PROGRAMAÇÃO E APERFEIÇOAMENTO DA APLICAÇÃO DO ALGORITMO WATERSHED PARA A SEGMENTAÇÃO DE GALÁXIAS BASEADO EM DADOS ESPECTROGRÁFICOS.

Murilo Moritz Parize[1] *- Marcelo Massocco Cendron*[2]

INTRODUÇÃO

A necessidade do desenvolvimento de tecnologias para a segmentação de dados espectrográficos de galáxias é de extrema importância para a astronomia, para delimitação de padrões e separação dos tipos de galáxias. Nesse intuito se faz necessário o aperfeiçoamento de técnicas e algoritmos computacionais que realizam tal atividade.

O algoritmo estudado e que foi aperfeiçoado é o *Watershed,* sendo uma técnica de segmentação de imagens baseada na divisão de superfícies em bacias hidrográficas. Grupos de pesquisas tem buscado classificar esses dados, de forma que sejam cruzados e novas bases sejam geradas. Inicialmente foram encontradas as diferenças entre os espectros de cada projeto, assim foram obtidos os valores que serão segmentados através da aplicação do método em questão.

Como o método escolhido gera uma alta demanda computacional é necessária uma melhoria no que diz respeito ao tempo de processamento, sendo proposto a utilização da Linguagem C juntamente com alguns *frameworks,* sendo eles o OpenCv, utilizado na segmentação de imagens e o OpenCl, que proporciona a programação para processamento diretamente na GPU (Unidade de Processamento Gráfico).

PROCEDIMENTOS METODOLÓGICOS

[1] Aluno do Instituto Federal Catarinense de Educação Ciência e Tecnologia, Campus Videira. Bacharelado em Ciência da Computação. E-mail: murilo.moritz33@gmail.com
[2] Professor Orientador do Instituto Federal Catarinense de Educação Ciência e Tecnologia, Campus Videira. E-mail: marcelo.cendron@ifc-videira.com.br

Para o desenvolvimento do projeto foram definidos algumas técnicas como citados anteriormente, sendo OpenCv e OpenCl a ferramenta ou IDE utilizada para a realização dessas tarefas foi o Code::Blocks.

O microcomputador utilizado para os experimentos e estudo das técnicas possui a seguinte configuração:

a) Processador Intel® Core2 Duo 2.1GHz
b) 3 GB de memória RAM
c) Sistema Operacional Windows 7 de 32 bits

O OpenCv é uma biblioteca de código aberto para visão computacional é gratuito para o uso acadêmico e comercial, possuindo interfaces de trabalho com diversas linguagens como C++, C, Python e Java sendo possível suporte em sistemas operacionais Windows, Linux, Mac Os, iOS e Android. Sua estrutura básica foi desenvolvida em C/C++, sendo possível o uso de processamento multi-core ou seja processamento paralelo. Essa ferramenta foi instalada diretamente suas bibliotecas na IDE para o desenvolvimento das atividades bem como as bibliotecas do OpenCL.

O OpenCL é um framework livre para programação de alto desempenho utilizada em ambientes computacionais equipados com CPU's multicore e GPU's many-core. Com o foco voltado para o paralelismo a programação consiste em funções que são executadas em múltiplos núcleos simultaneamente.

Após o tratamento dos dados no formato escolhido deu-se início a fase de geração de imagens e segmentação das imagens através do algoritmo Watershed, este foi aplicado de modo progressivo até encontrar a separação efetiva dos grupos de galáxias de modo que fosse possível delimitar com extrema clareza as propriedades espectrográficas encontradas em cada um desses grupos.

O algoritmo utilizado para a geração e segmentação de imagens utilizando-se de ferramentas do OpenCV, é possível observar que o processo de tratamento é progressivo sendo possível gerar imagens passo a passo da execução. No caso do tratamento de imagens intermediárias causa uma carga computacional maior, pois é preciso processar a imagem anterior e gravar a nova imagem, para que a mesma

seja tratada no passo seguinte. Esse processo foi realizado para fazer um acompanhamento e uma comparação dos resultados encontrados em cada um dos passos até chegarmos aos marcadores finais, abaixo temos algumas partes do algoritmo criado.

```
using namespace cv;
using namespace std;

int erosion_elem = 0;
int erosion_size = 1;
int dilation_elem = -1;
int dilation_size = 6;

int main()
{
   cv::Mat w0, w2, w3, w4;
   double timestamp = (double)getTickCount();
   CvMLData w;
   w.read_csv("arquivo2.csv"); // w = load('arquivo.csv');
   const CvMat* tmp = w.get_values();
   cv::Mat w1(tmp, true);

   w1.convertTo(w1, CV_8UC3);
   cvtColor(w1, w0, CV_GRAY2BGR );
   threshold( w1, w1, 20, 255 ,cv::THRESH_BINARY ); // w2 = im2bw(w1, 0.2);

   Mat element = getStructuringElement( MORPH_ELLIPSE,
                                        Size( 2*erosion_size + 1, 2*erosion_size+1 ),
                                        Point( erosion_size, erosion_size ) ); // controe esse elemento: strel('disk',2)
   erode( w1, w2, element ); // w4 = imopen(w3, strel('disk',2));

   Mat element2 = getStructuringElement( MORPH_ELLIPSE,
                                         Size( 2*dilation_size + 1, 2*dilation_size+1 ),
                                         Point( dilation_size, dilation_size ) );

   dilate( w2, w3, element2 ); // w5 = imdilate(w4, strel('disk',18));
   threshold( w3, w4, 1, 128 , 1 ); // w2 = im2bw(w1, 0.2);

   cv::Mat marcadores(w0.size(),CV_8U,cv::Scalar(0));
   marcadores = w2 + w4;
   marcadores.convertTo(marcadores, CV_32S );

   cv::watershed(w0, marcadores);
   marcadores.convertTo(marcadores,CV_8U);
   timestamp = (double)getTickCount() - timestamp; // get current time in seconds

   cv::waitKey(0);

   printf( "execution time = %gms\n", timestamp*1000./getTickFrequency() );
   return 0;
}
```

Tendo a imagem gerada foram aplicadas as funções de *threshold, erode e dilate* tratando a imagem de maneiras diferentes. O *threshold* realiza o filtro dos valores a serem considerados agindo como um limiar, *erode* realiza a erosão dos pontos próximos a um centro de padrões, por isso é usada juntamente com o *threshold,* e a função *dilate* realiza a dilatação dos pontos. A figura 1 mostra as etapas do processo.

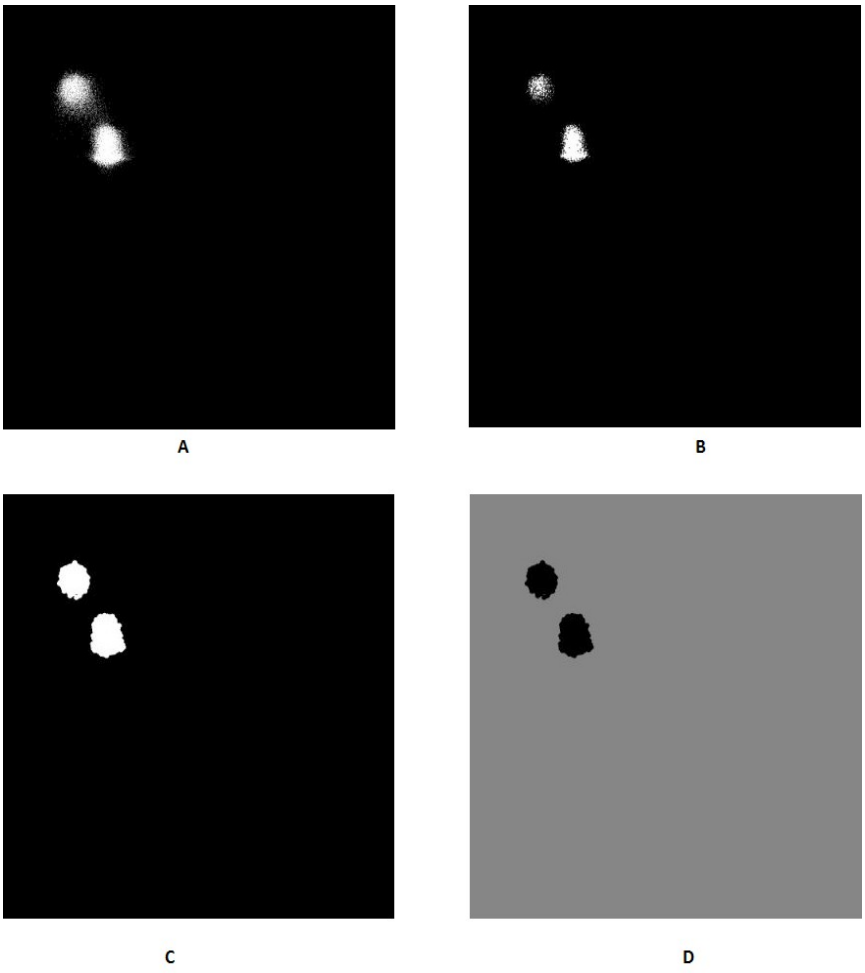

Figura 1: Passo a Passo uso de *threshold*(A-D), *erode*(B) e *dilate*(C)

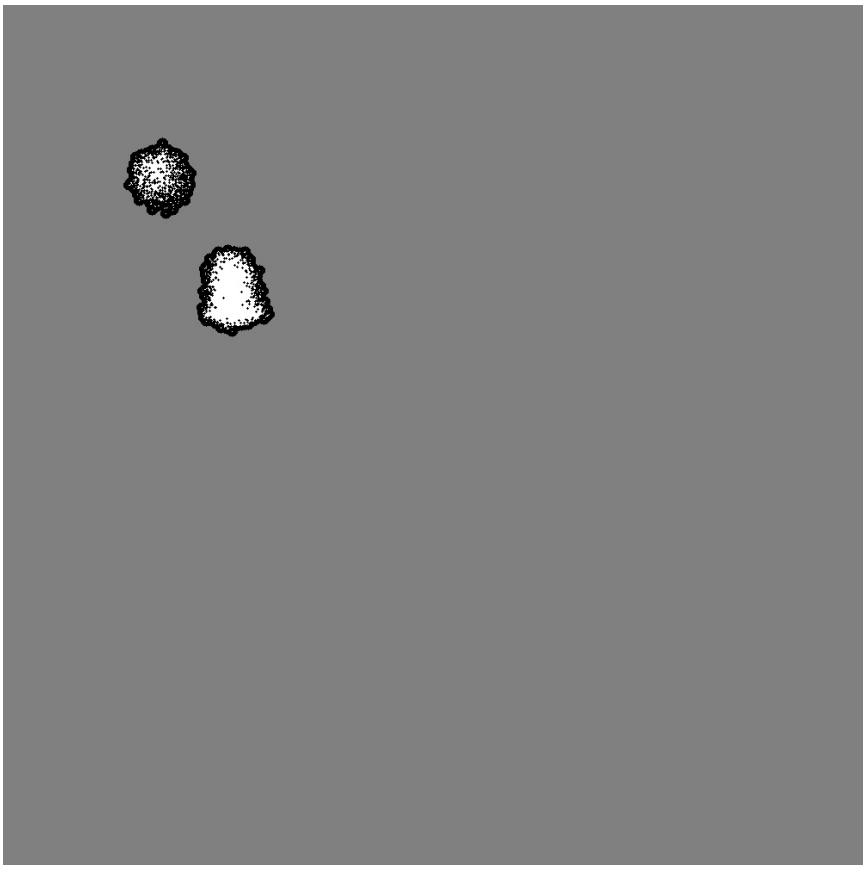

Figura 2: Resultado final segmentação

A separação fica nítida cabendo agora ao especialista da área de estudo das galáxias definir a qual dos grupos ou classificações elas pertencem.

CONSIDERAÇÕES FINAIS

O processo de migração para o código fonte em C++ apresentou a possibilidade de otimizar o código através da placa de vídeo. A dificuldade encontradas é que não há muitas opções de algoritmos otimizados para o processamento através da placa de vídeo. Isso percebemos que é uma oportunidade de pesquisa e desenvolvimento de bibliotecas que podem melhorar ainda mais o processamento desses algoritmos.

REFERÊNCIAS

BEUCHER, S.; LANTUÉJOUL, C. Use of watersheds in contour detection. **International workshop on image processing, real-time edge and motion detection**, Rennes, France, Setembro 1979.

GRUPO DE ASTROFÍSICA DA UFSC. **Grupo de Astrofísica**. UFSC, 2013. Disponivel em: <http://astro.ufsc.br/>. Acesso em: 21 de Julho 2014.

HUCKO, M.; SR AMEK, M. **Streamed Watershed Transform on GPU for Processing of Large Volume Data**. SCCG '12 Proceedings of the 28th Spring Conference on Computer Graphics, New York, NY, USA , 2012. 137-141.

MACQUEEN, J. B. **Some Methods for classification and Analysis of Multivariate Observations**. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, California, 1967. 281–297.

MATHWORKS, INC. **MATLAB - The Language of Technical Computing**. MathWorks, 2013. Disponivel em: <http://www.mathworks.com/products/matlab/>. Acesso em: Março 2014.

MULLIE, L. **Segmenting medical images with CUDA + MPI.** Watershed on the GPU, 2013. Disponivel em: <http://louismullie.com/code/watershed-cuda/index.html>. Acesso em: Maio 2014.

NASA. **Wide-field Infrared Survey Explorer.** NASA, 23 Abril 2013. Disponivel em: <http://www.nasa.gov/mission_pages/WISE/main/index.html>. Acesso em: 14 Maio 2013.

NICKOLLS, J.; DALLY, W. J. **The GPU Computing Era.** Micro, IEEE, 30, n. 2, 12 Abril 2010. 56 - 69.

NVIDIA CORPORATION. **What is GPU Computing?** NVIDIA, 2013. Disponivel em: <http://www.nvidia.com/object/what-is-gpu-computing.html>.

SLOAN DIGITAL SKY SURVEY. **Sloan Digital Sky Survey.** Sloan Digital Sky Survey, 31 julho 2012. Disponivel em: <http://www.sdss.org/>. Acesso em: 14 maio 2014.