VI Encontro Acadêmico de Computação - UFMA
 ISSN: 2594-486X

Dimmer: Um *Framework* para o Processamento de Imagens Oftalmológicas

André Felipe Carvalho Pinheiro [1], **João Dallyson Sousa de Almeida** [1], **Anselmo Cardoso Paiva** [1]

[1]Universidade Federal do Maranhão (UFMA)
Caixa Postal 65.085–580 – São Luís – MA – Brasil
andre.carvalhop94@gmail.com, jdallyson@gmail.com, anselmo.c.paiva@gmail.com

Abstract. *The use of CAD systems has grown all over the world driven by the development of medical image research. Aiming to make the development of applications for the processing of medical images be more easy, this paper shows a model of a framework with strategies of segmentation, localization of region and features extraction. The framework was tested with the implementation of classes for the segmentation and localization of optic disc in fundus image.*

Resumo. *O uso de sistemas CAD tem crescido ao redor do mundo impulsionado pelo desenvolvimento de pesquisas na área de imagens médicas. Almejando facilitar o desenvolvimento de aplicações para o processamento de imagens médicas este trabalho apresenta a modelagem de um framework com estratégias de segmentação, localização de regiões e extração de características. O framework foi testado com a implementação de classes para segmentação e localização do disco óptico em retinografias.*

1. Introdução

Sistemas de diagnóstico auxiliado por computador (CAD) são sistemas computacionais com a finalidade de auxiliar na tomada de decisão a respeito de um diagnóstico [Queiroz and Gomes 2006], diversos grupos ao redor do mundo e no Brasil tem direcionado seus estudos para o desenvolvimento de tais sistemas, alguns dos quais tratam de imagens de olho, como retinografia e tomografia de coerência óptica (OCT).

Dentre as principais funcionalidades dos sistemas CAD estão tarefas como segmentação, localização de regiões e extração de características. Diversas vezes pesquisadores são obrigados a reimplementar diversos desses algoritmos, bem como rotinas simples como processar diversas imagens de uma única vez ou a utilização de métricas para validação de seus resultados obtidos.

Visando tal problemática, segundo [Fayad et al. 1999] os *frameworks* proporcionam um design reutilizável de software sobre o qual é possível construir novas aplicações dentro de um domínio, poupando incontáveis horas de projeto. Inspirado pelo trabalho de [Sousa et al. 2005] que descreve um *framework* para visualização de dados volumétricos médicos, este trabalho tem por objetivo apresentar uma modelagem de *framework*. Seu uso pode auxiliar pesquisadores no desenvolvimento de aplicações na área de processamento de imagens médicas, propiciando uma gama de algoritmos e estrutura reutilizável para teste e solução de funcionalidades de sistemas CAD.

2. Padrões Utilizados e Modelagem

O reuso de código é um dos principais objetivos deste *framework*. Através de uma robusta modelagem o usuário pode desenvolver seu próprio método para o problema que deseja resolver, utilizando partes de soluções já desenvolvidas. Além de propiciar uma fácil comparação de seus resultados com outras metodologias que já tenham sido desenvolvidas. Este último caso pode ser bastante recorrente quando deseja-se aplicar alguma metodologia em uma base de dados diferente da que o autor utilizou em seus testes e validação, desde que essa metodologia esteja incorporada ao *framework*, poupando o usuário de reimplementa-la.

Grande parte das aplicações que podem ser construídas através deste *framework* podem fazer uso do OpenCV: uma biblioteca de código aberto para visão computacional e aprendizado de máquina, possui mais de 2500 algoritmos implementados [Bradski and Kaehler 2008]. Optou-se por aplicar o padrão de projeto *Façade* a fim de construir uma fachada simplificada para acesso a funções, classes e evitar incompatibilidade entre o uso de versões do OpenCV pelo usuário e o *framework*. A modelagem da fachada é descrita pela Figura 1.

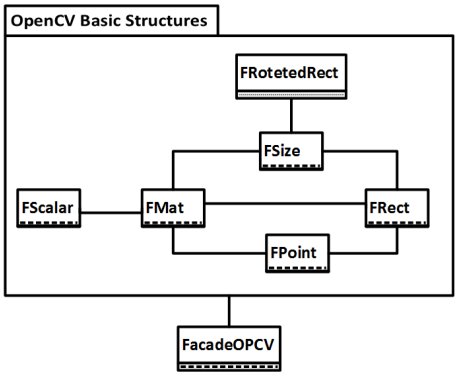

Figura 1. Modelagem do padrão *Façade* aplicado ao problema proposto.

No decorrer do estudo do problema identificaram-se três grandes classes de problemas a serem solucionados, sendo: localização da região do objeto de interesse (ROI), segmentação e extração de características.

Para cada classe desses problemas há diversas soluções disponíveis e milhares de outras que podem ser implementadas. Pensando nisso utilizou-se o padrão de projeto *Strategy*. A partir daí pode-se definir uma família de algoritmos, encapsula-los, intercambia-los e permitindo que o algoritmo varie independente do cliente que o utilize [Gamma 1995]. Sendo assim, independente da estratégia que seja implementada para segmentação, localização de ROI ou extração de características, as classes que utilizam essas estratégias não necessitam de alteração.

Para tanto utilizam-se três tipos de classe: uma de Contexto, a qual o cliente deve ter uma referência para poder utilizar uma das estratégias, uma interface de comportamento contendo a assinatura do método responsável por implementar a estratégia e por fim as classes concretas que implementam o comportamento. Pode-se então modelar a solução para as três classes de problema de acordo com a Figura 2. É importante destacar que para cada problema citado, podem haver diversas estratégias de solução (classe

Copyright (c) EAComp. ISSN: 2594-486X

concreta).

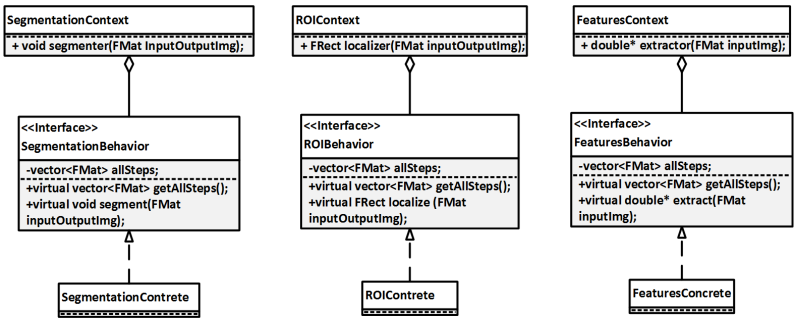

Figura 2. Modelagem do padrão *Strategy* aplicado ao problema proposto.

Empiricamente é possível notar que em todo o processo de teste de uma metodologia para o processamento de um grande conjunto de imagens, a única etapa que varia é a definição da ordem das técnicas que devem ser utilizadas. Para extração de características é provável que primeiro deseje-se localizar uma ROI ou segmentar alguma estrutura.

Para esse processo, etapas que vão desde a abertura dos arquivos até a comparação de resultados com *groud truth* são invariantes. Pensando nisso aplicou-se o padrão de projeto *Template Method* a fim de que o usuário do *framework* crie uma classe que implemente apenas um método da interface *Image Base*. Este contém um vetor de referências à interface *Processing* da qual todas as estratégias de segmentação, localização de ROI e extração de características herdam.

Dessa maneira o usuário pode definir com facilidade qual das estratégias disponibilizadas pelo *framework* ele poderá utilizar ou ainda se deseja implementar sua própria estratégia. Para este último caso deve-se criar uma classe que herde de *Processing* e implemente o método *run*. Como indicado na Figura 3, ao instanciar uma classe que herde a *Image Base* o usuário deve definir o método *processImageTemplate*. Este é responsável por coordenar a ordem das chamadas de cada *Processing* e outras operações adicionais que deseje-se fazer, como salvar cada imagem resultante do processo em um diretório ou comparar métricas de resultado.

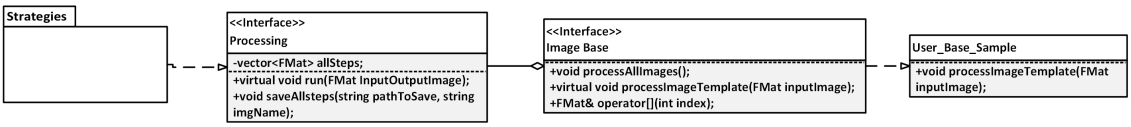

Figura 3. Modelagem do padrão *Template Method* aplicado ao problema proposto.

3. Resultados e Discussão

Uma maneira de avaliar a qualidade de um *framework* é mensurar sua capacidade de reutilização de código para que o usuário possa se concentrar em questões mais críticas do desenvolvimento. Pensando nisso os testes iniciais foram baseados na implementação de duas aplicações em imagens de fundo de olho, uma para localização da ROI e outra para a segmentação do disco óptico [Pinheiro et al. 2015].

Copyright (c) EAComp. ISSN: 2594-486X

A Figura 4 demonstra sua modelagem obedecendo as regras do paradigma de orientação a objetos propostos pelo padrão *Strategy*. Observa-se a facilidade de extensão do *framework* para quaisquer tipo de estratégia que pretenda-se implementar além de oferecer ao desenvolvedor uma gama de recursos para criação de sua aplicação. Um uso viável seria a construção de uma metodologia para segmentação de exsudatos eliminando o disco óptico como falso positivo [Zubair et al. 2016].

A priori o *framework* deve trabalhar apenas com imagens oftalmológicas, mas devido a generalidade de suas classes será possível utilizá-lo para processar outros tipos de imagens, o que pretende-se tornar objeto de estudo em trabalhos futuros.

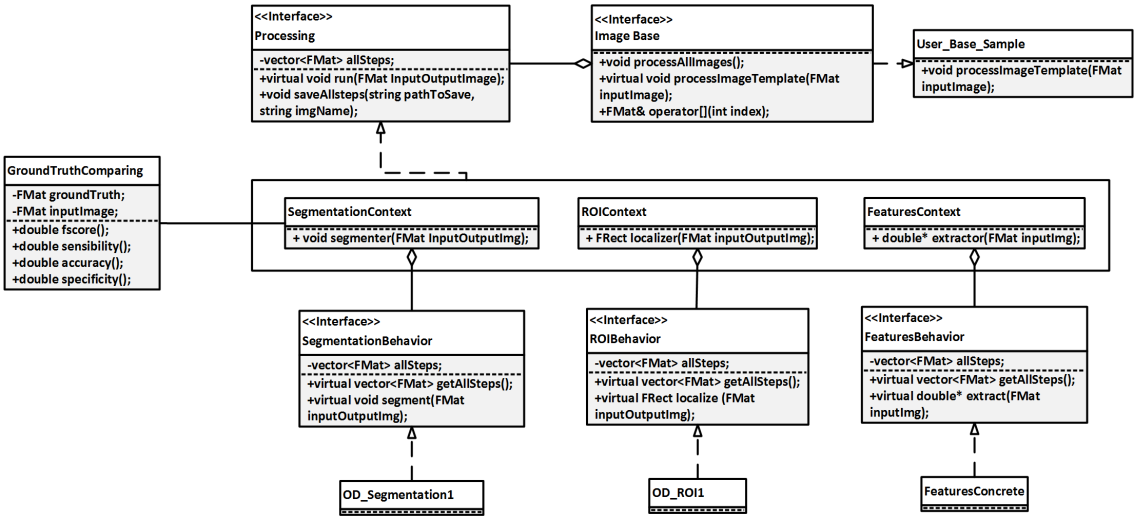

Figura 4. Diagrama de classes de testes realizados.

4. Considerações Finais

A utilização deste *framework* torna oportuna a economia de tempo dos desenvolvedores, fornecendo facilidades para o desenvolvimento de soluções através de imagens médicas. Como forma de trabalhos futuros, pretende-se agregar mais formas de estratégias para solução dos problemas propostos como segmentação da cavidade do disco óptico para auxílio do diagnóstico do glaucoma, segmentação de exsudatos, extração de características, dentre outros.

Além de fazer-se necessário o teste do *framework* por mais usuários para que avaliem suas funcionalidades, encontrem *bugs* e sugiram novas estratégias de algoritmos para serem agregadas visto que o seu uso obriga o programador a seguir seus padrões corroborando para sua utilização em um ambiente real e de grande valia para a comunidade científica fornecendo uma gama de soluções.

Referências

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc.

Fayad, M., Schmidt, D. C., and Johnson, R. E. (1999). *Building application frameworks: object-oriented foundations of framework design*.

 ISSN: 2594-486X

Gamma, E. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Education India.

Pinheiro, A. F. C., Almeida, J. D. S., Junior, G. B., and Silva, A. C. (2015). Metodologia computacional para detecção automática do glaucoma em imagens de fundo de olho. *XV Workshop de Informática Médica*.

Queiroz, J. E. R. and Gomes, H. M. (2006). Introdução ao processamento digital de imagens. *RITA*, 13(2):11–42.

Sousa, J. R. F. S., Silva, A. C., and de Paiva, A. C. (2005). Framework para representação, manipulação e visualização de dados volumétricos médicos. *V Workshop de Informática Médica*.

Zubair, M., Ali, H., and Javed, M. Y. (2016). Automated segmentation of hard exudates using dynamic thresholding to detect diabetic retinopathy in retinal photographs.