LAMPIRAN

LAMPIRAN

1. *Listing* Program (*source code*)

Berikut ini adalah listing dari program (*source code*) dalam bahasa pemrograman Matlab (*Matrix Laboratory*) yang digunakan dalam penelitian. Bahasa pemrograman Matlab yang digunakan adalah Matlab versi 7.6.0.324 (R2008a). Beberapa fungsi merupakan hasil modifikasi dari [14].

a. File *compressAG.m*

```
%% function compressAG
%% input :
%   IszName     : string contain input image's file name
%   OszName     : string contain output compressed 
image's file name
%   dSize       : size of domain blok
%   UkPop       : size of population
%   MaxG        : size of maximum number generation
%   errLimit    : limit for RMS
%   pCross      : probability of crossover
%   pMut        : probability of mutation
 %% %%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%   Start Function
function compressAG(IszName, OszName, dSize, UkPop, MaxG, 
errLimit, pCross, pMut)
    %% size of domain blok
    sdBlok = dSize;

    %% read input image
    I = imread(IszName);
    [m n] = size(I);

    m = m - (sdBlok -1);

    %% creating range blok
    fprintf('\nCreate Range.....');
    [Range rSize] = createRange(I, sdBlok/2);

    %% init Transformation matrix
    MTransform = zeros(rSize, 5);

    %% Do compress using Genetic Algoritm
    fprintf('\n\nCompresing .....                  ');

    %counter = 0;
    %time = rSize * dSize * 8;

    %% initiate fitness vector
    Fitness = zeros(1, UkPop);

    %% init graph
    Bgraf = 100/errLimit;
    hfig = figure;
    hold on;
    title('Kompresi Fraktal dengan AG');
    set(hfig, 'position', [50, 50, 600, 400]);
    set(hfig, 'DoubleBuffer', 'on');
    axis([1 MaxG 0 Bgraf]);
    hbestplot = plot(1:MaxG, zeros(1,MaxG),'.-');
    htext1 = text(0.6*MaxG, 0.30*Bgraf, sprintf('Best
Fitness: %7.4f',0.0));
    htext2 = text(0.6*MaxG, 0.25*Bgraf, sprintf('X :
%d',0));
    htext3 = text(0.6*MaxG, 0.20*Bgraf, sprintf('Y :
%d',0));
    htext4 = text(0.6*MaxG, 0.15*Bgraf, sprintf('Range :
%d',0));
    htext5 = text(0.6*MaxG, 0.10*Bgraf, sprintf('Time :
%5.2f s',0.0));

    tic;
    %% GA goes here
    for i = 1:rSize
        %% do GA for each range blok

        rBlok = GetImage(I, Range(i,:));
        X = initPopulation(UkPop, 3, m);

        %% clear graph
        set(hbestplot, 'YData', zeros(1,MaxG));
        for j = 1:MaxG
            A = [X(1,1) X(1,2) sdBlok];
            dTrans = ApplyTransform(I, A, X(1,3));
            [Fitness(1) sBit oBit] = fitness(rBlok, 
dTrans);
            MaxF = Fitness(1);
            MinF = Fitness(1);
            bestIndex = 1;
            bestX = X(1, :);

            for k = 2:UkPop
               A = [X(k, 1:2) sdBlok];
               dTrans = ApplyTransform(I, A, X(k,3));
               [Fitness(k) s o] = fitness(rBlok, dTrans);

               if Fitness(k) > MaxF
                  MaxF = Fitness(k);
                  bestIndex = k;
                  bestX = X(k, :);
                  sBit = s;
                  oBit = o;
               end

               if Fitness(k) < MinF
                  MinF = Fitness(k);
               end
            end

            %% Fill Graph
            plotvector = get(hbestplot,'YData');
            plotvector(j) = MaxF;
            set(hbestplot, 'YData', plotvector);
            set(htext1, 'string',sprintf('Best Fitnes: 
%7.4f', MaxF));
            set(htext2, 'string',sprintf('X: %d', 
bestX(1)));
            set(htext3, 'string',sprintf('Y: %d', 
bestX(2)));
            set(htext4, 'string',sprintf('Range: %d', 
i));
            set(htext5, 'string',sprintf('Time: %5.2f s', 
toc));
            drawnow

            if MaxF >= 100/errLimit
```

```
                break;
            end

            TemPopulasi = X;

            if mod(UkPop, 2) == 0
                starX = 3;
                TemPopulasi(1,:) = X(bestIndex,:);
                TemPopulasi(2,:) = X(bestIndex,:);
            else
                starX = 2;
                TemPopulasi(1,:) = X(bestIndex,:);
            end

            FitnessLin = LinearFitnessRank(UkPop,
Fitness, MaxF, MinF);

            %% generate new population

            for k = starX:2:UkPop
               IP1 = seleksi(UkPop, FitnessLin);
               IP2 = seleksi(UkPop, FitnessLin);

               if rand < pCross
                  offSpring = crossOver(X(IP1,:),
X(IP2,:));
                  TemPopulasi(k, :) = offSpring(1,:);
                  TemPopulasi(k+1, :) = offSpring(2,:);
               else
                  TemPopulasi(k, :) = X(IP1,:);
                  TemPopulasi(k+1, :) = X(IP2,:);
               end
            end

            for k = starX:UkPop
               TemPopulasi(k,:) =
mutation(TemPopulasi(k,:), pMut, m);
            end

            X = TemPopulasi;
        end

        MTransform(i,1:3) = bestX;
        MTransform(i,4:5) = [sBit oBit];
```

```
    end

    fprintf('%5.2f s',toc);
    fprintf('\nDone\n');

    PID = fopen(OszName, 'wb');

    fwrite(PID, rSize, 'uint16');
    fwrite(PID, 5, 'uint8');
    fwrite(PID, sdBlok, 'uint8');
    fwrite(PID, n, 'uint16');

    if n > 256
        for i = 1:rSize
            fwrite(PID, MTransform(i,1:3),'uint16');
            fwrite(PID, MTransform(i,4:5),'single');
        end
    else
        for i = 1:rSize
            fwrite(PID, MTransform(i,1:3),'uint8');
            fwrite(PID, MTransform(i,4:5),'single');
        end
    end

    fclose(PID);
```

b. File decompress.m

```
function decompress(IszName, OszName, iterate, OriImage)

    PID = fopen(IszName, 'rb');
    nBaris = fread(PID,1,'uint16');
    nKolom = fread(PID,1,'uint8');
    sdBlok = fread(PID,1,'uint8');
    imSize = fread(PID,1,'uint16');
    Range = zeros(nBaris, nKolom);

    if imSize > 256
        for i = 1:nBaris
            Range(i, 1:3) = fread(PID, [1 3], 'uint16');
            Range(i, 4:5) = fread(PID, [1 2], 'single');
        end
    else
        for i = 1:nBaris
```

```
            Range(i, 1:3) = fread(PID, [1 3], 'uint8');
            Range(i, 4:5) = fread(PID, [1 2], 'single');
        end
    end

    fclose(PID);

    I = zeros(imSize, imSize);
    [RangeBlok rSize] = createRange(I, sdBlok/2);

    fprintf('\n\nDecompresing .....                  ');
    counter = 0;
    time = iterate * nBaris;

    for i = 1:iterate
       for j = 1:nBaris
            S = ApplyTransform(I,[Range(j,1:2) sdBlok],
Range(j,3));
            S = ApplyGrayScale(S, Range(j,4),Range(j,5));
            I = ApplySubImage(I,RangeBlok(j,:),S);

            counter = counter + 1;
            persen = (counter/time)*100;

            if persen < 10
                fprintf('\b\b\b\b\b%3.2f%%',persen);
            elseif persen <= 10 && persen < 100
                fprintf('\b\b\b\b\b%3.2f%%',persen);
            else
                fprintf('\b\b\b\b\b\b%3.2f%%',persen);
            end
       end
    end

    I = uint8(I);
    O = imread(OriImage);
    nRMS = rms(I, O);
    PSNR = 20*log10(255/nRMS);
    fprintf('\nrms = %.3f', nRMS);
    fprintf('\nPSNR = %.3f db', PSNR);
    fprintf('\nDone\n');

    imwrite(I, OszName);
    figure;
    imshow(I);
```

c. File *createRange.m*

```
function [S n] = createRange(I, sz)

    [m, n] = size(I);
    count = 1;

    for i = 1:sz:m -sz +1
       for j = 1:sz:n -sz+1
          S(count,1) = i;
          S(count,2) = j;
          S(count,3) = sz;
          count = count + 1;
       end
    end

    n = count - 1;
```

d. File *crossOver.m*

```
function offSpring = crossOver(P1, P2)

    temp = rand;

    offSpring(1, 1) = round(temp*P1(1,1) + (1 -
temp)*P2(1,1));
    offSpring(1, 2) = round(temp*P1(1,2) + (1 -
temp)*P2(1,2));

    offSpring(2, 1) = round((1 - temp)*P1(1,1) +
temp*P2(1,1));
    offSpring(2, 2) = round((1 - temp)*P1(1,2) +
temp*P2(1,2));

    temp = rand;

    if temp < 0.5
       offSpring(1,3) = P1(1,3);
    else
       offSpring(1,3) = P2(1,3);
    end

    temp = rand;
    if temp < 0.5
       offSpring(2,3) = P2(1,3);
    else
       offSpring(2,3) = P1(1,3);
    end
```

e. File *distanceImage.m*

```
function [arc sBit oBit] = distanceImage(rBlok, dBlok)
    dBlok = double(dBlok);
    rBlok = double(rBlok);
    [x y] = size(rBlok);

    if ([x y] == size(dBlok))
        n = x*y;
        DR1 = sum(sum(dBlok.*rBlok,1),2);
        d1 = sum(sum(dBlok,1),2);
        r1 = sum(sum(rBlok,1),2);
        d12 = sum(sum(dBlok.^2,1),2);
        r12 = sum(sum(rBlok.^2,1),2);
        DR2 = d1 * r1;
        DR3 = d12;
        DR4 = d1^2;

        div = n*DR3 - DR4;

        if div == 0
           sBit = 0;
        else
           sBit = (n*DR1 - DR2)/div;
        end

        oBit = (r1 - sBit*d1)/n;
        %arc = (r12 + sBit*(sBit*d12-2*DR1+2*oBit*d1) +
oBit*(oBit*n-2*r1))/n;
        arc = sum(sum((sBit*dBlok + oBit -
rBlok).^2,1),2)/n;
        arc = sqrt(arc);
    end
```

f. File *fitness.m*

```
function [fit s o] = fitness(rBlok, dBlok)
```

```
    [arc s o] = distanceImage(rBlok, dBlok);
    fit = 100/(arc + 1);
```

g. File *flip.m*

```
function S = flip(I, type)
    [m n] = size(I);
    S = zeros(m,n);

    switch type
        case 1  %refeleksi terhadap sumbu-x
            for i=1:m
                S(i,:) = I(m + 1 -i,:);
            end
        case 2  %refeleksi terhadap sumbu-y
            for i=1:n
                S(:,i) = I(:, n + 1 -i);
            end
        case 3  %refeleksi terhadap y = x
            for i=1:n
                for j=1:m
                    S(j,i) = I(n+1-i, m + 1 -j);
                end
            end
        case 4  %refeleksi terhadap y = -x
            for i=1:n
                for j=i:m
                    S(j ,i) = I(i, j);
                    S(i ,j) = I(j, i);
                end
            end
    end
```

h. File *GetImage.m*

```
function S = GetImage(I, A)

    x = A(1,1);
    y = A(1,2);
    zx = A(1,3);
    S = zeros(zx,zx);
    S = I(x:x + zx -1,y:y + zx -1);
```

i. File *initPopulation.m*

```
function P = initPopulation(UkPop, UkGen, rSize)

    P = zeros(UkPop, UkGen);

    for i = 1:UkPop
        temp = rand;

        if temp <= 0.5
            P(i, 1) = mod(round(rand * 100), rSize) + 1;
            P(i, 2) = mod(round(rand * 100), rSize) + 1;
        else
            P(i, 1) = mod(round(rand * 1000), rSize) + 1;
            P(i, 2) = mod(round(rand * 1000), rSize) + 1;
        end

        P(i, 3) = mod(round(rand * 10), 8);
end
```

j. File *LinearFitnessRank.m*

```
function LFR = LinearFitnessRank(UkPop, Fitness, MaxF,
MinF)

    [SF, IndF] = sort(Fitness);
    LFR = zeros(1, UkPop);

    for i = 1:UkPop
       LFR(IndF(UkPop - i + 1)) = MaxF - (MaxF -
MinF)*((i - 1)/(UkPop - 1));
    end
```

k. File *mutation.m*

```
function P = mutation(P1, PMutation, rSize)

    P = P1;
    temp = rand;

    if temp < PMutation
```

```
       temp = mod(round(rand*10),3)+1;

       switch temp
           case 1,
               P(1,1) = mod(round(rand * 100), rSize) +
1;
           case 2
               P(1,2) = mod(round(rand * 100), rSize) +
1;
           case 3
               P(1,3) = mod(round(rand * 10), 8);
       end
    end
```

l. File *rms.m*

```
function arc = rms(I, O)
    I = double(I);
    O = double(O);
    [m n] = size(I);
    temp = sum(sum((I-O).^2,1),2);
    temp = temp /(n*m);
    arc = sqrt(temp);
```

m. File *seleksi.m*

```
function Pindex = seleksi(UkPop, LinearFitness)

    total = sum(LinearFitness);
    CumFit = 0;
    temp = rand;
    i = 1;

    while i <= UkPop
       CumFit = CumFit + LinearFitness(i);

       if CumFit/total > temp
          Pindex = i;
          break;
       end
       i = i + 1;
    end
```

n. File *ApplyGrayScale.m*

```
function S = ApplyGrayScale(I, s, o)
    I = double(I);
    S = I*s + o;
    %S = uint8(S);
```

o. File *ApplySubImage.m*

```
function Z = ApplySubImage(I, A, S)
    x = A(1,1);
    y = A(1,2);
    sz = A(1,3);
    I(x:x+sz - 1, y:y+sz - 1) = S;
    Z = I;
```

p. File *ApplyTransform.m*

```
function S = ApplyTransform(I, A, tipe)
    x = A(1,1);
    y = A(1,2);
    sz = A(1,3);
    D1 = zeros(sz, sz);
    D1 = GetImage(I, [x y sz]);
    D2 = imresize(D1, 0.5);

    switch tipe
        case 0,
            S = D2;
        case 1,
            S = flip(D2,2);
        case 2,
            S = flip(D2,1);
        case 3,
            S = imrotate(D2, 180);
        case 4,
            S = flip(D2,3);
        case 5,
            S = imrotate(D2, 90);
        case 6,
            S = imrotate(D2, 270);
        case 7,
            S = flip(D2,4);
    end
```