;
                  }
                  index=index+1;
      }
      else
      {
            for (int x=0;x<edge->width;x++)
                  edge->imageData[edge->widthStep * y + x*
                  edge->nChannels]=(uchar)edge ->
                  imageData [edge->widthStep*(y-1) +
                  x*edge->nChannels];
      }
}
//baris bawah
for (int y=koordinat[jmlh_baru][2];y<=edge->height-1;y++)
      for (int x=0;x<=edge->width;x++)
            edge->imageData[edge->widthStep * y + x*
            edge->nChannels]=0;
cvDestroyWindow("Gaussian Filter");
cvNamedWindow("Gaussian Filter",1);
cvShowImage("Gaussian Filter",edge);
//kosongkan SLL
while (!InitContour.empty())
{
      InitContour.pop_back();
}
std::cout<<"kosong"<<std::endl;
```

Diawali dengan membuat inisialisasi pada citra tulang belakang menggunakan gerakan mouse dan menyimpan setiap titik yang dilewati oleh mouse. Untuk setiap titik tersebut akan menjadi titik puncak dari fungsi gaussian sehingga semakin kekanan dan kekiri dari koordinat x dari titik puncak akan menghasilkan nilai gaussian yang mendekati nilai 0, nilai gaussian tersebut kemudian dikalikan

dengan nilai pixel awal untuk menghasilkan nilai piksel yang baru. Penentuan nilai variabel c disesuaikan dengan lebar dari tulang belakang.

Hasil yang diperoleh ditunjukan oleh Gambar 3.4.

Gambar 3.7 Hasil Gaussian cropping.

3.2.3. Perancangan GVF Snake

Perancangan GVF Snake meliputi perancangan kalkulasi energi internal (*active contour*) dan energi eksternal citra *edge map*.

1. Perancangan Energi Eksternal

GVF Field adalah medan yang digunakan untuk menunjukan arah gaya. Medan ini digunakan sebagai penentu arah partikel *active contour* untuk bergerak

menuju *edge map* dari citra tulang belakang. Sesuai dengan rumus yang ada pada persamaan 17 :

$$\boldsymbol{\varepsilon = \iint \mu\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) + |\nabla f|^2|\mathbf{v} - \nabla f|^2 dxdy} \quad \text{.....(23)}$$

Sesuai dengan artikel yang ditulis oleh (Cartas,A. dan Ayala, -) berjudul Gradient Vector Flow Snakes, rumus energi eksternal pada persamaan 17 diatas telah di sederhanakan menjadi :

$u_t = \boldsymbol{\mu \nabla^2 u - (u - f_x)\left(f_x^2 + f_y^2\right)}$ .....(24)

Dan

$v_t = \boldsymbol{\mu \nabla^2 v - \left(v - f_y\right)\left(f_x^2 + f_y^2\right)}$ .....(25)

Dimana

u_t = koordinat x untuk setiap iterasi

v_t = koordinat y untuk setiap iterasi

∇ = Laplacian factor

μ = mu

f_x = matrik x

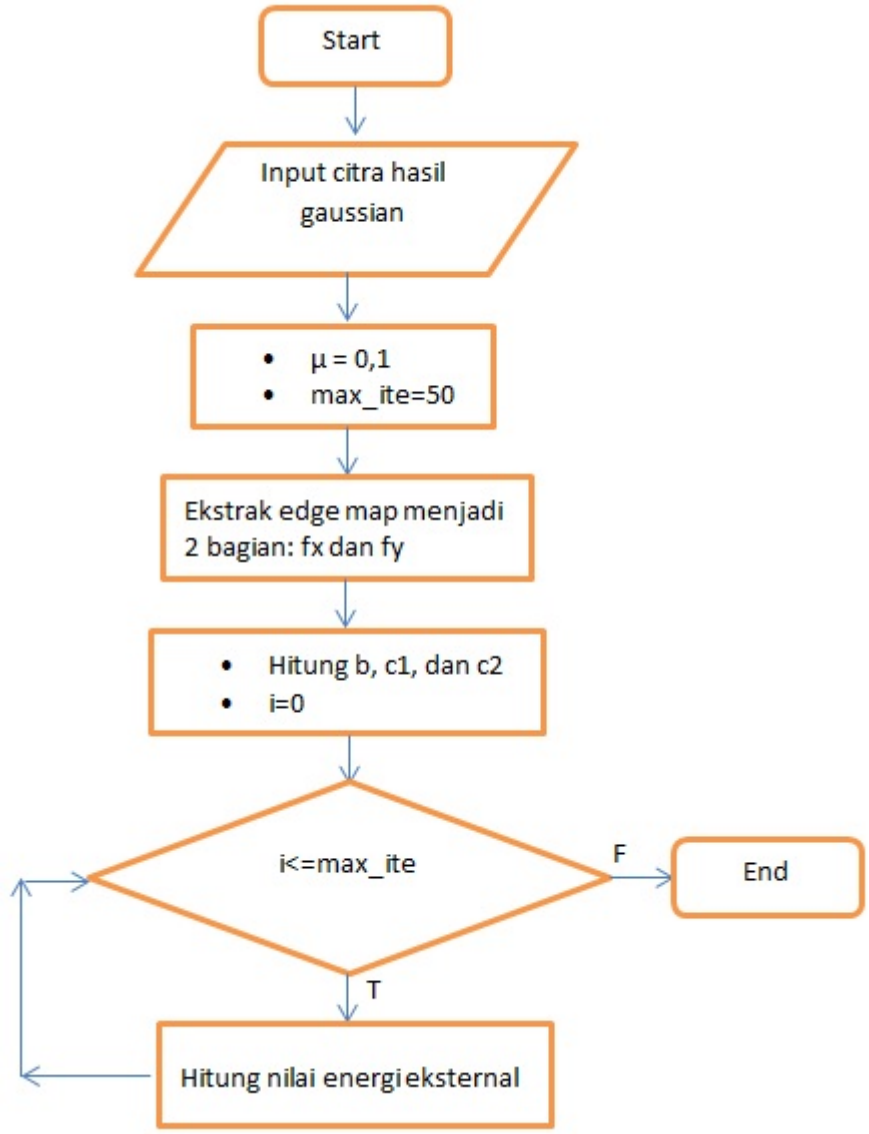
f_y = matrik y

$b = f_x^2 + f_y^2$

$c1 = b \times f_x$

$c2 = b \times f_y$

Alur perhitungan dari energi eksternal dapat dilihat pada flowchart dibawah ini :



Gambar 3.8 Flowchart energi eksternal

Yang jika diimplementasikan dalam bentuk program ditulis seperti di bawah ini:

```
void GVFC(IplImage * edge_map,CvMat * out_u,CvMat * out_v,float
mu,int max_ite )
{
      CvSize size = cvGetSize(edge_map);
      CvMat * fx = cvCreateMat(size.height,size.width,CV_32FC1);
      CvMat * fy = cvCreateMat(size.height,size.width,CV_32FC1);
      CvMat * f =   cvCreateMat(size.height,size.width,CV_32FC1);
      CvMat * b =  cvCreateMat(size.height,size.width,CV_32FC1);
      CvMat * c1 = cvCreateMat(size.height,size.width,CV_32FC1);
      CvMat * c2 = cvCreateMat(size.height,size.width,CV_32FC1);
      CvMat * u =   cvCreateMat(size.height,size.width,CV_32FC1);
      CvMat * v =cvCreateMat(size.height,size.width,CV_32FC1);
      CvMat * lu =   cvCreateMat(size.height,size.width,CV_32FC1);
      CvMat * lv =   cvCreateMat(size.height,size.width,CV_32FC1);
      normMat(edge_map,f);
```

```
CvSepFilter Px;
Px.init_deriv(f->width,f->type,fx->type,1,0,1);
Px.process(f,fx);
cvConvertScale(fx,fx,0.5);

CvSepFilter Py;
Py.init_deriv(f->width,f->type,fy->type,0,1,1);
Py.process(f,fy);
cvConvertScale(fy,fy,0.5);

for(int y=0;y<size.height;y++)
      for(int x=0;x<size.width;x++)
      {
            float fx_t =  CV_MAT_ELEM(*fx,float,y,x);
            float fy_t =  CV_MAT_ELEM(*fy,float,y,x);

            CV_MAT_ELEM(*u,float,y,x) = fx_t;
            CV_MAT_ELEM(*v,float,y,x) = fy_t;

            float bxy = fx_t*fx_t + fy_t*fy_t;

            CV_MAT_ELEM(*b,float,y,x) = bxy;
            CV_MAT_ELEM(*c1,float,y,x) = bxy*fx_t;
            CV_MAT_ELEM(*c2,float,y,x) = bxy*fy_t;
      }

      for(int i=0;i<max_ite;i++)
      {
            cvLaplace(u,lu,1);
            cvLaplace(v,lv,1);

            for(int y=0;y<size.height;y++)
                  for(int x=0;x<size.width;x++)
                  {
                  float tempu = (1.0f - CV_MAT_ELEM
                  (*b,float,y,x) )*
                  CV_MAT_ELEM(*u,float,y,x)
                  +mu*CV_MAT_ELEM(*lu,float,y,x)
                  +CV_MAT_ELEM(*c1,float,y,x);
                        CV_MAT_ELEM(*u,float,y,x) =tempu;

                  float tempv = (1.0f -CV_MAT_ELEM
                  (*b,float,y,x) )*CV_MAT_ELEM(*v,float,y,x)
                  +mu*CV_MAT_ELEM(*lv,float,y,x)
                  +CV_MAT_ELEM(*c2,float,y,x);
                  CV_MAT_ELEM(*v,float,y,x) = tempv;
                  }
      }
      cvCopy(u,out_u);
      cvCopy(v,out_v);
```

Penggunaan cvSepFilter digunakan untuk memisah matrik edge_map dalam 2 bagian, yaitu fx dan fy. Kemudian dilanjutkan dengan menghitung nilai b, c1 dan c2. Untuk menghitung fungsi laplace digunakan cvLaplace yang ada OpenCV.

Kemudian melakukan perhitungan nilai gaussian untuk setiap piksel menggunakan fungsi gaussian untuk menentukan nilai piksel yang baru dengan mengkalikan nilai gaussian yang didapat dengan nilai awal dari piksel.

2. Perancangan Energi Internal

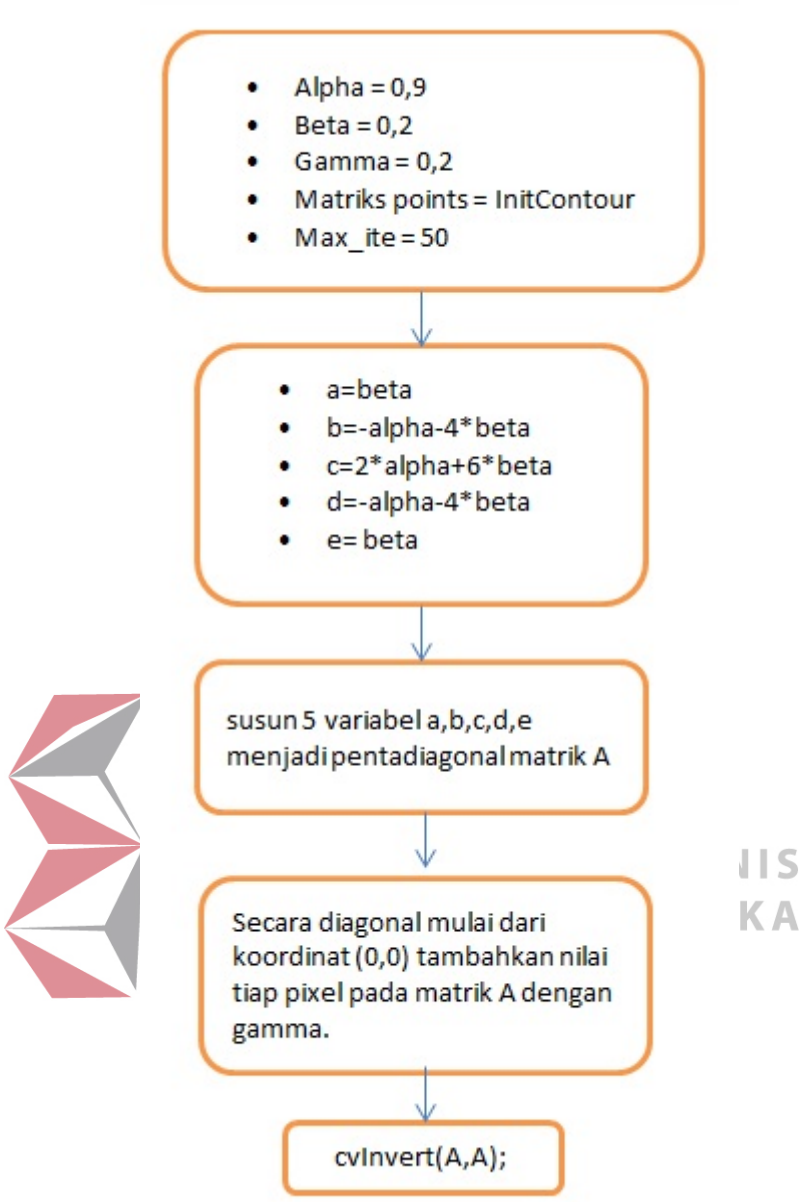
Energi internal didefinisikan pada persamaan 15 sebagai :

$F_{int} = \alpha x''(s) - \beta x''''(s)$.....(26)

Merujuk pada jurnal (Luengo C, 2009) dimana energi internal dijabarkan dalam bentuk matrik berukuran N x N seperti di bawah ini :

$-2\alpha-6\beta$	$\alpha+4\beta$	$-\beta$	0	0	0	⋯	$-\beta$	$\alpha+4\beta$
$\alpha+4\beta$	$-2\alpha-6\beta$	$\alpha+4\beta$	$-\beta$	0	0	⋯	0	$-\beta$
$-\beta$	$\alpha+4\beta$	$-2\alpha-6\beta$	$\alpha+4\beta$	$-\beta$	0	⋯	0	0
0	$-\beta$	$\alpha+4\beta$	$-2\alpha-6\beta$	$\alpha+4\beta$	$-\beta$	⋯	0	0
0	0	$-\beta$	$\alpha+4\beta$	$-2\alpha-6\beta$	$\alpha+4\beta$	⋯	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮
$-\beta$	0	0	0	0	0	⋯	$-2\alpha-6\beta$	$\alpha+4\beta$
$\alpha+4\beta$	$-\beta$	0	0	0	0	⋯	$\alpha+4\beta$	$-2\alpha-6\beta$

Dari matriks diatas dapat diketahui bahwa hanya 5 diagonal saja yang mempunyai nilai, sisanya bernilai 0. Matrik ini yang kemudian disebut sebagai *cyclic pentadiagonal banded matrix.* Implementasi energi internal dapat dilihat pada flowchart dibawah ini.



Gambar 3.9 Flowchart energi internal

Kalkulasi energi internal memakai rumus pada persamaan 15 dimana pada program ditulis sebagai berikut :

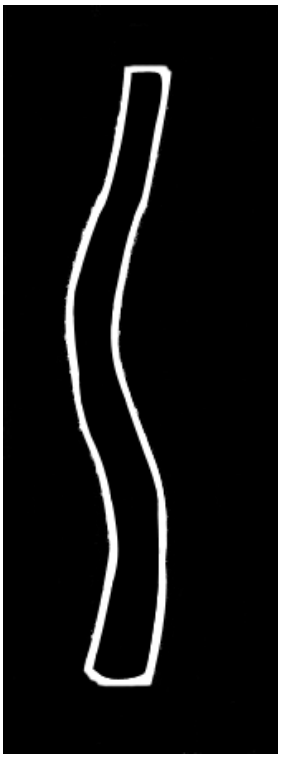
```
void snakeDeform( std::vector<CvPoint>& points,double alpha,double
beta,double gamma,double kappa,CvMat * u,CvMat * v,int max_ite)
{
	assert(points.size()>5);

	CvSize size = cvGetSize(u);
	float a = beta;
	float b = -alpha-4*beta;
	float c = 2*alpha+6*beta;
	float d = -alpha - 4*beta;
```

```
float e = beta;

int y,x,i,k;
int len = points.size();

CvMat * A = cvCreateMat(len,len,CV_32FC1);
cvZero(A);
for( y = 0;y<len;y++)
{
      int index = (2*len-2 + y )%len;
      CV_MAT_ELEM(*A,float,y,index) = a;

      index = (index+1)%len;
      CV_MAT_ELEM(*A,float,y,index) = b;

      index = (index+1)%len;
      CV_MAT_ELEM(*A,float,y,index) = c;

      index = (index+1)%len;
      CV_MAT_ELEM(*A,float,y,index) = d;

      index = (index+1)%len;
      CV_MAT_ELEM(*A,float,y,index) = e;
}

//A = inv(A+gamma*I);
for( y=0;y<len;y++)
      for( x=0;x<len;x++)
      {
            if(y==x)
                  CV_MAT_ELEM(*A,float,y,x) =
                  CV_MAT_ELEM(*A,float,y,x);
      }
cvInvert(A,A);
```


Implementasi energi internal dilakukan dengan membuat matrik pentadiagonal dari komponen alpha, beta dan gamma sesuai dengan format matrik pentadiagonal. Kemudian dilakukan invers pada matrik A.

3. Pengujian Parameter

Pengujian parameter dilakukan untuk mengetahui efek dari masing-masing parameter yang ada, dan menentukan parameter yang paling cocok untuk digunakan pada sampel. Pengujian ini dilakukan dengan memakai citra berbentuk S-shaped.



Gambar 3.10 Template S-shaped

Ada tiga parameter yang akan diuji :

3.1 Parameter Alpha

3.2 Parameter Beta

3.3 Parameter Gamma

Parameter-parameter diatas mempunyai nilai antara 0-1.

3.1 Parameter Alpha

Pengujian pada parameter yang mengatur elastisitas partikel Snake dilakukan dengan mengatur parameter alpha dengan nilai 0 sampai dengan 1 sedangkan parameter beta dan gamma secara berturut-turut bernilai 0,2 dan 0,2.

3.2 Parameter Beta

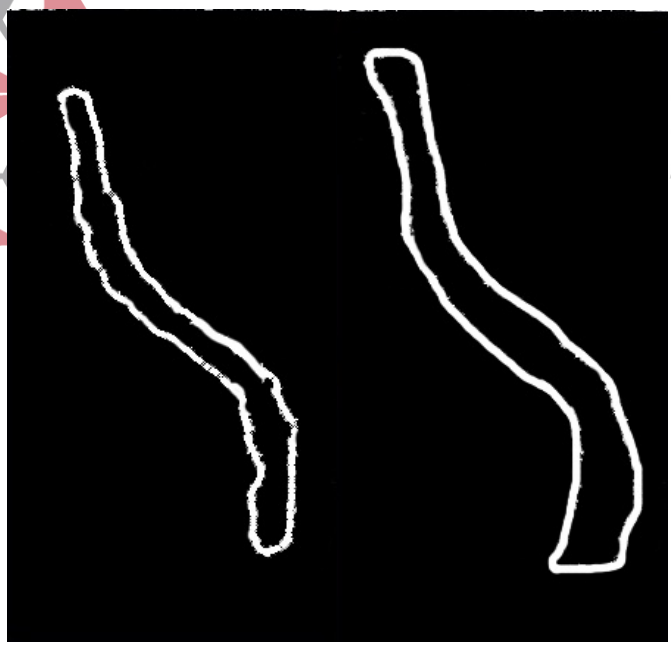
Pengujian pada parameter yang mengatur kekakuan partikel Snake dilakukan dengan mengatur parameter beta dengan nilai 0 sampai dengan

1 sedangkan parameter alpha dan gamma secara berturut-turut bernilai 0,9 dan 0,2.

3.3 Parameter Gamma

Pengujian pada parameter yang mengatur kelekatan antara partikel snake dengan citra dilakukan dengan mengatur parameter gamma dengan nilai 0 sampai dengan 1 sedangkan parameter alpha dan beta secara berturut- turut bernilai 0,9 dan 0,2.

Pengujian akurasi dilakukan menggunakan citra S-shaped dengan metode ROC dimana nilai piksel dari hasil GVF Snake dibandingkan dengan nilai piksel pada template yang dibuat sebagai pembanding.



(a) (b)

Gambar 3.11 Citra sampel ke 8 (a) Hasil GVF Snake (b) Citra pembanding.

Hasil pengujian ketiga parameter dalam persen menggunakan ROC ditunjukan pada tabel 3.1

Tabel 3.1 Hasil pengujian akurasi pada parameter alpha, beta dan gamma.

Parameter	Nilai Parameter										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
Alpha	73,39	81,67	80,9	80,49	82,7	80,79	81,34	82,21	81,46	83,16	83,1
Beta	81,78	80,79	83,5	81,61	82,92	82,54	82,66	82,89	80,47	82,67	83,46
Gamma	0	82,65	83,39	80,79	79,03	80,07	78,08	78,41	79,37	78,64	80,9

Dari Tabel 3.1 dapat diketahui hasil pengujian akurasi menggunakan metode ROC yang menunjukan nilai persentase akurasi pada parameter alpha, beta, dan gamma dengan nilai mulai dari 0 sampai 1 dengan kenaikan 0,1. Dari Tabel 3.1 dapat diketahui nilai untuk setiap parameter yang dapat memberikan akurasi terbaik untuk segmentasi tulang belakang berturut-turut adalah alpha 0.9, beta 0.2, dan gamma 0.2.