Uma Proposta baseada em Contêineres para Mitigar a Crise de Reprodutibilidade na Comunidade de Visão Computacional

Anderson Adaime De Borba
Dept. de Economia
IBMEC-SP
São Paulo, Brasil
anderborba@gmail.com
https://orcid.org/0000-0001-8479-9128

Luiz Barboza
Dept. de Eng. de Software)
CESAR School)
Recife, Brasil
lcbj@cesar.school

***Abstract*—**
***Palavras chaves*—**

I. Introdução

A pesquisa descrita em (1) menciona que cerca de 75% dos pesquisadores encontraram problemas ao tentar reproduzir resultados de artigos publicados em importantes revistas científicas na área de sensoriamento remoto, mais especificamente na área de imagem SAR (*Synthetic Aperture Radar*). A reportagem da Nature que pode ser vista em (2) mostra número semelhante, em torno de 70% dos pesquisadores entrevistados não conseguiram reproduzir resultados publicados em artigos científicos. Adicionalmente, podemos destacar em (2) a afirmação que mais da metade dos pesquisadores não reproduziram seus próprios resultados.

Instigado por essas evidências, decidimos verificar a reprodutibilidade dos resultados do artigo (3). Para este fim, aplicamos duas metodologias: o uso do repositório https://github.com/ no qual disponibilizamos as bases de dados e os códigos necessários para a pesquisa; O uso do aplicativo Docker (4) para gerenciamento da tecnologia conhecida como contêineres (5). E o repositório de imagens chamado de Docker hub, onde disponibilizamos a imagem do contêiner, que pode ser acessado em https://hub.docker.com/. Nessas metodologias controlamos todos os processos, desde o armazenamento dos dados até a reprodução dos resultados do artigo selecionado.

Destacamos que a ideia principal do presente artigo é reproduzir os experimentos numéricos do artigo (3). Para isso, se faz necessário a descrição do procedimento de fusão de evidências de bordas, para então descrever os processos usados para garantir a reprodutibilidade dos resultados.

As metodologias empregadas para a detecção de bordas podem ser encontradas nas referências (3) e (6), assim como as referência teóricas usadas para construir o método de fusão de evidências de bordas.

Sendo assim, o presente artigo está estruturado da seguinte forma: Na seção II descrevemos a base de dados (imagem) usada para a reprodutibilidade dos resultados. Na seção III mostramos as metodologias aplicacadas para a fusão de evidências de bordas. A estrutura do processo de reprodutibilidade está descrita na seção IV. Os resultados são mostrados na seção V. Para finalizar propomos algumas discussões na seção VI.

II. Base de dados

No presente trabalho usamos como base de dados a imagem de radar com abertura sintética polarimétrica (PolSAR) adquiridas com o sensor aerotransportado AIRSAR em uma aeronave DC-8. A aquisição das imagens é realizada com a aeronave voando a 8 kilometros da superfície terrestre com velocidade de $215\,\mathrm{m\,s^{-1}}$. Vamos descrever características específicas da imagem.

A imagem da região de Flevoland com dimensão 750×1024 píxeis é uma imagem capturada pelo sensor AIRSAR com banda-L. A imagem tem 4 visadas, 9 canais com resolução de aproximadamente $12.10\,\mathrm{m}$ na direção azimutal e $6.6\,\mathrm{m}$ na direção lateral do sensor. A decomposição de Pauli da imagem capturada no sensor é mostrada na Figura 1.

III. Metodologia

A metologia que usamos segue os seguintes procedimentos:

(i) Especificar manualmente ou automaticamente a região de interesse ROI;
(ii) Em cada ROI calcular o centro de massa e, a partir desse ponto, traçar retas radiais cruzando duas amostras distintas. O artigo (7) propõe um método para garantir que cruzamos duas amostras. Nas

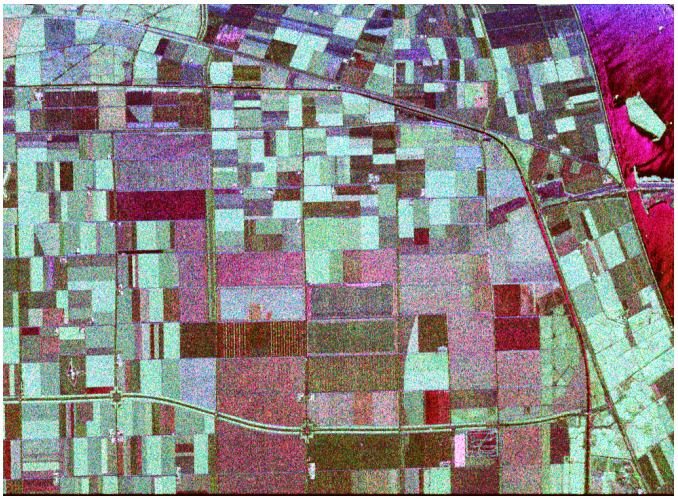


Fig. 1: Imagem da região de Flevoland

radiais serão extraídas informações para o passo (iii);

(iii) Nos dados extraídos em cada reta radial o método MLE é aplicado para estimar o ponto de transição entre duas amostras, chamado de evidência de borda;

(iv) Métodos de fusão são aplicados nas evidências de bordas obtidas em cada canal, com o objetivo de detectar as bordas na ROI escolhida.

A. Estimativa por máxima verossimilhança

A técnica de máxima verossimilhança (MLE – *Maximum Likelihood Estimation*) permite estimar os parâmetros de um modelo estatístico usando uma amostra de dados observados. Essa técnica é desejável por apresentar boas propriedades em relação a outras abordagens.

Suponha $\boldsymbol{z} = (z_1, \ldots, z_n)$ um vetor de variáveis aleatórias independentes e identicamente distribuídas segundo uma distribuição caracterizada pela função de densidade de probabilidade $f(\boldsymbol{z}, \theta)$ que, por sua vez, é indexada pelos parâmetros $\theta = (\theta_1, \ldots, \theta_d)^T$ pertencentes ao espaço paramétrico Θ. Define-se então a função de verossimilhança

$$L(\theta; \boldsymbol{z}) = \prod_{k=1}^{n} f(z_k; \theta), \tag{1}$$

e a função log-verossimilhança

$$\mathcal{L}(\theta; \boldsymbol{z}) = \ln L(\theta; \boldsymbol{z}) = \sum_{k=1}^{n} \ln f(z_k; \theta). \tag{2}$$

O vetor de parâmetros θ é estimado pelo vetor $\widehat{\theta}$ tal que $\mathcal{L}(\widehat{\theta}; \boldsymbol{z}) \geq \mathcal{L}(\theta; \boldsymbol{z})$ para todo θ no espaço dos parâmetros Θ. A estimativa de máxima verossimilhança pode ser representada por

$$\widehat{\theta} = \arg\max_{\theta \in \Theta} \mathcal{L}(\theta; \boldsymbol{z}), \tag{3}$$

em que $\boldsymbol{z}$ agora denota as observações.

B. A máxima verossimilhança usando os parâmetros estimados

Cada linha radial $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$ é particionada em duas amostras disjuntas na posição j,

$$\boldsymbol{z} = (\underbrace{z_1, z_2, \ldots, z_j}_{\boldsymbol{z}_{\mathsf{I}}}, \underbrace{z_{j+1}, z_{j+2}, \ldots, z_n}_{\boldsymbol{z}_{\mathsf{E}}}),$$

para as quais são definidos modelos estatísticos diferentes, um modelo para $\boldsymbol{Z}_{\mathsf{I}} \sim W(\Sigma_{\mathsf{I}}, L_{\mathsf{I}})$, e outro modelo para $\boldsymbol{Z}_{\mathsf{E}} \sim W(\Sigma_{\mathsf{E}}, L_{\mathsf{E}})$.

O método MLE, descrito na seção (III-A), é aplicado nas amostras internas $\boldsymbol{z}_{\mathsf{I}}$ e externas $\boldsymbol{z}_{\mathsf{E}}$ para estimar $(\Sigma_{\mathsf{I}}, L_{\mathsf{I}})$ e $(\Sigma_{\mathsf{E}}, L_{\mathsf{E}})$ maximizando (2), e obtendo $(\widehat{\Sigma}_{\mathsf{I}}, \widehat{L}_{\mathsf{I}})$ e $(\widehat{\Sigma}_{\mathsf{E}}, \widehat{L}_{\mathsf{E}})$.

A função de verossimilhança é definida no ponto j de acordo com a função

$$L(j; \boldsymbol{z}) = \prod_{k=1}^{j} f_{\mathbf{Z}}(\boldsymbol{z}_k; \boldsymbol{\Sigma}_k, L_k) \prod_{k=j+1}^{n} f_{\mathbf{Z}}(\boldsymbol{z}_k; \boldsymbol{\Sigma}_k, L_k). \tag{4}$$

Usando propriedades de logaritmos naturais para cada termo do produtório (4) é definida a função log-verossimilhança total dependendo de j:

$$\mathcal{L}(j) = \sum_{k=1}^{j} \ln f_{\mathbf{Z}}(\boldsymbol{z}_k; \boldsymbol{\Sigma}_k, L_k) + \sum_{k=j+1}^{n} \ln f_{\mathbf{Z}}(\boldsymbol{z}_k; \boldsymbol{\Sigma}_k, L_k). \tag{5}$$

O estimador de máxima verossimilhança $\widehat{\jmath}_{\mathsf{ML}}$ é uma evidência de borda, pois representa uma aproximação da transição entre regiões, sendo

$$\widehat{\jmath}_{\mathsf{ML}} \quad = \quad \arg\max_{j} \mathcal{L}(j). \tag{6}$$

É importante destacar que os parâmetros estimados em (6) são os números de visadas L (do lado interno e do lado externo), as matrizes de covariância $\boldsymbol{\Sigma}$ (do lado interno e do lado externo), e o ponto j. A evidência de borda é encontrada aplicando-se o método GenSA (8).

C. O MLE para a distribuição gama

Cada canal de intensidade pode ser modelado por uma lei gama, como apresentado em (7):

$$f_Z(z; \mu, L) = \frac{L^L}{\Gamma(L)\mu^L} z^{L-1} \exp\left\{-\frac{L}{\mu} z\right\}, \tag{7}$$

onde, $\mu > 0$ e $L > 0$. Aplicando o logaritmo natural obtemos a função,

$$\ln f_Z(z; \mu, L) = L \ln \frac{L}{\mu} - \ln \Gamma(L) + (L-1) \ln z - \frac{L}{\mu} z. \tag{8}$$

Dada a amostra $\boldsymbol{z} = (z_1, \ldots, z_n)$ extraída dos canais de intensidades hh, hv, e vv, e utilizando a função logarítmica (8), definimos a função log-verossimilhança

$$\mathcal{L}(\boldsymbol{z}; \mu, L) = \sum_{k=1}^{n} \ln f_Z(z_k; \mu, L).$$

resultando na função log-verossimilhança para a PDF univariada (7) na sua forma reduzida

$$\mathcal{L}(\boldsymbol{z};\mu,L)=n\left[L\ln\frac{L}{\mu}-\ln\Gamma(L)\right]+L\sum_{k=1}^{n}\ln z_k-\frac{L}{\mu}\sum_{k=1}^{n}z_k. \tag{9}$$

A função (9) é usada para estimar os parâmetros $(\widehat{\mu},\widehat{L})$ com o método MLE de (μ,L) baseado na amostra $\boldsymbol{z}$. No trabalho (3) fixamos o parâmetro múltiplas visadas em $\mathsf{L}=4$, e realizamos a estimativa para o parâmetro μ usando a média dos elementos da amostra considerada. Este procedimento fornece uma estimativa $(\widehat{L},\widehat{\mu})$ para $(\widehat{L},\widehat{\mu})$ que maximiza a função (9).

De acordo com a seção III-B é extraído de cada canal de intensidade da imagem PolSAR uma faixa de dados $\boldsymbol{z}=(z_1,z_2,\ldots,z_n)$ de forma que seja particionada em duas amostras, disjuntas, na posição j:

$$\boldsymbol{z}=(\underbrace{z_1,z_2,\ldots,z_j}_{\boldsymbol{z}_{\mathsf{I}}},\underbrace{z_{j+1},z_{j+2},\ldots,z_n}_{\boldsymbol{z}_{\mathsf{E}}}),$$

para as quais são definidos modelos estatísticos diferentes, um modelo para $\boldsymbol{Z}_{\mathsf{I}}\sim\Gamma(\mu_{\mathsf{I}},L_{\mathsf{I}})$, e outro modelo para $\boldsymbol{Z}_{\mathsf{E}}\sim\Gamma(\mu_{\mathsf{E}},L_{\mathsf{E}})$.

As funções log-verossimilhança reduzidas aplicadas nas amostras internas $\boldsymbol{z}_{\mathsf{I}}$ e externas $\boldsymbol{z}_{\mathsf{E}}$ são usadas para estimar os parâmetros $(\mu_{\mathsf{I}},L_{\mathsf{I}})$ e $(\mu_{\mathsf{E}},L_{\mathsf{E}})$ maximizando (9), e obtendo $(\widehat{\mu}_{\mathsf{I}},\widehat{L}_{\mathsf{I}})$ e $(\widehat{\mu}_{\mathsf{E}},\widehat{L}_{\mathsf{E}})$.

A log-verossimilhança total é definida no ponto j pela seguinte função

$$\begin{aligned}\mathcal{L}(j;\widehat{\mu}_I,\widehat{L}_I,\widehat{\mu}_E,\widehat{L}_E)=&-\left(\frac{\widehat{L}_{\mathsf{I}}}{\widehat{\mu}_{\mathsf{I}}}\sum_{k=1}^{j}z_k+\frac{\widehat{L}_{\mathsf{E}}}{\widehat{\mu}_{\mathsf{E}}}\sum_{k=j+1}^{n}z_k\right)+\\ &j\left[\widehat{L}_{\mathsf{I}}\ln(\widehat{L}_{\mathsf{I}}/\widehat{\mu}_{\mathsf{I}})-\ln\Gamma(\widehat{L}_{\mathsf{I}})\right]+\widehat{L}_{\mathsf{I}}\sum_{k=1}^{j}\ln z_k+\\ &(n-j)\left[\widehat{L}_{\mathsf{E}}\ln(\widehat{L}_{\mathsf{E}}/\widehat{\mu}_{\mathsf{E}})-\ln\Gamma(\widehat{L}_{\mathsf{E}})\right]+\widehat{L}_{\mathsf{E}}\sum_{k=j+1}^{n}\ln z_k.\end{aligned} \tag{10}$$

e aplicando o método GenSA (8) encontramos a evidência de borda

$$\widehat{\jmath}=\arg\max_{j\in[\min_s,N-\min_s]}\ell(j;\widehat{\mu}_I,\widehat{L}_I,\widehat{\mu}_E,\widehat{L}_E),$$

D. Métodos de fusão de informação para as evidências de bordas

As evidências de bordas estão armazenadas em n_c imagens binárias $\{\widehat{\boldsymbol{\jmath}}_c\}_{1\le c\le n_c}$, onde o pixel de valor 1 denota uma estimativa de borda e o pixel de valor 0 denota um elemento onde não foi detectada borda. As imagens (matrizes) têm tamanho $m\times n$, e denotamos $\ell=mn$. Essas imagens foram usadas para a fusão resultando na imagem I_{F} com as bordas detectadas.

No trabalho selecionado para usar as reprodutibilidades foram propostas 4 técnicas de fusão de evidências de bordas:

- Média simples;
- Transformada wavelet estacionária multi-resolução (MR-SWT);
- Análise de componentes principais (PCA);
- Estatísticas ROC (E-ROC);

O fluxograma do processo de fusão pode ser observada na figura 2,

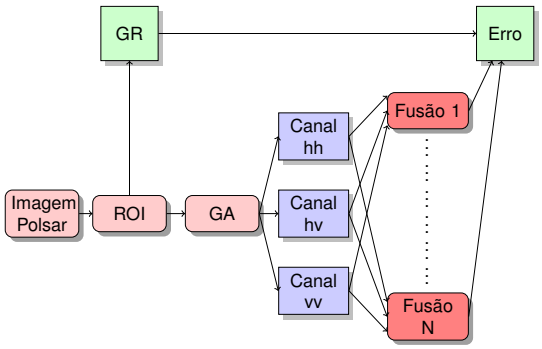


Fig. 2: Fluxograma do processo de fusão de evidências

IV. Estrutura do processo de reprodutibilidade

No trabalho (3) foi utilizado como linguagem de programação o matlab (9) e a linguagem de programação R (10). Neste trabalho fizemos a tradução dos códigos para a linguagem de programação Python (11). A escolha por utilizar outra linguagem de programação não foi em detrimento a nenhuma das linguagem usadas anteriormente, trata-se somente de uma decisão de projeto, pois acreditamos que a reprodutibilidade não depende da linguagem de programação empregada.

A. Usando o repositório Github

Os programas e a base de dados utilizados estão armazenados no repositório Github, no endereço eletrônico https://github.com/anderborba/**********. Para obter os resultados pode-se fazer o *download* do projeto e rodar os programas com o comando do python `python3.7 CodigoAndersonCompleto_tengarss.py ./Data/AirSAR_Flevoland_Enxuto.mat`. Quando executamos o comando, o programa está apto a ler a base de dados no diretório `/Data`, e gravar os resultados no diretório `/figuras` no formato pdf. Qualquer informação sobre os códigos pode ser encontrada no projeto, incluindo as orientações para execução e reprodução. O que pode ser um ponto de falha, pode depender de execução humana.

B. Usando a tecnologia Docker

Docker é um software de código aberto executado nos sistemas operacionais Linux, Windows e Mac OS, responsável pelo empacotamento dos nossos códigos e suas dependências em contêineres.

Para entender o funcionamento do software Docker, explicaremos o que são contêineres e como são usados.

Os contêineres são tecnologias que permitem empacotar e isolar aplicações juntamente com todo o seu ambiente de execução. Isto facilita a movimentação e portabilidade da aplicação para diferentes sistemas operacionais e equipamentos, mantendo a funcionalidade total da aplicação. Portanto, com o uso dos contêineres podemos garantir que nossos códigos sejam confiáveis, escaláveis e portáveis.

A primeira versão foi lançada em 2013. Veja (4).

O Docker pode ajudar nos seguintes problemas:

(i) Conflito de dependências – Se existe a necessidade de executar programas com duas versões de um mesmo software podemos empacotar cada versão em um contêiner, por exemplo, temos que executar programas em duas versões diferentes do software OpenCV (12), podemos instalar cada versão em um contêiner diferente.

(ii) Dependências ausentes – As dependências necessárias para seu software são empacotadas conjuntamente nos contêineres.

(iii) Diferentes plataformas – Mover o seu software de um SO para outro SO pode ser um problema. O Docker resolve o problema executando o contêiner nesse diferentes sistemas operacionais.

Muitos desenvolvedores de software usam o Docker para simplificar o processo de construção, execução e gerenciamento de seus software.

Podemos considerar que o Docker virtualiza um sistema operacional para usar o hardware do computador onde os contêineres serão executados. Para cada contêiner não é necessário ter um sistema operacional, veja 3, diferença que podemos observar em relação VMs. As VMs necessitam de seu próprio sistema operacional, veja 4. Assim, a principal diferença está no fato dos contêineres compartilharem o sistema operacional.

O fluxograma do sistema de contêineres pode ser visto na figura 3 e o fluxograma das VMs está na 4. Nas figuras realizamos as seguintes abreviações: APP significam os aplicativos necessários, no nosso caso são os programas em Python que criamos para o processo de detecção de evidências de bordas e fusão das informações destas evidências. Enquanto Bins são os binários e Bibls são as bibliotecas que usamos.

C. Obter a imagem do contêiner e reproduzir os resultados

Nesta seção, vamos descrever como podemos obter e reproduzir os resultados do trabalho (3). O pré-requisito comum para os sistemas operacionais Linux, Mac OS, e windows é a instalação do software Docker.

Com o Docker instalado nos respectivos sistema operacionais podemos seguir o seguinte procedimento:

(i) `docker pull quantpolsar/polsar_fusion`. Comando para baixar a imagem Docker que está no repositório https://hub.docker.com/.

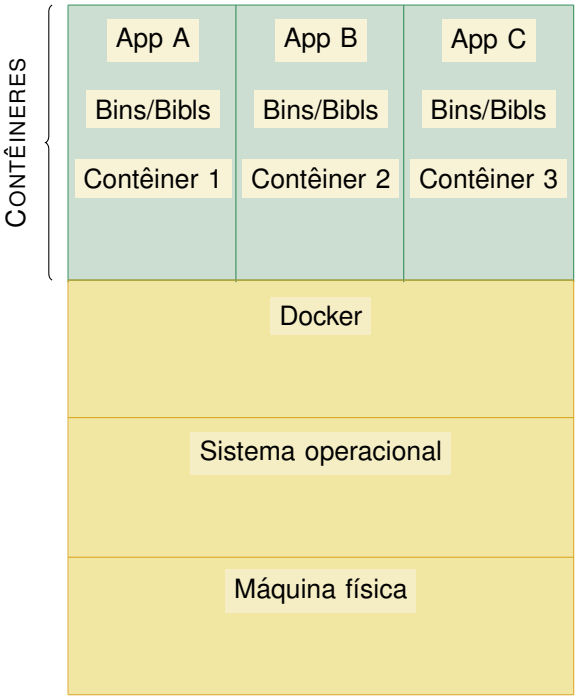


Fig. 3: Fluxograma dos Contêineres

(ii) `docker run -e DATA=./Data/AirSAR_Flevoland_Enxuto.mat -p 8080:80 quantpolsar/polsar_fusion`. Comando para rodar o Docker com o parâmetros `DATA` definindo sua base de dados, ou seja, a imagem PolSAR. Bem como o parâmetro `-p 8080:80` que associa uma porta física do SO hospedeiro a porta virtual na qual a aplicação está rodando dentro do contêiner.

(iii) Após o processamento dos canais de intensidade, as imagens com as bordas detectadas são disponibilizadas em um servidor web no contêiner. Elas podem ser acessadas através de um navegador de internet digitando o seguinte endereço `http://localhost:8080/imagens/`.

(iv) `docker ps`. Comando do Docker para a obtenção do número do processo.

(v) `docker kill número do processo`. Comando para encerrar todo o processo.

Além do acesso às imagens com bordas detectadas via servidor web, é possível ter acesso a todos os códigos e base de dados, executando o container de maneira interativa (parâmetro `-it`) `docker run -e DATA=./Data/AirSAR_Flevoland_Enxuto.mat -p 8080:80 -it --entrypoint=/bin/bash quantpolsar/polsar_fusion` alternativamente ao passo descrito no item (ii). Esse processo deve ser executado quando encerramos o processo do item (ii) . Com isso, você estará dentro do contêiner e poderá encontrar os códigos em python na raiz do SO, encontrar a base de dados, e as figuras geradas,

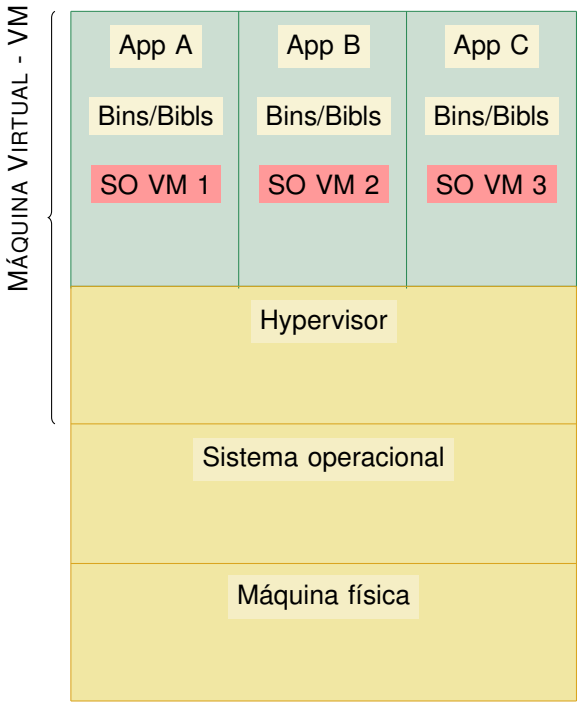


Fig. 4: Fluxograma das máquinas virtuais

respectivamente nos caminhos `/Data` `/figuras`. Adicionalmente, observamos que o passo no item (ii) vai congelar o terminal com o endereço que deve ser acessado para analisar os resultados nas figuras com as imagens PolSAR. Para descongelar o processo é basta abrir outro terminal, e executar os passos dos itens (iv) e (v).

D. A imagem Docker

A imagem Docker é construída a partir de um arquivo de definição legível por humanos e por máquinas, chamado Dockerfile.

A lista 1 mostra a estrutura do Dockerfile

```
1  FROM ubuntu:18.04
2  COPY . /
3  RUN apt-get update -y && apt-get install -y python3-
       dev python3-pip
4  RUN apt-get install -y nginx
5  RUN ln -s /figuras /var/www/html/imagens
6  RUN echo "<meta http-equiv='refresh' content='0;URL=
       http://localhost:8080/imagens/'/>" > /var/www/
       html/index.html
7  RUN pip3 install scikit-image
8  ENV DATA ./Data/AirSAR_Flevoland_Enxuto.mat
9  RUN python3 main.py $DATA
10 EXPOSE 80
11 CMD echo "Acesse http://localhost:8080" && nginx -g
       'daemon off;'
```

Listing 1: Estrutura do Dockerfile

Vamos fazer uma breve descrição para cada linha do arquivo Dockerfile da lista 1:

1) Define o SO.
2) Copia todos os arquivos para o projeto Docker.
3) Instala a linguagem de programação python.
4) Instala um servidor web para visualizar as figuras.
5) Realiza um link para o paras as imagens geradas.
6) Cria uma página em html para visualizar as figuras.
7) Instala as bibliotecas python usandas no programas.
8) Define a base de dados.
9) Comando em python que roda os programas.
10) Define uma porta interna para o contêiner.
11) Gera uma mensagem indicando como acessar as figuras e habilita um servidor web para mostrar as mesmas.

A figura 5 mostra a estrutura do contêiner que usamos para reproduzir os resultados do artifo (3). Com o Dockerfile construído, rodamos o

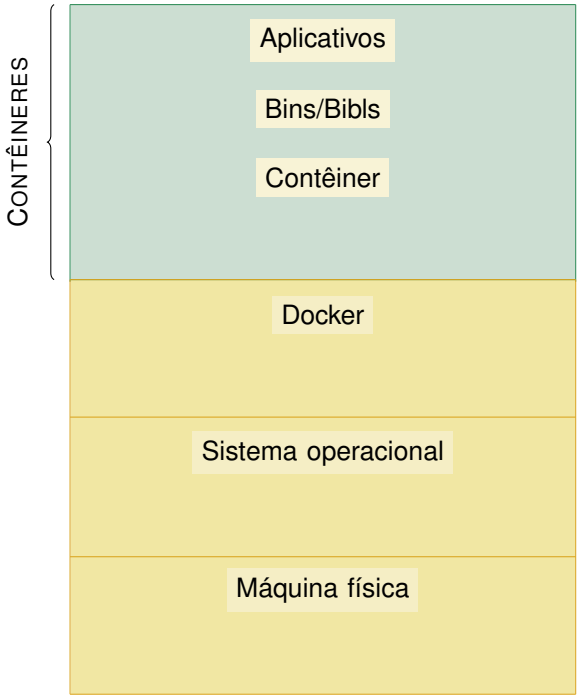


Fig. 5: Fluxograma usado no Docker

seguintes comandos do Docker para gerar a imagem. Primeiro, usamos o comando `docker build . -t quantpolsar/polsar_fusion` para gerar a imagem Docker, depois, usamos `docker push quantpolsar/polsar_fusion` para armazenar a imagem no repositório https://hub.docker.com/.

V. RESULTADOS

Os métodos apresentados no artigo (3) foram reproduzidos e executados com um computador Intel© Core i7-9750HQ CPU $2.6\,\text{GiHz}$ $16\,\text{GiB}$.

Esse métodos tomaram como referência a região de interesse (FLEV-ROI) destacada na Figura 6. A figura também mostra as radiais usadas para extrair informações da imagens.

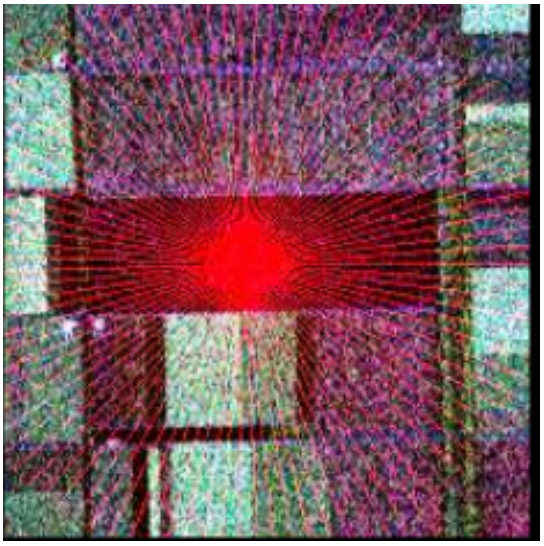
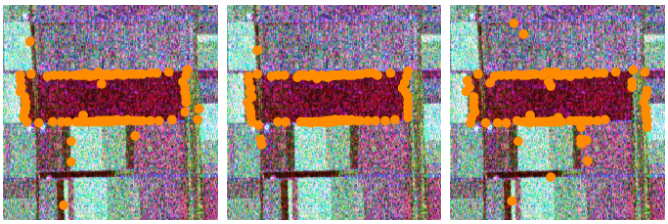


Fig. 6: Região de interesse da imagem Flevoland

As Figuras 7(a), 7(b), e 7(c) mostram, respectivamente, as evidências de bordas nos canais hh, hv, e vv, obtidas pelo método MLE. Para a FLEV-ROI foi estabelecido 100 radiais com comprimento de 120 píxeis, onde foi constatado uma forte oscilação da função de máxima verossimilhança total nos píxeis dos extremos das radiais, para evitar a oscilação nesta região foi definida uma folga com 14 píxeis. Esse valor foi escolhido empiricamente e pode variar de acordo com a região de interesse da imagem, o canal, o sensor e a imagem. Na FLEV-ROI os 14 píxeis, escolhidos para cada extremidade, foram suficientes para contornar o problema da oscilação.

Os parâmetros para as funções de máxima verossimilhança reduzidas são $L = 4$ fixo, e μ é estimado usando a médias dos elementos da amostra. Estes parâmetros são aplicados nas funções de máxima verossimilhança total com o objetivo de encontrarmos o seu valor máximo e o seu argumento correspondente. O método GenSA foi usado para realizar este processo de maximização, obtendo as evidências de bordas com precisão.

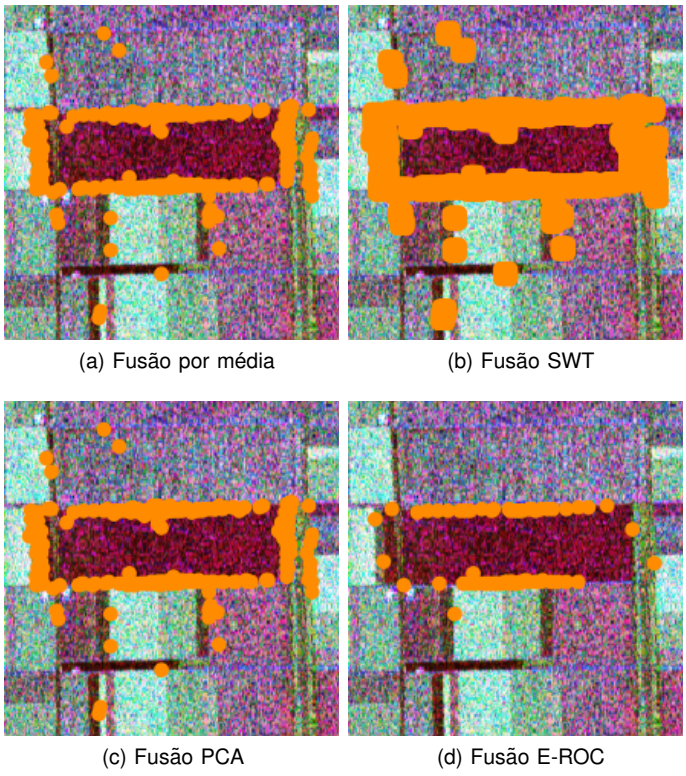


(a) Canal hh (b) Canal hv (c) Canal vv

Fig. 7: Evidências de bordas para os três canais de intensidades na FLEV-ROI

A inspeção visual destas figuras mostra a melhor acurácia dos métodos no canal hv, e presença de *outliers* no canal vv.

Figuras 8(a), (b), (c), e (d) mostram os resultados numéricos para os métodos propostos de fusão de evidência de bordas.



(a) Fusão por média (b) Fusão SWT

(c) Fusão PCA (d) Fusão E-ROC

Fig. 8: Métodos de fusão para a região FLEV-ROI

Os métodos de fusão por média e fusão PCA produzem resultados similares. A vantagem do método PCA está em realizar a média ponderada das evidências de bordas nos diferentes canais, possibilitando a quantificação da importância de cada canal no processo de fusão.

O MR-SVD produz uma considerável vantagem em descartar outliers, porém o tempo de processamento é maior em comparação com os demais métodos.

O método usando a estatística ROC produz bordas acuradas, com poucos outliers, porém de forma esparsa. Acreditamos que este método tem potencial na medida em que mais canais forem considerados, ou ao serem aplicadas outras funções de densidades de probabilidades para obter evidências de bordas.

O método baseado em *wavelets* produz densas bordas, mesmo com a inpeção visual mostrarem melhores em possibilidades de detectar bordas, o método produz outliers. Destacamos que a detecção pode ser melhorada com o uso de pós-processamento (13), podendo ser aplicado em todos os métodos, inclusive em cada canal onde as evidências forem detectadas.

VI. Discussão

Os dois procedimentos utilizados alcançaram o objetivo de reproduzir os resultados da detecção de bordas. Portanto, destacamos a importância do uso desses procedimentos no desenvolvimento de novas pesquisas.

O procedimento de usar o Github como repositório da base de dados e dos códigos é uma boa prática para alcançar a reprodutibilidade dos resultados. Porém, a responsabilidade de execução dos processos serem centradas no usuário, podem acarretar problemas na adequação de compiladores e suas bibliotecas.

A utilização dos contêineres mostraram-se confiáveis e portáveis, pois, cada contêiner possui a estrutura adequada para rodar os programas. Lembrando que definimos no Dockerfile toda a estrutura que precisamos, o SO, os códigos e o servidor web para visualizar os arquivos no formato PDF (Portable Document Format).

A portabilidade foi testada usando o procedimento dos contêineres com o auxílio do Docker nos principais sistemas operacionais, no Linux, MacOS, e Windows e em computadores com diferentes configurações de hardware. Conseguimos reproduzir exatamente os mesmos resultados do artigo selecionada.

Na criação da imagem Docker usamos somente softwares livres e de fácil acesso ao usuário.

A facilidade de obter o Docker para diferentes sistemas operacionais e as vantagens citados no parágrafo anterior mostram que o uso desta tecnologia é muito conveniente. Com o desenvolvimento dessa tecnologia teremos plataformas melhores e eficientes para gerenciar os contêineres.

Finalizando, citamos a confiabilidade, eficiência, e portabilidade como fatores para ampliar o uso da tecnologia dos contêineres.

Referências

[1] T. Balz and F. Rocca, "Reproducibility and replicability in sar remote sensing," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 3834–3843, 2020.

[2] M. Baker, "Reproducibility crisis," *Nature*, vol. 533, no. 26, pp. 353–66, 2016.

[3] A. A. De Borba, M. Marengoni, and A. C. Frery, "Fusion of evidences for edge detection in PolSAR images," in *2019 IEEE Recent Advances in Geoscience and Remote Sensing: Technologies, Standards and Applications (TENGARSS)*, Oct 2019, pp. 80–85.

[4] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, Mar. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2600239.2600241

[5] LXD, "Container and virtualization tool," 2021, acessado em 13/10/2021. [Online]. Available: https://linuxcontainers.org/

[6] "Fusão de evidências de bordas dos canais de intensidades de imagens de radar polarimétrico de abertura sintética," Ph.D. dissertation, 2020, escola de Engenharia Mackenzie (EE). [Online]. Available: http://tede.mackenzie.br/jspui/handle/tede/4716

[7] J. Gambini, M. Mejail, J. Jacobo-Berlles, and A. C. Frery, "Feature extraction in speckled imagery using dynamic B-spline deformable contours under the G0 model," *International Journal of Remote Sensing*, vol. 27, no. 22, pp. 5037–5059, 2006.

[8] Y. Xiang, S. Gubian, B. Suomela, and J. Hoeng, "Generalized Simulated Annealing for Global Optimization: The GenSA Package," *The R Journal*, vol. 5, no. 1, pp. 13–28, 2013.

[9] MATLAB, *version 8.3.0.532 (R2014a)*. Natick, Massachusetts: The MathWorks Inc., 2014.

[10] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2020. [Online]. Available: https://www.R-project.org/

[11] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[12] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[13] A. C. Frery, J. Jacobo-Berlles, J. Gambini, and M. Mejail, "Polarimetric SAR image segmentation with B-Splines and a new statistical model," *Multidimensional Systems and Signal Processing*, vol. 21, pp. 319–342, 2010.