Processador em tempo real para classificação de imagens SAR Polarimétricas

José Roberto Colombo Junior
Naiallen Carolyne Rodrigues Lima
David Fernandes

Instituto Tecnológico de Aeronáutica – ITA-DCTA
Praça Eduardo Marechal Gomes, 50, CEP 12228-900, São José dos Campos, SP, Brasil
colombojrj@gmail.com, naiallen@yahoo.com.br, david@ita.br

Abstract. The Synthetic Aperture Radar (SAR) is an imaging sensor whose operation is independent of sunlight, climatic factors and allows great penetration in the vegetation cover. The goal of this paper is to implement in real time a classifier based on the unsupervised classification method, which uses the SAR polarimetric parameters entropy (H), alpha angle (α) and anisotropy (A). The proposed method uses the H-α plan and the H-α-A space to classify the polarimetric SAR image into nine and eighteen classes, respectively. Once this classification method is highly parallelizable, it was implemented to run in real time in a Graphical Processor Unity (GPU), that is a device with great processing capability. As a result it was found that the implemented classifier allows real time operation in an airborne SAR sensor with requirements for defense missions.

Palavras-chave: real time image processing, SAR image classification, SAR polarimetry, processamento de imagem em tempo real, classificação de imagem SAR, polarimetria SAR.

1. Introdução

O Radar de Abertura Sintética (SAR, do inglês *Synthetic Aperture Radar*) é um sensor imageador ativo, de alta frequência, em geral microondas, embarcado em plataformas móveis, aéreas ou orbitais, que é utilizado em operações de sensoriamento remoto para obtenção de imagens (Cumming, I. G.; Wong, 2005), complementando assim as informações de sistemas imageadores ópticos. Dentre os benefícios da utilização de radares de abertura sintética tem-se a possibilidade de imagear regiões onde sensores ópticos são ineficientes, como em condições de nuvens, chuva ou em áreas com vegetação densa quando o interesse é a observação do terreno. Devido as suas características de sensor ativo, o SAR independe de qualquer tipo de iluminação natural ou de emissão própria do alvo.

Dentre as aplicações em tempo real pode-se citar a sua utilização em defesa (detecção de alvos, detecção de alvos em movimento etc), monitoramento ambiental (vigilância de áreas de risco, avaliação de danos etc) e como subsidio a tomada de decisões em uma cadeia de Comando e Controle (C2). Sistemas que operam em tempo real devem fazer a aquisição dos sinais ecos (sinal bruto) a cada pulso transmitido pelo Radar e sintetizam continuamente, por meio de processamento, a imagem da cena de modo que após um tempo de latência (tempo entre a aquisição dos dados brutos e a formação da imagem) aceitável a imagem formada seja continuamente apresentada, na mesma cadência com que os dados brutos são adquiridos.

Chiaratti e Fernandes (2013) fizeram a análise dos requisitos de processamento para a obtenção de imagens SAR em tempo real em função das características do imageamento (modos de operação do SAR) considerando o Algoritmo Range-Dopler para sintetizar as imagens. Chioquetta (2011), Trofino et. al. (2012) e Trofino (2014) implementaram processadores SAR em tempo real utilizando, respectivamente GPU com CUDA e uma CPU de uso geral.

Este trabalho incorpora ao processamento em tempo real a etapa de classificação de imagens SAR polarimétricas, de modo que além das imagens SAR a sua classificação

também é obtida em tempo real. Como a informação contida na imagem classificada pode se restringir a um número inteiro para cada classe presente na imagem, em modos de operação do SAR nos quais somente a imagem classificada deve ser transmitida para uma estação no solo, pode-se obter uma grande redução na taxa de bits a ser transmitida e em consequência ter-se uma transmissão de alta qualidade.

O trabalho realizado está organizado da seguinte forma: no Capítulo 2 descreve-se o método de classificação polarimétrica empregado, no Capítulo 3 mostra-se a arquitetura de implementação do classificador em tempo real, no Capítulo 4 são estabelecidos os requisitos de operação em tempo real, no Capítulo 5 são apresentados os resultados obtidos e no Capítulo 6 são feitos os comentários finais a respeito do trabalho realizado.

2. Classificação Entropia-Alfa-Anisotropia de imagens SAR polarimétricas

A classificação de imagens SAR polarimétricas é realizada empregando métodos baseados em mecanismos de espalhamento de alvos. O ponto de partida para caracterização das ondas retoespalhadas e, consequentemente, para todos os passos na classificação de imagens polarimétricas abordadas nesse trabalho é a Matriz de Complexa de Espalhamento [*S*], de cada pixel do conjunto de imagens polarimétricas, descrita por (Cloude e Pottier, 1996, 1997; Hellmann, 2000 e Villaça, 2008):

$$[S] = \begin{bmatrix} S_{hh} & S_{hv} \\ S_{vh} & S_{vv} \end{bmatrix} \quad (1)$$

sendo S_{hh}, S_{hv}, S_{vh}, e S_{vv} os valores complexos dos pixels das imagens polarimétricas, o índice *h* indica a polarização horizontal e o índice *v* indica a polarização vertical.

A matriz de Espalhamento pode ser reescrita na forma do vetor de Pauli, que descreve a matriz [*S*] como um vetor complexo, que se for considerado o teorema da reciprocidade ($S_{hv} = S_{vh}$) fica (Cloude e Pottier, 1996, 1997):

$$\overrightarrow{K_{p3}} = \frac{1}{\sqrt{2}} \begin{bmatrix} S_{hh} + S_{vv} \\ S_{hh} - S_{vv} \\ 2S_{hv} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \end{bmatrix} \quad (2)$$

A partir desse vetor é construída a matriz de coerência dada por $[T] = \langle \vec{K}_{p3} . \vec{K}_{p3}^{\,H} \rangle$ onde o sobrescrito *H* significa transposta e complexa e $\langle \cdot \rangle$ a média das amostras, calculada em uma vizinhança (de tamanho arbitrário) do pixel considerado.

Cloude e Pottier (1996, 1997) propuseram um método para o cálculo dos parâmetros entropia, anisotropia e ângulo α utilizando os autovetores e autovalores da matriz de coerência. Esses parâmetros são os elementos utilizados para a classificação abordada neste trabalho.

A partir dos autovalores ordenados de [*T*], representados por $\lambda_1 \geq \lambda_2 \geq \lambda_3$ é calculada a Entropia (*H*), a Anisotropia (*A*) e o ângulo alfa (α), expressos respectivamente por:

$$H = \sum_{i=1}^{3} -p_i . log_3 . P_i \,, \quad \text{com} \quad p_i = \frac{\lambda_i}{\lambda_1 + \lambda_2 + \lambda_3} \quad (3)$$

$$A = \frac{\lambda_2 - \lambda_3}{\lambda_2 + \lambda_3} \qquad \text{e} \qquad \alpha = \sum_{i=1}^{3} -p_i \cos^{-1}(k_{xi}) \quad (4)$$

sendo k_{xi} o primeiro elemento do autovetor associado ao autovalor λ_i .

A Entropia ($0 \leq H \leq 1$) é um indicador da aleatoriedade ou desordem do espalhamento nas imagens polarimétricas, $H = 0$ indica que existe apenas um mecanismo de espalhamento determinístico e $H = 1$ indica que não existe um mecanismo de espalhamento dominante. A Anisotropia ($0 \leq A \leq 1$) indica o grau de

contribuição dos mecanismos secundários e o ângulo alfa (α) se refere ao tipo de mecanismo de espalhamento (superfície isotrópica, dipolo, diedros etc).

Cloude e Pottier (1996, 1997) propuseram ainda um método de classificação utilizando um plano 2D, formado por H x α, dividido em nove zonas. Esse plano foi posteriormente estendido para o espaço formado por H-α-A, no qual tem-se nove regiões do plano H-α acima de A =0,5 e nove regiões abaixo deste limiar. Obtém-se assim 18 zonas de classificação: nove zonas H-α para A < 0,5 e mais nove zonas H-α para A ≥ 0,5.

3. Implementação em tempo real do classificador SAR polarimétrico

Cada pixel da imagem SAR polarimétrica está associado, conforme descrito no Capítulo 2, a um vetor de Stokes, definido na Equação (2). As funções das imagens polarimétricas *hh+vv*, *hh-vv* e 2*hv* nada mais são que grandes matrizes e o processamento do pixel (*i*, *j*) independe do processamento do pixel (*m*, *n*), com (*m*, *n*) ≠ (*i*, *j*). Ou seja, o algoritmo possui alto nível de paralelização.

Em virtude disso, a utilização de técnicas de processamento paralelo pode melhorar consideravelmente o desempenho computacional dessas tarefas, que podem ser executadas muito mais rapidamente em uma (ou mais) GPU(s)[1] que em processadores tradicionais (NVIDIA, 2014).

Com as imagens SAR polarimétricas *hh*, *hv* e *vv* em formato compatível com o padrão XDR (*eXternal Data Representation*), elas podem ser carregadas na memória do computador por meio da função OPENR do programa IDL, por exemplo. Esse programa permite executar funções escritas em C/C++ compiladas como bibliotecas dinâmicas (DLL, do inglês *Dynamic Link Library*) através da função CALL_EXTERNAL (IDL/EXELIS, 2014). Assim, a proposta é implementar o classificador em linguagem de programação C++ para ser executado dentro da placa de vídeo e que possa ser chamado de dentro do IDL.

A implementação foi realizada com as seguintes etapas:

a) Com as imagens disponíveis no escopo do IDL (memória RAM do computador), elas são copiadas para a memória global da placa de vídeo, para serem processadas com maior rapidez;
b) Quando o processo de classificação das imagens termina, elas são copiadas da memória da GPU para a memória RAM, estando assim disponíveis no escopo local do IDL;
c) Em seguida o programador tem a opção de gerar gráficos para mostrá-los na tela ou então armazenar a imagem classificada em um arquivo SAV (com uso da função SAVE, do IDL). Esse procedimento pode ser ilustrado com o diagrama de blocos apresentado na Figura 1.

Os algoritmos foram implementados e testados com uso de um computador com processador i7-3970X (3.5GHz), 32 GB de memória RAM, GPU Nvidia Tesla C2075, Microsoft Windows 7 64 bits, IDL 7.0 64 bits, CUDA 6.0 e Microsoft Visual Studio 2010.

[1] Do inglês *Graphics Unit Processor*

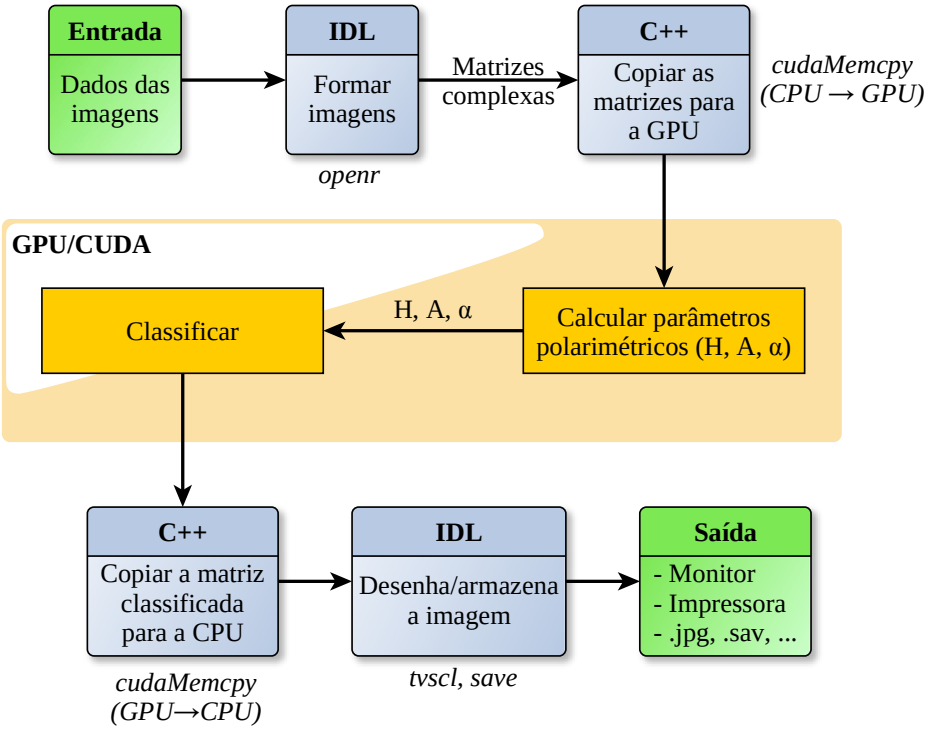


Figura 1. Representação em diagrama de blocos da implementação do classificador de imagens.

A seguir é descrito o procedimento para desenvolver funções escritas em C++ para serem executadas dentro do IDL e é apresentada a implementação experimental do processador e do classificador.

3.1 Programação C++ com IDL

O programa IDL possui uma função chamada CALL_EXTERNAL que permite carregar bibliotecas dinâmicas e executá-las como se fossem parte da API (do inglês, *Application Programming Interface*) de rotinas do próprio IDL. Isso permite que o programador faça uso de outras linguagens de computação, tais como o C, C++ ou até mesmo o Fortran, que em geral são utilizadas para maximizar o desempenho computacional. É possível, inclusive, compilar o código que será executado em outra unidade de processamento, como por exemplo, a placa de vídeo.

As funções escritas em outras linguagens devem seguir um padrão estabelecido pelo IDL (IDL/EXELIS, 2014) para que sejam compatíveis. Em C++ o formato a ser empregado é:

```
RET_TYPE nome_da_funcao(int argc, void *argv[]),
```

sendo **RET_TYPE** o tipo de variável a ser retornada, **nome_da_funcao** o nome da função em C++, *argc* é o número de argumentos passados e o vetor *argv* contém o endereço ou valor desses argumentos.

Vale salientar que, no geral, essa função não é utilizada para fazer o cálculo numérico mas para atuar como uma espécie de *parser*, ou seja, fazendo a atribuição dos endereços das variáveis de interesse (que estão organizadas no vetor *argv*) para variáveis com nomes intuitivos. Por exemplo, deseja-se passar duas variáveis para um programa em C++, sendo uma das variáveis um vetor de números reais e a outra um inteiro que contém a dimensão do vetor. A função *parser* chamará uma segunda função e passará como argumentos *argv*[0] e *argv*[1], enquanto que na segunda função o

programador poderá fazer uso de variáveis com nomes amigáveis, como por exemplo, *vetor* e *tamanho*.

Como o programa deve ser executado na placa de vídeo é necessário configurar o Microsoft Visual Studio 2010 para utilizar o compilador da Nvidia, o *nvcc*. A maneira mais fácil para fazer isso é criar um novo projeto "*CUDA 6.0 Runtime*". Em seguida, é necessário configurar o tipo de projeto para "*Dynamic Library (DLL)*". Para fazer a biblioteca dinâmica (DLL) compatível com o IDL, é necessário incluir o cabeçalho *idl_export.h* e "linkar" o código com a biblioteca *idl.lib*, que se encontram no diretório de instalação do IDL.

3.2 Implementação em CUDA

As funções que são executadas na placa de vídeo são chamadas de *kernels* (NVIDIA, 2014). Cada *kernel* pode ser invocado pela CPU e, além dos parâmetros necessários, há a configuração de como as *threads* serão distribuídas. A linguagem de programação CUDA C permite organizar as *threads* em blocos (Guarita, 2011). Nesse trabalho, escolheu-se atribuir uma *thread* para cada pixel (*i*, *j*) da imagem.

A matriz de coerência [*T*] possui um polinômio característico de 3ª ordem e as suas raízes (autovalores) foram calculados com uso do método iterativo de Newton. Para iniciar a busca numérica pelas três raízes foi calculada inicialmente a derivada do polinômio característico, resultando em um polinômio 2ª ordem, que possui solução analítica simples. Um ponto de partida é escolhido como sendo um número a esquerda da menor raiz, o segundo como estando à direita da maior raiz e o último, como sendo a média das raízes. Mesmo que um polinômio de 3ª ordem possua solução analítica, seria necessário calcular raízes quadradas (ou cúbicas) várias vezes, que, por sua vez, são resolvidas empregando métodos iterativos, de forma que o cálculo analítico pode ser, em geral, mais demorado que a solução adotada.

O processador implementado em CUDA está disponível em https://bitbucket.org/colombojrj/processador-sar-cuda.

4. Cenário de observação da cena e requisitos de tempo de processamento

A geometria de aquisição dos dados brutos de um Radar de Abertura Sintética (SAR) no modo de imageamento de uma faixa de terreno (*strip mode*) é mostrada na Figura 2, sendo *h* a altitude da plataforma, v_p sua velocidade, r_o a distância do centro da cena à trajetória do sensor, ϑ_0 o ângulo de visada do centro da cena e θ_e a abertura angular da antena em elevação.

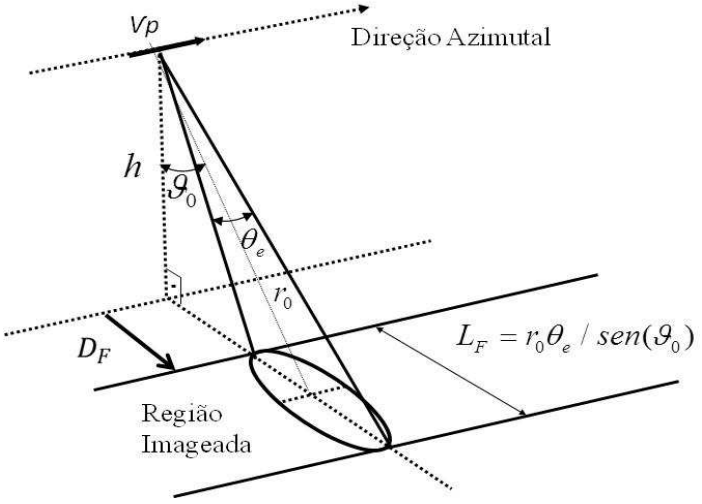


Figura 2. Geometria de iluminação do sensor SAR

O imageamento de uma cena pode ocorrer segundo vários modos de operação que definem, entre outros parâmetros, a largura da faixa imageada (L_F) e a sua distância no solo ao sensor SAR (D_F). A Tabela 1 mostra três modos de operação (M_1, M_2 e M_3) previstos no Projeto AEROSAR FINEP, Convênio 01.10.058.00, em desenvolvimento do Instituto tecnológico de Aeronáutica (ITA), que tem como interveniente a Fundação de Desenvolvimento da Pesquisa (FUNDEP). A Tabela 2 mostra os parâmetros do sensor SAR previsto neste projeto, que tem por objetivo a construção de um simulador e um modelo funcional de engenharia de um processador SAR que opere em tempo real para missões de defesa.

Tabela 1. Requisitos dos Modos de Operação.

Parâmetros		M_1	M_2	M_3
h	Altitude da plataforma	4 km	8 km	12 km
D_F	Distância da região imageada	4,5 km	18 km	72 km
L_F	Largura da região imageada	0,5 km	2 km	8 km
δ_R	Resolução em alcance	0,5 m	2 m	8 m
δ_A	Resolução azimutal	0,5 m	2 m	8 m

Tabela 2. Parâmetros do sensor SAR.

Comprimento de onda	0,03 m
Período de repetição de pulso	625 μs
Duração do pulso	667,13 ns
Frequência de amostragem	300 MHz
Squint angle	9,97º
Chirp rate	449,3 THz/s
Abertura azimutal da antena	6,88º
Velocidade da plataforma	200 m/s

Chiarrati e Fernandes (2013) calcularam para os três modos de operação o Tempo de Processamento (T_{Proc}) de uma imagem SAR e o Tempo Crítico (T_{Crit}) que é o máximo período de tempo em que deve ser processada uma imagem SAR para que a cadência de dados de saída seja a mesma da entrada e portanto não haja acumulo crescente de dados não processados, deste modo T_{Proc} deve ser tal que $T_{Proc} \leq T_{Crit}$. O T_{Proc} é também aproximadamente igual ao Tempo de Latência (T_{Lat}), diferença de tempo entre a aquisição e a imagem formada. Os resultados desse calculo são resumidos na Tabela 3.

Tabela 3. Tempos de processamento da imagem e limites.

	$T_{Proc} \approx T_{Lat}$	$T_{mcla} = T_{Crit} - T_{Proc}$
M_1 – Imagem 1024x4096 (Alcance x Azimute) $\Rightarrow T_{Cri}$ =1,60*s*		
1 bloco em alcance	76,00*ms*	**1,524*s***
3 blocos em alcance	101,50*ms*	1,498*s*
M_2 – Imagem 2048x4096 (Alcance x Azimute) $\Rightarrow T_{Cri}$ =1,75*s*		
1 bloco em alcance	163,93*ms*	1,586*s*
10 blocos em alcance	160,52*ms*	**1,589*s***
M3 – Imagem 8192x4096 (Alcance x Azimute) $\Rightarrow T_{Cri}$ =1,80s		
1 bloco em alcance	742,28*ms*	1,057*s*
9 blocos em alcance	683,35*ms*	1,116*s*
34 blocos em alcance	561,90*ms*	**1,238*s***

Na Tabela 3, para os três modos de operação, foi considerado uma imagem com 4096 pixels em azimute e um tamanho variável (1024, 2048 e 8192) de pixels em distância em conformidade com o modo considerado. Nessa Tabela também foi incluída uma coluna com o valor do tempo máximo que pode ser utilizado para a classificação

de uma imagem (T_{mcla}), dado por $T_{mcla} = T_{Crit} - T_{Proc}$, considerando-se que o processo de formação das três imagens SAR polarimétricas (*hh*, *vv* e *hv*) necessárias para a classificação polarimétrica é feito em paralelo.

Observa-se na Tabela 3 que o menor tempo de processamento da imagem SAR e como consequência o maior tempo disponível para se classificar as imagens é:

a) no modo M_1 o processamento de apenas um bloco em alcance; b) no modo M_2 o processamento de 10 blocos em alcance e c) no modo M_3 o processamento de 34 blocos em alcance. Na Tabela 3 os tempos máximos para a classificação, para cada modo, são apresentados em negrito.

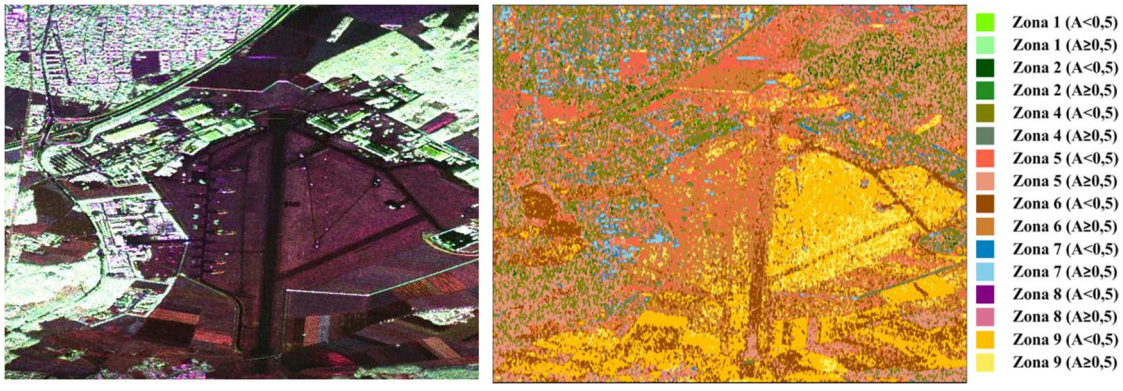
5. Resultados

Utilizou-se para teste um conjunto de imagens SAR polarimétricas (*hh*, *hv* e *vv*), de dimensões 4000x1580 pixels, do SAR aeroembarcado E-SAR do Instituto de Micro-ondas e Radar do Centro Alemão de Pesquisa Aeroespacial (DLR).

Como referência, implementou-se em IDL o processo de classificação H-α-A utilizando-se somente a CPU. Essa implementação foi executada 100 vezes, levando em média 5,22 minutos com desvio padrão de 0,0087 minutos. A maior demanda de tempo (91%) foi no cálculo dos autovalores e autovetores das matriz coerência [*T*] de cada pixel da imagem polarimétrica.

Em seguida, o classificador foi implementado para ser executado na placa de vídeo, que foi novamente executado 100 vezes, levando em média 0,224s com desvio padrão de 0,00089s. Verificou-se que a execução do processador H-α-A na GPU foi quase 1400 vezes mais rápida que a implementação usando apenas a CPU. Considerando-se imagens polarimétricas de dimensões 1024x4096, 2048x4096 e de 8192x4096, segundo os modos de operação da Tabela 3, os tempos de classificação dessas imagens seriam aproximadamente 0,149s, 0,297s e 1,189s, respectivamente. Comparando-se com os tempos disponíveis para a classificação dispostos na Tabela 3 verifica-se que todos os requisitos dos modos de operação M_1, M_2 e M_3 são atendidos plenamente (0,149s < 1,524s, 0,297s <1,589s e 1,189s < 1,238s), permitindo a operação em tempo real.

A Figura 3 mostra uma composição colorida da imagem original e a imagem classificada no plano H-α-A, na qual foi utilizada uma vizinhança de 5x5 pixels para estimativa da matriz coerência [*T*].



a) b)

Figura 1. Classificação Plano H-α-A. a) Composição colorida da imagens SAR polarimétrica (Shh + Svv→ Red, 2Shv→ Green e Shh - Svv → Blue) e b) Imagem Classificada em 16 Classes do Plano H-α-A.

Na classificação apresentada na Figura 3 foram utilizadas efetivamente 16 classes das 18 teoricamente possíveis, pois duas das classes possíveis (Zona 3 com A≥0,5 e Zona 3 com A <0,5) não são admissíveis no processo de classificação polarimétrica

(Cloude e Pottier, 2013), (Hellmann, 2000) e (Villaça, 2008). Na Figura 3 tem-se a Zona 1 com regiões de vegetação com estrutura geométrica bem resolvida, a Zona 2 representa as regiões espalhamento da vegetação com alta entropia, enquanto a Zona 4 representa regiões de vegetação em área urbana, a Zona 5 retrata as regiões de superfícies vegetadas, a Zona 6 representa pistas e/ou solo exposto, a Zona 7 representa regiões urbanas, a Zona 8 representa regiões vegetação com elementos de espalhamento anisotrópicos altamente correlacionados e, por fim, a Zona 9 contém regiões de superfícies bem suaves.

Para a classificação apresentada calculou-se ainda para regiões conhecidas de 32x32 pixels o coeficiente Kappa obtendo-se Kappa = 0,9665 para uma classificação em 8 classes e Kappa = 0,9099 para a classificação em 16 classes.

6. Comentário Finais

Neste trabalho foi implementado um classificador de imagens SAR polarimétricas para operar em tempo real, utilizando processamento distribuído com uma GPU.

Comprovou-se com os resultados obtidos que o classificador implementado atende os requisitos operacionais do Projeto FINEP AEROSAR, de forma que pode ser implementado para operar em tempo real.

Agradecimentos

Os autores agradecem o apoio da FINEP (Convênio 01.10.058.00), CNPq (Bolsas DTI) e FUNDEP (Projeto 16988) para a realização deste trabalho.

REFERÊNCIAS

Cumming, I. G.; Wong, F. H. **Digital Processing of synthetic aperture radar data: algorithms and implementations**. Norwood: Artech House, 2005. ISBN: 1-58053-058-3. 660 p.

Chiaratti,A.; Fernandes, D. Avaliação da influência de requisitos operacionais na geração de imagens SAR em tempo real. Simpósio de Aplicações Operacionais em Áreas de Defesa, XV SIGE, 2013.

Guarita, F. C.**Avaliação da Arquitetura CUDA para Síntese de Imagens SAR**. Dissertação (Mestrado em Engenharia Aeronáutica) - Instituto Tecnológico de Aeronáutica, São José dos Campos, Brasil. 2011.

Trofino, S.; d´Amore, R.; Fernandes, D. Avaliação do processador SAR Range-Doppler para operação em tempo real em um computador de uso geral. Simpósio de Aplicações Operacionais em Áreas de Defesa, XIV SIGE, 2012.

Trofino, S. H. **Implementação de um processador SAR Range-Doppler em um computador pessoal visando operação em tempo real**. 2014. 113 f. Dissertação (Mestrado em Engenharia Eletrônica e Computação) - Instituto Tecnológico de Aeronáutica, São José dos Campos. 2014.

Cloude, S. R.; Pottier, E.; An Entropy Based Classification for Land Application of Polarimetric SAR. IEEE Transactions On Geoscience and Remote Sensing, v. 35, n. 1, 68-78, 1997.

Cloude, S. R.; Pottier, E.; A Review of Target decomposition theorems in radar polarimetric. IEEE Transactions On Geoscience and Remote Sensing, v. 34, n. 2, 498-518, 1996

Hellmann, M. **Classification of Fully Polarimetric SAR Data for Cartografic Application**. Oberpaffenhofen: DLR. Tese de Doutorado – PhD, 2000.

Villaça, D. D. **Classificação de imagens SAR utilizando a resposta polarimétrica**. Dissertação (Mestrado em Engenharia Eletrônica e Computação) - Instituto Tecnológico de Aeronáutica, São José dos Campos. 2008.

CUDA C Programming Guide (NVIDIA). Disponível em: <http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>. Acesso em: 15/outubro/2014.

Documentation Center (IDL/EXELIS). Disponível em: <http://www.exelisvis.com/docs>. Acesso em: 15/outubro/2014.

Passing Parameter (IDL/EXELIS). Disponível em: <http://www.exelisvis.com/docs/PassingParameters.html>. Acesso em: 15/outubro/2014.