**Resumo do primeiro ano da bolsa.**

No primeiro ano da bolsa foram realizados estudos sobre imagens digitais de mamogramas com o objetivo de detectar as microcalcificações.

1. Estudo dos Digitalizadores

Este estudo inclui a ánalise dos tipos de digitalizadores e os parâmetros de digitalização de imagens.

Este estudo foi direcionado para a digitalização de imagens mamográficas. Desta forma foram digitalizadas várias imagens para procurar os melhores resultados.

Concluímos porém que para uma inspeção visual seria melhor utilizar a ferramenta do auto ajuste. Foi realizado também um estudo dos digitalizadores encontrados na literatura.

1. Estudo dos filtros de imagens e aplicação destes sobre os filtros.

Neste estudo foram utilizados os filtros morfológicos e de convolução. Além disso estes filtros form aplicados sobre as imagens obtidas no primeiro semestre. Nosso objetivo foi o de buscar a melhor visualização as microcalcificações.

Concluímos porém que estes filtros tem eficácia condicionada a aplicação conjunta com técnicas de segmentação e sua aplicação isolada tem valor apenas para uma inspeção visual.

**Resumo das atividades desenvolvidas no 3.o semestre da bolsa.**

Estudo das técnicas de segmentação encontradas na literatura, utilizadas para segmentação de microcalcificações. Aplicação dos filtros selecionados no primerio ano da bolsa em conjunto com técnicas de segmentação, para garantir o mínimo de falsos positivos.

**Resumo das atividades desenvolvidas no 4.o semestre da bolsa.**

Estudo teórico, Implementação do algoritmo de watershed e análise dos resultados.

**Dificuldades encontradas.**

Além de ser necessário um conhecimento prévio de linguagem pascal, delphi, foi

preciso estudar linguagem C para adaptar o código.

Carga horária elevada no curso, prejudicou o andamento da pesquisa de iniciação científica.

1. **Parte Teórica.**
2. **Introdução**

Quando aplicamos um determinado algoritmo de segmentação numa imagem mamográfica, temos como objetivo principal destacar as estruturas que conhecemos como microcalcificações. Entretanto podemos, contrariando nosso objetivo, aglutinar, ou juntar duas regiões que deveriam ser consideradas em separado.

Como exemplo, muitas das técnicas de segmentação fazem uso da operação morfológica de dilação. Em certos casos este “crescimento”, aumento de

pixels sobre uma região, pode fazer com que tenhamos duas microcalcificações sobrepostas, assim não teremos uma visão correta do objeto.

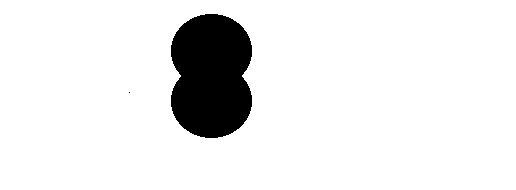
O algoritmo de Watershed , que siginifica “divisor de águas” faz esta diferenciação, e nos permite determinar se dois objetos estão sobrepostos, e também separá-los.

De acordo com [ ], este algoritmo funciona para um conjunto de regiões

convexas conectadas. A idéia básica é separar as estruturas conectadas.

1. **Descrição do algoritmo.**

Considere a região abaixo.



***figura 1. Exemplo de duas estruturas sobrepostas.***

Se a distância de cada célula a um pixel no fundo ( a menor distância) é determinada, podemos determinar uma figura tri- dimensinal, onde a distância ao fundo, seria o eixo que nos dá a altura da figura. Quando procedemos desta maneira obtemos uma duas regiões que se assemelham a duas montanhas com dois picos distintos ligadas por um cume. Este cume no meio dos dois picos representam a máximas distância dos picos que corresponde a curva que tem a máxima distância dos picos e que permanecem dentro da região. O cume será localizado , e alguns dos pixels serão apagados, obtendo-se assim duas regiões.

Inicialmente calcula-se a transformada da distância de cada pixel até o fundo. Isto é feito em dois passos através da imagem.

No primeiro passo, começando do pixel superior a direita (0,0), e movendo-se

nas linhas da esquerda para a direita. Os pixels objeto são identificados e recebem um valor numérico igual a distância de cada pixel a um pixel na fronteira da imagem, acima ou abaixo do pixel.

O segundo passo, começa do pixel mais abaixo e a direita. Move-se então da direita para a esquerda através das linhas e acima na direção das colunas. É calculado a distância de cada pixel objeto a uma borda abaixo ou a direita. Se esta nova distância for menor que o valor existente, ele é armazenado como um novo pixel. Quando completo este procedimento, cada pixel representa a distância dos pixels ao fundo.

Podemos perceber dois picos, cada um tendo um valor 8, e representando o centro da célula. Considere estes dois como sendo os pixels semente para uma dilação. Se camadas de pixels são adicionadas a estes pixels simultaneamente , as duas regiões sendo aumentadas , encontrar-se-ão no cume. Deve-se entretanto tomar cuidado para que duas regiões distintas sejão conectadas no processso.

Inicialmente, todos os pixels tendo o valor máximo são marcados, então todos os pixels que tenham um valor de um ou menos do que o máximo, e que são também vizinhos do pixel marcado são eles próprios marcados , se e somente se eles não conectam duas regiões.

Esta condição é testada usando máscaras , ou templates Por exemplo o pixel central em cada um dos casos da figura 2 não seria marcado pois conectaria duas regiões.

\* \* o \* o \* \* o o

o . o \* . \* o . o

o \* \* o o o \* o \*

***Figura 2***

Desde que um pixel tem 256 possíveis máscaras para serem testadas, para testar a regra de “não conectar novas regiões” é preciso utililzar uma “look up table” , LUT. Esta é indexada por um inteiro no intervalo de de 0 -255. O índice para a tabela é calculado atribuindo-se uma posição de bit para um dos 8 pixels vizinhos formando um inteiro de 8 bits da região 3x3. O pixel central , ou alvo, não é usado quando construímos o índice.

Aplicamos a máscara abaixo para este processo:

128 64 32

16 \* 8

4 2 1

A matriz 3x3 dos pixels é aplicada a matriz 3x3 de pesos comos seria feito

para uma máscara de convolução. Neste caso, desde que o peso é uma potência única de

2, o resultado pode ser transformado em um número de 8 bits (inteiro) e usado como índice para uma tabela. Assim, por exemplo, considere a região 3x3 abaixo:

1 1 0

0 \* 0

0 1 1

O índice da LUT será:

128 + 64 +0

+0 +0

+0 +2 +1= 194= (11000011), representação binária.

Esta condição resultaria dem 2 regiões sendo conectadas , a entrada da LUT deverá ser 0, ou falsa, significando que “não adicione um pixel aqui”. Desde que existem somente 256 possíveis padrões , é possível codifica-los em um curto período de tempo. Isto é feito facilmente por um programa que imprime todos os padrões possíveis , um de cada vez, e pergunta ao usuário para a entrada ao usuário.

O algoritmo pode ser resumido como abaixo:

**è** Calcular a transformada da distância

è Localizar os valores da matriz D proveniente do passo anterior mais altos que são aqueles objetos que estão mais distantes da borda. Estes pixels terão valor K.

è Agora começando nos picos de D, marcamos aqueles pixels que :

1. São adjacentes a um pixel com valor K
2. tem valor K-1
3. não conectar duas regiões marcadas mas não conectadas.

O passo iii) é implementado usando a tabela de LUT como discutido. Agora o valor de K é decrementado em 1, o processo se repete até que K torna-se 0.

**2 Materiais e métodos**

**2.1 Introdução**

Os procedimentos consistiram de adaptar o algoritmo de Watershed descrito em

[Parker 94 ] para linguagem **Delphi**, e no teste deste algoritmo. Em seguida verificou-se a eficiência deste aplicando-se o algoritmo em regiões de interesse.

O método de segmentação de Watershed foi escolhido pois não havia sido implementado neste laboratório. Inicialmente foi feito um estudo da descrição deste algoritmo Em seguida, procedemos com a adaptação do código C contido no livro para linguagem **Delphi**, para depois implementa-lo e testa-lo em regiões já previamente selecionadas nos estágios anteriores da bolsa.

A linguagem **Delphi** foi escolhida pois apresenta maior facilidade para manipular imagens em relação à outras linguagens. Isto permite que o tempo hábil seja utilizado para

implementação do algoritmo em si, ao invés de perdermos grande parte do tempo “alocando” memória para podermos utilizar as imagens.

Quanto ao teste deste algoritmo, procedemos de duas maneiras: aplicamos este à

imagens simuladas, criadas no **paint brush** do **Windows 95** e também em regiões de interesse, ou seja, pequenos trechos de imagens maiores.

Este algoritmo atua para separar duas regiões distintas, neste caso específico duas

microcalcificações, que tenham sido aglutinadas por algum algoritmo de segmentação. Entretanto este algoritmo, não foi desenvolvido para regiões de tamanho reduzido como as microcalcificações, podendo haver portanto efeitos não desejados durante o processo.

Além disso, o algoritmo descrito em [Parker 94], funciona mais eficazmente para objetos

que possuem entre si, formas semelhantes, como sementes ou grãos.

1. **Métodos**

**2.2.1 Sobre a manipulação das imagens**

O Delphi nos proporciona uma fácil manipulação de imagens de **bitmaps**. Tudo que é necessário é adicionar um componente **Timage.** Este nos permite facilmente manipular o conteúdo de uma imagem.

Através do métodos **LoadFromFile** e **SaveToFile** podemos ler e salvar imagens.

Uma vez que tenhamos utilizado **LoadFromFile** para abrir uma imagem no componente **Timage**, podemos através de um simples “loop for”, enviar o contéudo de uma imagem,

para uma matriz, para em seguida trabalharmos com esta.

Para isso utilizamos do método **GetRvalue(image1.Canvas.Pixels[i,j]),** ele nos permiteacessar os elementos da imagem, através dos índices i e j. Uma vez processada, esta matriz, para enviar o conteúdo da matriz processada para aimagem utilizamos o seguinte procedimento :

**Image2.Canvas.Pixels[i,j] := RGB (ValorPixel, ValorPixel, ValorPixel**);

**Image2** é um objeto **Timage** onde queremos mostrar a imagem processada. Com este

procedimento, e utilizando a propriedade **image.canvas.pixels[i,j]** para receber o valor

de pixel que desejamos armazenar na posição [i,j] da imagem, podemos mostrar o resultado da matriz processada pelo método de watershed.

**2.2.2Aplicação do algoritmo.**

Para testar o algoritmo utilizamos imagens, regiões de interesse, e imagens de **bitmaps** editadas no **Paint Brush** do **Windows 95.** Assim apenas abríamos as imagens e em seguida aplicávamos o algoritmo.

1. **Listagem do Programa.**

Abaixo apresentamos a listagem do programa implementado.

|  |
| --- |
| unit Unit1;  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Adaptação e implementação do Algoritmo de Watershed descrito no livro  Pratical Computer Vision using C, de J.R Parker, nas páginas 305 até 308.    Este programa aplica o algoritimo de Watershed numa imagem de bit maps  (ou imagens com extensão bmp), com o objetivo de segmentar esta imagem,  e separar regiões aglutinadas.  Em [] o algoritmo está descrito em linguagem C. Optamos no entanto pelo  Delphi pela facilidade de manipulação de imagens, além do fato do bolsista  estar mais familiarizado com a linguagem Pascal.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  interface  uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  Menus, ExtCtrls, StdCtrls;  type  TMatImagem = array [0..500,0..500] of Longint;  {o tipo TMatImagem é uma matriz de dimensão de 500x500  entretanto a dimensão da matriz tem a dimensão da imagem}  Tvetor = array [0..255] of Longint;  {O tipo Tvetor é um vetor de dimensão 500 utilizado no procesimento Lut}  TForm1 = class(TForm)  MainMenu1: TMainMenu;  OpenDialog1: TOpenDialog;  SaveDialog1: TSaveDialog;  Image1: TImage;  File1: TMenuItem;  Load1: TMenuItem;  Save1: TMenuItem;  Exit1: TMenuItem;  Algoritmo1: TMenuItem;  Watershed1: TMenuItem;  Image2: TImage;  Config1: TMenuItem;  val1: TMenuItem;  val2: TMenuItem;  procedure Exit1Click(Sender: TObject);  procedure Load1Click(Sender: TObject);  procedure Save1Click(Sender: TObject);  procedure Watershed1Click(Sender: TObject);  private  {Private declarations }  {variáveis globais utilizadas em todos os procedimentos}  A,B:TMatImagem;  lut:TVetor;  nr,nc: Longint;  Procedure waters(var val:longint);  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  procedimento de watershed, este segmenta regiões, e tenta desconectar  regiões convexas.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  Procedure lutint;  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Este procedimento inicializa um vetor que é utilizado na Look Up Table,  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  Procedure copy;  { \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Este procedimento copia uma matriz A para uma outra matriz B  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  Procedure remarca(var v1:Longint; var v2 :Longint);  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Este procedimento remarca com valor V2qualquer elemento de uma matriz  A caso este elemento seja igual ao valor v1  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  Function dt(var val: Longint): Longint;  { \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Esta função calcula a transformada da distância de uma Imagem, representada  como uma matriz.Os pixels com valores val são trocados pela sua distância  ao fundo da imagem.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  Function wslev(var lev:Longint):Longint;  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  função que testa a condição de conectar ou não as regiões que estã  sendo crescidas.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  public  { Public declarations }  end;  var  Form1: TForm1;  implementation  {$R \*.DFM}  {Inicio do código do procedimento de Watershed}  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  Procedure Tform1.waters(var val:longint);  var  i,j,k,n:integer;  max:Longint;  begin  lutint;  max:= dt(val);  for i:=0 to nr-1 do  for j:=0 to nc-1 do  if A[i,j] = max then A[i,j]:= 255;  max:= max-1;  copy ;  while max > 0 do  begin  while n <> 0 do  begin  n:=wslev(max);  end;  max:= max-1;  end;  for i:=0 to nr-1 do  for j:=0 to nc-1 do  begin  if (A[i,j]<>0)and(A[i,j]<>255) then A[i,j]:= 0;  end;  end;  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  {fim do algoritmo de watershed}  {inicio do procedimento Lut}  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  procedure TForm1.lutint;  begin  {0-40}  lut[0]:=0;lut[1]:=1;lut[2]:=1;lut[3]:=1;lut[4]:=1;lut[5]:=0;lut[6]:=1;  lut[7]:=1;lut[8]:=1;lut[9]:=1;lut[10]:=1;lut[11]:=1;lut[12]:=0;lut[13]:=0;  lut[14]:=1;lut[15]:=1;lut[16]:=1;lut[17]:=0;lut[18]:=1;lut[19]:=1;lut[20]:=1;  lut[21]:=0;lut[22]:=1;lut[23]:=1;lut[24]:=0;lut[25]:=0;lut[26]:=1;lut[27]:=1;  lut[28]:=0; lut[29]:=0;lut[30]:=1;lut[31]:=1;lut[32]:=1;lut[33]:=0;lut[34]:=0;  lut[35]:=0;lut[36]:=0;lut[37]:=0;lut[38]:=0;lut[39]:=0;lut[40]:=1;  {41-80}  lut[41]:=1;lut[42]:=1;lut[43]:=1;lut[44]:=0;lut[45]:=0;lut[46]:=1;  lut[47]:=1;lut[48]:=0;lut[49]:=0;lut[50]:=0;lut[51]:=0;lut[52]:=0;  lut[53]:=0;lut[54]:=0;lut[55]:=0;lut[56]:=0;lut[57]:=0;lut[58]:=1;  lut[59]:=1;lut[60]:=0;lut[61]:=0;lut[62]:=1;lut[63]:=1;lut[64]:=1;  lut[65]:=0;lut[66]:=0;lut[67]:=0;lut[68]:=0;lut[69]:=0;lut[70]:=0;  lut[71]:=0;lut[72]:=1;lut[73]:=1;lut[74]:=0;lut[75]:=1;lut[76]:=0;  lut[77]:=0;lut[78]:=1;lut[79]:=1;lut[80]:=1;  {81-120}  lut[81]:=0;lut[82]:=1;lut[83]:=1;lut[84]:=1;lut[85]:=0;lut[86]:=1;  lut[87]:=1;lut[88]:=1;lut[89]:=1;lut[90]:=1;lut[91]:=1;lut[92]:=1;  lut[93]:=1;lut[94]:=1;lut[95]:=1;lut[96]:=1;lut[97]:=0;lut[98]:=0;  lut[99]:=0;lut[100]:=0;lut[101]:=0;lut[102]:=0;lut[103]:=1;lut[104]:=1;  lut[105]:=1;lut[106]:=1;lut[107]:=1;lut[108]:=0;lut[109]:=0;lut[110]:=1;  lut[111]:=1;lut[112]:=1;lut[113]:=0;lut[114]:=1;lut[115]:=1;lut[116]:=1;  lut[117]:=0;lut[118]:=1;lut[119]:=1;lut[120]:=1;  {121-200}  lut[121]:=1;lut[122]:=1;lut[123]:=1;lut[124]:=1;lut[125]:=1;lut[126]:=1;  lut[127]:=1;lut[128]:=1;lut[129]:=0;lut[130]:=0;lut[131]:=0;lut[132]:=0;  lut[133]:=0;lut[134]:=0;lut[135]:=0;lut[136]:=0;lut[137]:=0;lut[138]:=0;  lut[139]:=0;lut[140]:=0;lut[141]:=0;lut[142]:=0;lut[143]:=0;lut[144]:=1;  lut[145]:=0;lut[146]:=1;lut[147]:=1;lut[148]:=1;lut[149]:=0;lut[150]:=1;  lut[151]:=1;lut[152]:=0;lut[153]:=0;lut[154]:=1;lut[155]:=1;lut[156]:=0;  lut[157]:=0;lut[158]:=1;lut[159]:=1;lut[160]:=0;lut[161]:=0;lut[162]:=0;  lut[163]:=0;lut[164]:=0;lut[165]:=0;lut[166]:=0;lut[167]:=0;lut[168]:=0;  lut[169]:=0;lut[170]:=0;lut[171]:=0;lut[172]:=0;lut[173]:=0;lut[174]:=0;  lut[175]:=0;lut[176]:=0;lut[177]:=0;lut[178]:=0;lut[179]:=0;lut[180]:=0;  lut[181]:=1;lut[182]:=1;lut[183]:=0;lut[184]:=0;lut[185]:=1;lut[186]:=1;  lut[187]:=1;lut[188]:=0;lut[189]:=0;lut[190]:=0;lut[191]:=1;lut[192]:=1;  lut[193]:=0;lut[194]:=0;lut[195]:=0;lut[196]:=0;lut[197]:=0;lut[198]:=0;  lut[199]:=0;lut[200]:=1;  {201-255}  lut[201]:=1;lut[202]:=1;lut[203]:=1;lut[204]:=0;lut[205]:=0;lut[206]:=1;  lut[207]:=1;lut[208]:=1;lut[209]:=0;lut[210]:=1;lut[211]:=1;lut[212]:=1;  lut[213]:=0;lut[214]:=1;lut[215]:=1;lut[216]:=1;lut[217]:=1;lut[218]:=1;  lut[219]:=1;lut[220]:=1;lut[221]:=1;lut[222]:=1;lut[223]:=1;lut[224]:=1;  lut[225]:=0;lut[226]:=0;lut[227]:=0;lut[228]:=0;lut[229]:=0;lut[230]:=0;  lut[231]:=0;lut[232]:=1;lut[233]:=1;lut[234]:=1;lut[235]:=1;lut[236]:=0;  lut[237]:=0;lut[238]:=1;lut[239]:=1;lut[240]:=1;lut[241]:=0;lut[242]:=1;  lut[243]:=1;lut[244]:=1;lut[245]:=0;lut[246]:=1;lut[247]:=1;lut[248]:=1;  lut[249]:=1;lut[250]:=1;lut[251]:=1;lut[252]:=1;lut[253]:=1;lut[254]:=1;  lut[255]:=1;  end;  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  {fim do procedimento lut}  {Inicio do procedimento copy}  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  procedure Tform1.copy;  var  i,j:integer;  begin  for i:=0 to nr-1 do  for j:=0 to nc-1 do  begin  B[i,j]:= A[i,j];  end;  end;  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  {fim do procedimento copy}  {inicio do procedimento remarca}  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  procedure tform1.remarca(var v1:Longint; var v2 :Longint);  var  i,j: Longint;  begin  for i:=0 to nr-1 do  for j:=0 to nc-1 do  if (A[i,j]=V1) then A[i,j]:= v2;  end;  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  {fim do procedimento de remarcar}  {funcao que calcula a transformada da distancia}  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  function tform1.dt(var val: Longint): Longint;  var  i,j,k,n,m,ii,jj,v1,v2:Longint;  begin  v1:=val;  v2:=254;  remarca(v1,v2);  v1:=255;  v2:=0;  remarca(v1,v2);  n:=0;  m:=1;  while m<>0 do  begin  m:= 0;  for i:= 0 to nr-1 do  for j:= 0 to nc-1 do  for ii:= -1 to 1 do  begin  if A[i,j]<>254  then continue;  if ((i + ii)<0)or((i + ii)>=nr)then continue;  for jj:= -1 to 1 do  begin  if (ii<>0)and(jj<>0) then continue;  if ((j+jj)<0)or((j+jj)>=nc) then continue;  if A[i+ii,j+jj]= n then  begin  A[i,j]:= n+1;  m:=1;  end;  end;  end;  n:=n+1;  end;  dt:= n-1;  end;  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  {fim da função que calcula a transformada da distância}  {Inicio da funçao Wlev}  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  function tform1.wslev(var lev:Longint):Longint;  var  ind,i,j,k,n,m: Longint;  mask : array [0..20,0..20] of Longint;  begin  mask[2,2]:= 01;  mask[2,1]:= 02;  mask[1,2]:= 04;  mask[1,2]:= 010;  mask[1,0]:= 020;  mask[0,2]:= 040;  mask[0,1]:= 0100;  mask[0,0]:= 0200;  k:=0; ind:=0;  for i:=1 to (nr-2) do  for j:=1 to (nc-2) do  begin  if A[i,j] <> lev then continue;  ind:= 0;  for n:=-1 to 1 do  begin  for m:=-1 to 1 do  begin  if (n=0)and(m=0)then continue;  if A[i+n,j+m] = 255 then  ind := 1;  end;  end;  if ind=0 then continue;  ind:=0;  for n:=-1 to 1 do  for m:=-1 to 1 do  begin  if (n=0)and(m=0) then continue;  if B[i+n,j+m]= 255 then ind:=ind + mask[n+1,m+1];  end;  if lut[ind]<>0 then  begin  B[i,j]:= 255;  k:=1;  end;  end;  if k<>0 then copy;  wslev:=k;  end;  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  {fim do procedimento de wlev}  {Procedimentos ligados aos objetos visuais que compõe  o programa}  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  procedure TForm1.Exit1Click(Sender: TObject);  begin  close;  end;  {carrega uma imagem para o objeto image1}  procedure TForm1.Load1Click(Sender: TObject);  begin  with OpenDialog1 Do  if execute Then  Begin  with Image1.picture do  begin  LoadFromfile(Filename);  end;  Caption := ExtractFilename(Filename);  end;  end;  {salva uma imagem ligada ao objeto image2}  procedure TForm1.Save1Click(Sender: TObject);  begin  with Savedialog1 do  if execute then  begin  with image2.picture do  begin  Savetofile(filename);  end;  end;  end;  {fim dos procedimentos ligados a parte visual}  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  {inicio do subprograma que chama o procedimento de watershed  além de transformar as imagens , arquivos bmp em matrizes}  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  procedure TForm1.Watershed1Click(Sender: TObject);  var  valorpixel,i,j,val: Longint;  begin{inicio do subprograma principal}  {Podemos acessar a dimensão da matriz através das propriedades  abaixo, obtendo assim sua dimensão heightxwidht}  {Deste modo trabalhamos com matrizes, facilitando  tanto o cálculo quanto o entendimento.  Uma vez lida a imagem podemos facilmente  armazenar o seu conteúdo numa matriz para podermos trabalhar  com esta}  nr:= image1.picture.bitmap.height;  nc:= image1.picture.bitmap.width;  {copia imagem para uma matriz A}  for i:=0 to nr-1 do  for j:=0 to nc-1 do  begin  A[i,j]:=GetRvalue(image1.Canvas.Pixels[i,j]);  end;  {fim do procedimento que copia imagem para  a matriz A}  val:=10;  Screen.Cursor:= crHourGlass;  waters(val);{chama procediemto de watershed para  ser aplicado sobre a matriz }  for i:=0 to nr-1 do  for j:=0 to nc-1 do  begin  valorpixel:= A[i,j];  Image2.Canvas.Pixels[i,j] := RGB (ValorPixel, ValorPixel, ValorPixel);  end; {imagem final é mostrada}  Screen.Cursor:= crDefault;  end;  {fim do subprograma que le, aplica o watershed e mostra a imagem}  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  {fim do programa principal}  {\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}  end. |

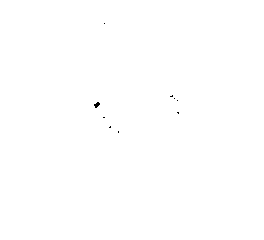
**3 Resultados.**

A imagem seguinte é uma região de interesse extraída de uma mamografia, digitalizada e selecionada em semestres anteriores da bolsa de Iniciação Cientifica.



***Figura3- Região de interesse contendo microcalcificações***

Após aplicarmos o algoritmo de Watershed , obtemos a imagem contida na figura 4:



***Figura 4-Resultado da segmentação “watershed”***

Podemos observar que as microcalcificações são destacadas, após a aplicação deste algoritmo, inclusive com a eliminação de parte do ruído de fundo da imagem.

Utilizando outra região de interesse de uma mamografia pré- selecionada , mostrada abaixo:



***Figura 5- Região de interesse contendo as microcalcificações.***

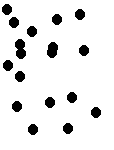
Com a aplicação do algoritmo obtemos a imagem mostrada na figura 6:



***Figura 6- Resultado da segmentação “watershed”***

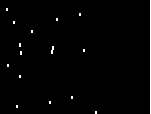
As microcalcificações são destacadas, no entanto, bastante ruído de fundo também é.

Editamos uma imagem de **bit maps** de modo a simular uma região contendo objetos de formas semelhantes, que poderiam estar aglutinados. O algoritmo implementado deveria separar estas regiões. Devemos atentar para os objetos que estão sobrepostos.



***Figura7- Imagem de Bit maps***

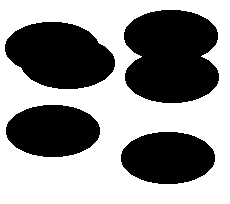
Ao aplicarmos o algoritmo na imagem acima, obtemos o seguinte resultado, apresentado na figura 8.



***Figura 8- Resultado do Algoritmo de Watershed sobre a imagem da figura 7.***

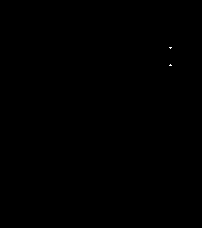
Podemos verificar na figura 8 que, apesar de perdermos a informação sobre a sua forma, os objetos sobrepostos foram separados. Deste modo se nosso objetivo fosse contar o número de objetos na imagem, o procedimento de Watershed seria de grande auxilio.

Editamos uma outra imagem de bit maps (desta vez utilizando uma forma diferente para os objetos a serem separados) apresentada na figura baixo:



***Figura 9- Imagem de Bit maps***

O resultado da aplicação do algoritmo de Watershed esta apresentado na figura 10.



***Figura 10- Imagem resultante da aplicação do Watershed sobre a imagem da figura 9.***

Na figura 10 observamos dois pares de objetos sobrepostos. Na imagem

processada apenas um destes pares é separado. O outro par, e outros objetos não conectados, são no entanto perdidos . Entretanto o par que pode ser mais facilmente considerado como sendo dois objetos conectados, é separado. O outro par não separado, foi interpretado como sendo uma estrutura única. Podemos perceber que a forma do objeto influi na atuação do algoritmo. Objetos como discos são reconhecidos enquanto que objetos de forma cilíndrica não o são, se sua forma é estreita.

**4 Conclusões.**

Quando aplicado em regiões de interesse o algoritmo apresenta uma boa eficiência, no processo de segmentar as microcalcificações. Entretanto o teste em imagens em mamogramas completos deveria ser tentado futuramente. Aqui devido a dificuldade do processamento: lentidão, impossibilidade de comprovar totalmente a eficiência em mamogramas completos, inviabilizam este procedimento.

Entretanto, este algoritmo foi desenvolvido para segmentar estruturas de dimensões maiores do que as microcalcificações. Estas apresentam um tamanho bastante reduzido, algo que pode comprometer a eficiência do algoritmo. Faz-se necessário um aperfeiçoamento deste algoritmo futuramente, de modo a adaptar seus procedimentos às características das microcalcificações.

O algoritmo quando usado para separar objetos convexos conectados, apresenta uma boa eficiência como mostrado nas figuras 8 e 10. Entretanto nesta última, houve perda do sinal original.

Além disso, é preciso verificar a atuação do Watershed em conjunto com outras

técnicas de segmentação, abrindo um campo novo de possibilidades.

Quanto às técnicas de digitalização, ainda é preciso buscar os parâmetros ótimos para este processo Mais ainda: devemos definir qual critério é melhor para digitalzarmos

uma mamografia, o visual ou àqueles parâmetros que possibilitam um melhor resultado no processamento. Concluímos no entanto, que a melhor “ferramenta” para a digitalização, é utilizar os auto ajuste que acompanha o sistema do digitalizador. Isto é claro levando em conta o critério visual.

As técnicas de pré processamento (filtros) auxiliam sobremaneira na inspeção visual das microcalcificações em mamogramas digitais. Entretanto para efeito de processamento elas são ineficientes.

Quanto às técnicas de segmentação elas demandam ainda que as características das microcalcificações sejam adaptadas à elas, para obtermos resultados mais expressivos.

**5 Referências bibliográficas.**

[Carroll 88]

**Carrol, D.W; Programação em Turbo Pascal. Makron Books do Brasil Editora Ltda, 1988.**

[Holzner 95]

**Holzner, S; C Programação. Editora Campus 1993.**

[Mueller 96]

**Mueller, Jonh; Guia para o Delphi 2. Makron Books do Brasil Editora Ltda, 1997.**

[Parker 94]

**Parker, J.R; Pratical Computer Vision Using C. Jonh Wiley & Sons, Inc 1994.**

[Ruben 95]

**Rubenking, N.J; Programação em Delphi para Leigos. Berkley Brasil Editora, 1995.**