Framework de apresentação de marcações e imagens de alta resolução na interface gráfica da estação fotogramétrica digital livre E-Foto

Sarina Lustosa da Costa Santos [1,2]
Guilherme Lucio Abelha Mota [1]
João Araújo Ribeiro [1]
Ricardo Cordeiro de Farias [2]

[1] Universidade do Estado do Rio de Janeiro – UERJ
Rua São Francisco Xavier 524 - 20550-900 - Rio de Janeiro - RJ, Brasil
{sarinalustosa, guimota, joao.araujo}@gmail.com

[2] Universidade Federal do Rio de Janeiro – UFRJ
Avenida Horácio Macedo, 2030, Bloco H – 21941-914 – Rio de Janeiro – RJ, Brasil
rfarias@gmail.com

Abstract. The E-Foto project aims to develop an open source digital photogrammetric workstation (DPW). That is one of the most important open source endeavors in this field. During the recent reengineering process of E-Foto DPW, it has arisen the demand for an alternative infrastructure for viewing high resolution images and the respective symbolic layer. It should be mentioned that, in the context of E-Foto, such symbolic layer gives support to the presentation of control, verification and photogrammetric points, as well as of fiducial marks and cartographic features. The main reason for such demand is the slowness of the Qt computer graphics framework. That inconvenience took place after version 4 of Qt, whose computer graphics implementation is based on Flash, which is proprietary software and has a poor performance on every platform. The alternative framework presented herein makes use of the graphic APIs Open Graphics Library (OpenGL) and Simple Directmedia Layer (SDL). While, on one hand, OpenGL gives support to hardware accelerated computer graphics, on the other hand, in the E-Foto DPW scope, SDL underlies the digital image processing activities. This development branch aims to build up a framework for viewing high resolution images and their respective marks for the E-Foto DPW. This framework is implemented in C++ programming language and the above referenced graphical libraries, overtaking the previously mentioned performance limitations. The experimental results has shown the success of the new framework for viewing images and their respective symbolic layer.

Palavras-chave: digital photogrammery, computer graphics, fotogrametria digital, computação gráfica.

1. Introdução

Apesar dos grandes avanços no desenvolvimento de software livre dentro do campo das geotecnologias, deve-se reconhecer que a fotogrametria digital é uma área do conhecimento pouco explorada. Neste campo da ciência, uma das iniciativas mais representativas corresponde ao projeto E-Foto, que reúne estudantes, profissionais e docentes de computação e cartografia, visando à implementação de uma Estação Fotogramétrica Digital (EFD) livre distribuída sob licença GNU/GPL.

Atualmente, a EFD E-Foto está passando por um processo de reengenharia. Lançada em 2007, a versão 0.0.8.1 deste software foi a última a atingir o status de estável[1]. Tal versão é composta de sete módulos educacionais utilizados com sucesso em atividades de ensino – disciplinas regulares de graduação e pós-graduação, cursos de extensão e treinamentos em fotogrametria digital. Os módulos da versão 0.0.8 são: Retificação de Imagens, Orientação Interior, Orientação Exterior (ressecção espacial), Fototriangulação, Normalização, MNE e Orto-retificação, além da restituição fotogramétrica 3D.

Apesar de sua reconhecida contribuição para as atividades de ensino de fotogrametria digital, os módulos da versão 0.0.8.1 são isolados do ponto de vista lógico, não havendo trocas de dados nem resultados entre eles. Portanto, não há suporte ao fluxo de trabalho

1 A versão 1.0 da EFD E-Foto foi lançada no dia 08/11/2010 sendo a primeira versão lançada após o início do processo de reengenharia.

fotogramétrico. Além disso, somente é possível realizar procedimentos de fotogrametria com as imagens, referentes a um voo sobre o Campus da Uerj situado no bairro do Maracanã, que são disponibilizadas junto ao software. Considerando os aspectos não funcionais, nota-se outro problema, a excessiva dependência do *framework* da IDE Qt (Blanchette et al., 2006), em porções do código fonte da EFD onde isso poderia e deveria ter sido evitado. Este problema impôs sérias dificuldades durante a migração do código fonte do Qt 3 para o Qt 4 que produziu a versão 0.1 da EFD E-Foto. Tal procedimento tomou muito mais tempo da equipe de desenvolvimento do que esperado e, mesmo assim, acabou por produzir uma versão sem a qualidade e estabilidade desejadas.

Diante destas condições, a equipe de desenvolvimento decidiu dar início a um profundo processo de reengenharia da EFD E-Foto, cujos objetivos principais são: (1) estabelecer o compartilhamento de dados e resultados pelos módulos da EFD; (2) possibilitar a utilização de quaisquer imagens; (3) limitar a dependência do código fonte das bibliotecas de classes do Qt.

O presente artigo se insere no contexto das iniciativas que visam a busca de alternativas às bibliotecas do Qt a fim de restringir seu uso no código fonte e, consequentemente, a dependência da EFD E-Foto. Mais precisamente, este documento se dedica ao *framework* de apresentação de marcações e imagens de alta resolução na interface gráfica da EFD livre. Neste âmbito, o termo marcações ser refere a uma camada superposta às imagens de alta resolução, onde estão traçados, seguindo a padronização de simbologia, os pontos (controle, teste, fotogramétricos etc) e as feições cartográficas.

Os motivadores desta demanda derivam de incompatibilidades entre as versões 3 e 4 do Qt, que impactaram negativamente o desenvolvimento do projeto E-Foto. A infraestrutura de computação gráfica do Qt que, até então, dava suporte à apresentação de imagens e marcações, passou a ficar demasiadamente lenta na versão Linux da EFD do E-Foto[2]. A razão se deve, basicamente, ao fato do Qt, a partir da versão 4, utilizar a arquitetura do Flash, que fica lenta em ambiente Unix. Além disso, o código da implementação do Flash utilizada pelo Qt não é aberto. Portanto, não é possível conhecer os detalhes de implementação que poderiam garantir o desempenho desejado e a portabilidade esperada da EFD.

Após a análise de sistemas da visualização e marcação de imagens de alta resolução, ficou evidente a necessidade de desenvolvimento de um *framework* capaz de concentrar as responsabilidades relacionadas a estes recursos na EFD E-Foto, incluindo, por exemplo, *zoom*, marcação de pontos e visualização 3D. Outros requisitos importantes são a compatibilidade entre as diversas licenças de software, além do acesso ao código fonte das APIs utilizadas. O levantamento das alternativas levou em conta também a necessidade de restabelecimento da estabilidade, da portabilidade de plataformas de sistemas operacionais e da compatibilidade entre versões perdida a partir do Qt 4. Assim, ficou definida a utilização da arquitetura de computação gráfica Open Graphics Libray (OpenGL) (Harn and Baker, 2004) e da biblioteca multimídia Simple Directmedia Layer (SDL) (Loki Software e Hall, 2001) como infraestrutura para o desenvolvimento do *framework* ora apresentado. O uso do OpenGL possui a vantagem adicional de proporcionar uma velocidade final melhor, por ser executado diretamente pelo hardware gráfico.

O presente documento esta organizado da seguinte forma. A próxima seção analisa e apresenta os requisitos de visualização e marcação de imagens da EFD E-Foto, enquanto a seção a seguir se dedica ao OpenGL e ao SDL. As seções que se seguem apresentam respectivamente o projeto do *framework* ora proposto e os resultados obtidos. Por fim são apresentadas as conclusões.

2 A EFD E-Foto é multiplataforma. A disponibilização regular de versões para os sistemas operacionais Linux e Windows é um requisito não funcional definido pelo projeto E-Foto.

2. Levantamento de requisitos do *framework* de apresentação de marcações e imagens

Entende-se por fotogrametria (Coelho Filho e Brito, 2007) o conjunto de técnicas e rotinas de processamento de imagens fotográficas, visando a reconstrução do espaço tridimensional (espaço-objeto) através de imagens bidimensionais (espaço-imagem). Tais técnicas formam um conjunto de etapas do processo fotogramétrico. Basicamente, tais etapas podem ser subdivididas em três grupos: cadastramento de dados de entrada, cálculo dos parâmetros das transformações e produção de resultados fotogramétricos. Dentre as diversas etapas deste processo, a seguir, são enumeradas aquelas que demandam a utilização do *framework* ora proposto: (1) Inclusão de imagens ao projeto fotogramétrico; (2) Orientação interior; (3) Resecção espacial; (4) Fototriangulação; (5) Restituição fotogramétrica 3D; (6) Extração do MNE; (7) Ortoretificação; (8) Normalização de pares estereoscópicos; e (9) Retificação de imagens. A Tabela 1 apresenta o resultado do levantamento de requisitos do *framework* ora proposto.

Tabela 1. Requisitos de marcação e visualização de imagens do processo fotogramétrico

Atividades	**Requisitos dos recursos gráficos do *framework* de apresentação de marcações e imagens de alta resolução**
Inclusão de imagens ao projeto fotogramétrico	1. Leitura e carregamento de imagens de alta resolução 2. Apresentação de imagens de alta resolução - 2.1. *Zoom* - 2.2. *Pan*
Orientação interior	1. Leitura e carregamento de imagens de alta resolução 2. Apresentação de imagens de alta resolução - 2.1. *Zoom* - 2.2. *Pan* 3. Apresentação de marcações - 3.1. Marcas fiduciais 4. Medição de pontos - 4.1. Medição das marcas fiduciais
Resecção espacial	1. Leitura e carregamento de imagens de alta resolução 2. Apresentação de imagens de alta resolução - 2.1. *Zoom* - 2.2. *Pan* 3. Apresentação de marcações - 3.1. Marcas fiduciais - 3.2. Pontos de controle e de teste 4. Medição de pontos - 4.1. Medição dos pontos de controle e de teste
Fototriangulação	1. Leitura e carregamento de imagens de alta resolução 2. Apresentação de imagens de alta resolução - 2.1. *Zoom* - 2.2. *Pan* 3. Apresentação de marcações - 3.1. Marcas fiduciais - 3.2. Pontos de controle, de teste e de amarração 4. Medição de pontos - 4.1. Medição dos pontos de controle, de teste e de amarração
Restituição fotogramétrica 3D	1. Leitura e carregamento de imagens de alta resolução 2. Apresentação de imagens de alta resolução - 2.1. *Zoom* - 2.2. *Pan* 3. Apresentação de marcações - 3.1. Marcas fiduciais - 3.2. Pontos de controle, de teste e de amarração - 3.3. Feições cartográficas 4. Medição de pontos - 4.1. Medição 3D dos vértices das feições 5. Visualização 3D - 5.1. Separação espacial - 5.2. Anaglifo (Jensen, 2009)
Extração do MNE	1. Leitura e carregamento de imagens de alta resolução 2. Apresentação de imagens de alta resolução - 2.1. *Zoom* - 2.2. *Pan* 3. Apresentação de marcações - 3.1. Marcas fiduciais - 3.2. Pontos de controle, de teste e de amarração - 3.3. Feições cartográficas - 3.4 Grade do terreno. 4. Medição de pontos - 4.1. Medição 3D dos vértices das feições 5. Visualização 3D - 5.1. Separação espacial - 5.2. Anaglifo
Ortoretificação	1. Leitura e carregamento de imagens de alta resolução 2. Apresentação de imagens de alta resolução - 2.1. *Zoom* - 2.2. *Pan*
Normalização de pares estereoscópicos	1. Leitura e carregamento de imagens de alta resolução 2. Apresentação de imagens de alta resolução - 2.1. *Zoom* - 2.2. *Pan* 3. Apresentação de marcações - 3.1. Marcas fiduciais - 3.2. Pontos de controle, de teste e verificação da qualidade da normalização 4. Medição de pontos - 4.1. Medição dos pontos de verificação da qualidade da Normalização
Retificação de imagens	1. Leitura e carregamento de imagens de alta resolução 2. Apresentação de imagens de alta resolução - 2.1. *Zoom* - 2.2. *Pan* 3. Apresentação de marcações - 3.1. Pontos de controle 4. Medição de pontos - 4.1. Medição de pontos de controle

Os requisitos enumerados na coluna da direita da Tabela 1 estão apresentados em detalhe na Tabela 2.

Tabela 2. Detalhamento dos requisitos de marcação e visualização de imagens

1. Leitura e carregamento de imagens de alta resolução
Leitura de imagens de alta resolução e carregamento para a memória do computador numa estrutura que permita a realização de operações de processamento de imagens
2. Apresentação de imagens de alta resolução - 2.1. *Zoom* - 2.2. *Pan*
Apresentar imagens de alta resolução previamente carregadas na memória na interface gráfica utilizando diferentes fatores de escalas e quantidades de movimento linear que podem ser alterados pela manipulação do usuário – não deve ser utilizado nenhum filtro de processamento de imagens e a perspectiva do *viewport* deve ser ortogonal
3. Apresentação de marcações - 3.1. Marcas fiduciais - 3.2. Pontos de controle, de teste, de amarração e de verificação da qualidade da normalização - 3.3. Feições cartográficas
Apresentar os diferentes tipos de marcações em imagens utilizadas na fotogrametria digital com flexibilidade de padrão de simbologia
4. Medição de pontos - 4.1. Medição dos pontos de controle, de teste e de amarração - 4.2. Medição 3D dos vértices das feições
Capturar os eventos de clique e movimento do *mouse* de forma a permitir a obtenção precisa das coordenadas do cursor no sistema da imagem digital e quando necessário convertê-lo para outros sistemas de coordenadas 2D e 3D
5. Visualização 3D - 5.1. Separação espacial - 5.2. Anaglifo
Apresentação do terreno em 3D usando as técnicas de separação espacial e anaglifo

A seção a seguir apresenta brevemente a infraestrutura de computação gráfica utilizada no desenvolvimento do *framework* de apresentação de marcações e imagens de alta resolução.

3. Infraestrutura de Computação Gráfica

3.1 Arquitetura OpenGL

OpenGL (Write el al. 2007) é uma arquitetura aberta para computação gráfica independente tanto de plataforma de sistema operacional quanto de dispositivo gráfico. Sua arquitetura é considerada um padrão industrial para aplicações gráficas. A API da OpenGL incorpora funções que fornecem acesso a praticamente todos os recursos do hardware de vídeo. A base do desempenho desta arquitetura se deve ao fato do código OpenGL ser executado diretamente pelo hardware de vídeo.

3.2 Biblioteca SDL

SDL é uma biblioteca multimídia multiplataforma e livre escrita na linguagem de programação C. Esta biblioteca possibilita o acesso a recursos de áudio, de gerenciamento de janelas gráficas, e a outros dispositivos de entrada e saída, além, através do OpenGL, ao hardware 3D e ao *framebuffer* de vídeo 2D. Entretanto, no presente artigo, seu uso se limita ao processamento de imagens e à passar essas imagens para o OpenGL.

4 Desenvolvimento do *Framework*

A modelagem das classes do *framework* ora proposto foi realizada tendo como meta o atendimento dos requisitos apresentados na Tabela 2. O primeiro dos diagramas de classe que ilustram a estrutura do *framework* de apresentação de marcações e imagens de alta resolução é mostrado na Figura 1. Nele podem ser observadas as Classes SImage, STile, SWidget, SPin, SPoint e Sviewport, acompanhadas da indicação da utilização, onde cabível, das bibliotecas SDL e OpenGL.

A classe SWidget, que é, basicamente, responsável pela exibição dos objetos gráficos na interface gráfica, possui em sua composição objetos das classes SImage, SPin, SPoint e Sviewport. Tais classes estão respectivamente encarregadas da representação das imagens de alta resolução, dos ícones das marcações dos pontos, dos pontos e dos *viewports*. A classe STile, por sua vez, é responsável por garantir a máxima eficiência possível durante a

manipulação das imagens de alta resolução através do hardware gráfico. Isto é conseguido através da subdivisão da imagem em *tiles* em conformidade com as máximas dimensões suportadas pelo hardware de vídeo. O teste para a verificação deste valor é feito em tempo de execução pelas classes do próprio *framework*.

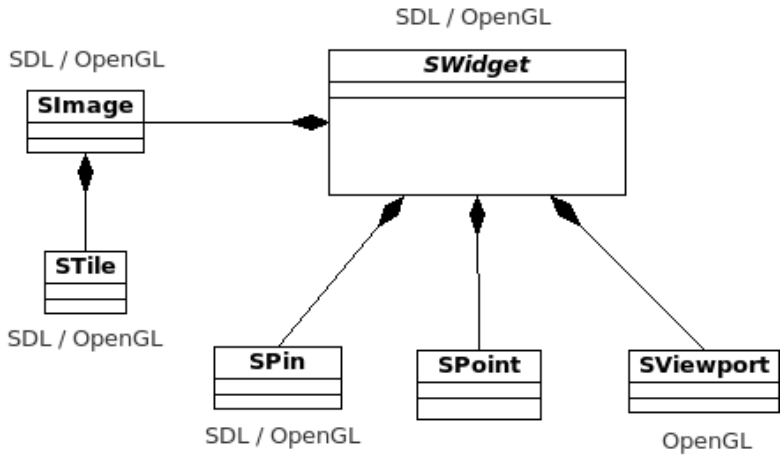

Figura 1. Diagrama de classes conceitual fornecendo a composição da Classe SWidget

A Figura 2, na qual novamente está explicitado o uso do SDL e OpenGL, apresenta relacionamentos entre classes anteriormente omitidos.

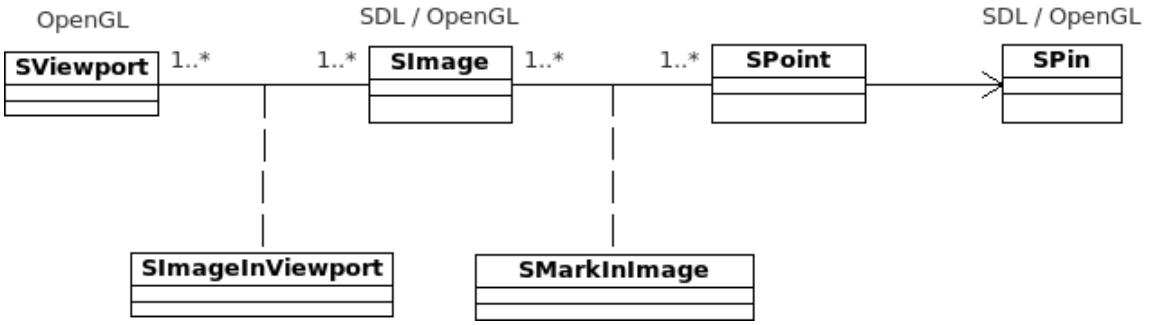

Figura 2. Diagrama de classes conceitual explicitando os relacionamentos entre as classes SViewport , SImage, SPoint e SPin.

Na Figura 2 pode ser notada a existência de associações bidirecionais entre as classes SViewport e SImage e entre SImage e SPoint. A multiplicidade destas associações é do tipo muitos para muitos. Como consequência deste fato, precisaram ser introduzidas duas classes de associação SImageInViewport e SmarkInImage. Por fim, nota-se também a existência de uma associação unidirecional de SPoint para SPin.

A análise da distribuição das responsabilidades entre as diversas classes do *framework* requer o reexame da Tabela 2. O requisito 1, leitura e carregamento de imagens de alta resolução, é uma responsabilidade compartilhada entre as classes SImage e STile. Por outro lado, o atendimento ao requisito 2, apresentação de imagens de alta resolução incluindo recursos de *zoom* e *pan* independentes nas diferentes exibições das imagens, é possibilitado pelas classes SViewport, SImage, SImageInViewport e SWidget. Estas mesmas classes dão suporte ao requisito 5, visualização 3D através dos métodos separação espacial e anaglifo. O requisito 3, apresentação de marcações relativas às marcas fiduciais, aos pontos de controle, de teste, de amarração e de verificação da qualidade da normalização, além das feições cartográficas, é suportado pelas classes SImage, SmarkInImage, SPoint e SPin. Por fim, o requisito 4, medição dos diferentes tipos de pontos requer a captura de eventos. Esta

responsabilidade varia em função do *framework* de interface gráfica. A respeito da conversão entre os sistemas de coordenadas a fim de obter-se a coordenada no espaço imagem, esta responsabilidade fica a encargo da classe QGLMethods, que concentra o código OpenGL do *framework*. Outro ponto, que precisa ser enfatizado é o fato da classe abstrata SWidget precisar ser implementada valendo-se dos recursos disponibilizados pelo *framework* de interface gráfica utilizado. Deve ser mencionado que, no presente artigo, isso for realizado por intermédio da IDE Qt, mais especificamente, através da biblioteca QtOpenGL.

5. Resultados e Discussão

A presente seção analisa o desempenho do *framework* de apresentação de marcações e imagens de alta resolução desenvolvido neste artigo ao fornecer suporte aos requisitos apresentados nas Tabelas 1 e 2. Esta análise simula a utilização do *framework* nas operações de Orientação Interior, Orientação Exterior e Restituição Fotogramétrica 3D. Deve ser mencionado que esta abordagem experimental contempla todos os requisitos enumerados.

O protótipo desenvolvido para os experimentos está implementado no *framework* Qt4. Sua escolha se deve somente ao fato da equipe de desenvolvimento do projeto E-Foto estar familiarizada com ele. A interface do protótipo permite que o usuário crie e destrua *viewports* com diferentes dimensões, adicione e remova imagens, podendo ainda escolher através de quais *viewports* as imagens serão visualizadas. Além disso, a interface permite escolher a visualização em modo anaglifo, ativar a movimentação das imagens, o *zoom* ou a marcação de pontos. No último caso, a interface permite que o usuário adicione marcações, escolhendo a imagem do ícone que será utilizado para simbolizá-lo. Para um melhor controle das imagens, a interface fornece uma lista com todas as imagens, *viewports* e pontos criados pelo usuário.

As seções a seguir apresentam e analisam os resultados dos experimentos.

5.1 Avaliação Uso do *Framework* na Orientação Interior

O presente experimento avalia o suporte dado à Operação de Orientação Interior. Os requisitos a serem suportados pelo *framework* nesta operação são o carregamento e a visualização de imagens, *pan*, *zoom*, a medição de pontos e a apresentação das respectivas marcações.

Na metade esquerda da Figura 3, pode-se observar os resultados obtidos na marcação de pontos correspondentes aos vértices do edifício principal da Uerj e, em sua porção direita a aplicação de *zoom* a fim de facilitar a medição de uma marca fiducial. Como pode ser observado, o recurso de ampliação da imagem (*zoom in*) não utiliza nenhuma técnica de *anti-aliasing* ou antiserrilhamento (Baker and Hearn, 2004), mostrando em detalhes os pixels da imagem. Isso é necessário, pois, para se medir as marcas fiduciais é necessário observar com clareza o pixel mais brilhante.

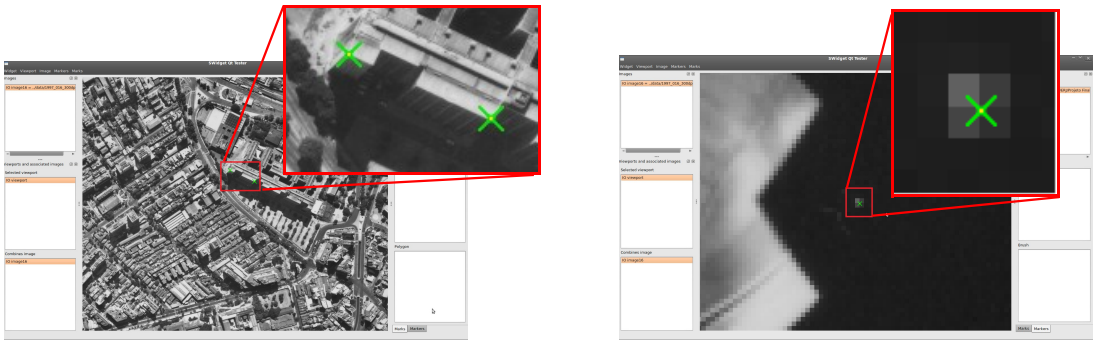

Figura 3. – Resultado da marcação de pontos e da aplicação de *zoom* em imagens

5.2 Avaliação do Uso do *Framework* na Orientação Exterior

O presente experimento tem como objetivo avaliar o suporte dado à Orientação Exterior, operação que requer as medições dos pontos de controle na imagem. Deve ser enfatizado que, ao se analisar as Tabelas 1 e 2, se pode concluir que as funcionalidades que foram avaliadas no experimento anterior são basicamente as mesmas exigidas para este novo teste. Contudo, no caso da Orientação Exterior, pode ser desejável a inclusão de um um *minimap*, que pode ser observado no canto superior esquerdo da área de visualização do *framework*, de forma que ele auxilie na navegação da imagem. Este *minimap* nada mais é do que a mesma imagem sendo exibida em um outro *viewport* com uma dimensão bem menor do que o *viewport* principal. Veja a Figura 4.

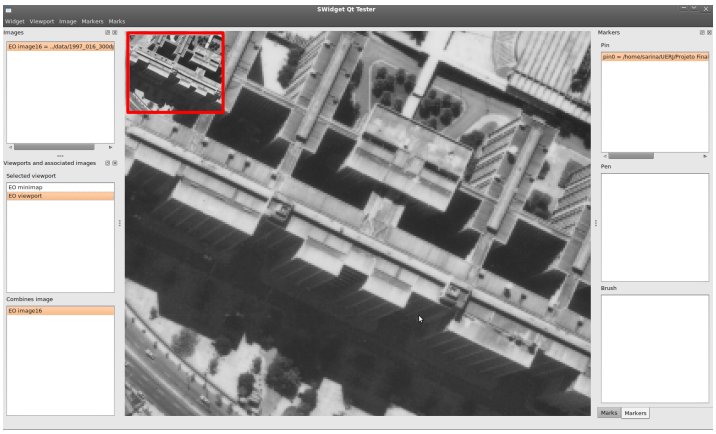

Figura 4 - Ilustração do resultado obtido com o segundo teste de avaliação, o uso do *framework* na OE com o auxilio de um *minimap* no canto esquerdo superior

5.3 Avaliação do Uso do *Framework* na Restituição Fotogramétrica

Além das funcionalidades testadas anteriormente, a avaliação do uso do *framework* na visão estéreo, exige a apresentação simultânea de duas imagens distintas em *viewports* diferentes. A porção esquerda da Figura 5 ilustra o resultado deste experimento no modo de visão 3D com a técnica de separação espacial.

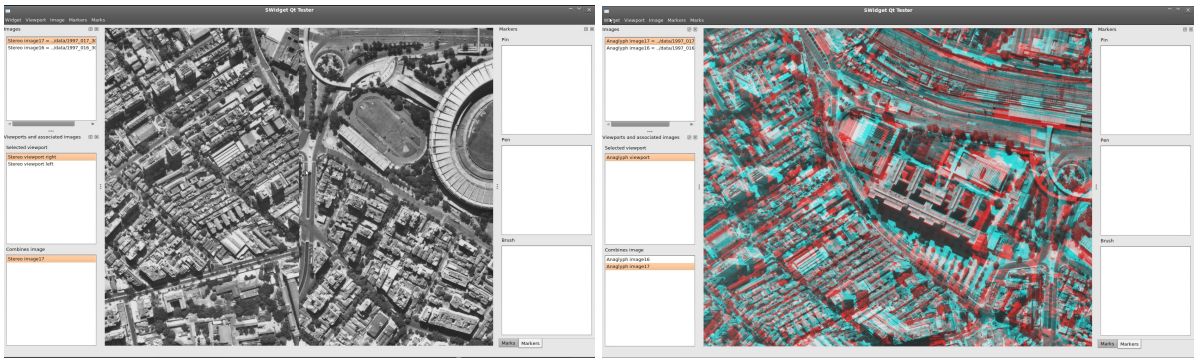

Figura 5. Ilustração da visão estéreo obtida com o terceiro teste de avaliação nos modos separação espacial e anaglifo

No caso da visão estéreo usando a técnica do anaglifo, exige do *framework*, além das funcionalidades testadas anteriormente, a possibilidade de apresentar duas imagens diferentes em um mesmo *viewport*. Em cada imagem é aplicado um filtro de cor específico e as imagens são sobrepostas. Além disso, é preciso permitir a movimentação de forma independente das imagens. Este resultado do presente experimento pode ser observado na porção direita da Figura 5.

6. Conclusões

O presente artigo se dedica a apresentar à comunidade de sensoriamento remoto o *framework* de apresentação de marcações e imagens de alta resolução na interface gráfica da estação fotogramétrica digital livre E-Foto. A plataforma utilizada no seu desenvolvimento compreende API gráfica OpenGL e a biblioteca multimídia SDL.

Inicialmente, foram feitos testes de prova de conceito a fim de confirmar a viabilidade do uso da OpenGL e da SDL para o desenvolvimento do *framework*. Tais testes compreenderam o desenvolvimento de pequenos programas exemplo que implementam todos os requisitos do *framework*. Em seguida, deu-se inicio a modelagem conceitual do *framework* através da definição das classes, das respectivas responsabilidades e relacionamentos entre classes. O modelo conceitual foi, posteriormente, derivado em um modelo físico e implementado na linguagem de programação C++.

O *framework* desenvolvido foi analisado quanto ao atendimento dos requisitos previamente elicitados. Para a realização desta análise, foi necessário o desenvolvimento de um protótipo. Devido a grande familiaridade da equipe de desenvolvimento do projeto com a IDE Qt4, esta foi escolhida como a plataforma de desenvolvimento do protótipo. Assim sendo, o *framework* foi analisado quanto a seu uso em algumas etapas do processo fotogramétrico, são elas: Orientação Interior, Orientação Exterior e Restituição Fotogramétrica. Dentre os recursos avaliados encontram-se a manipulação e visualização das imagens carregadas, juntamente com *zoom*, *pan*, transparência, aplicação de filtros de cor, marcação de pontos e desenho de figuras geométricas na tela.

Os resultados obtidos não deixam dúvida quanto ao atendimento aos requisitos por parte do Framework, que satisfazem os requisitos e objetivos deste trabalho. Consequentemente, o *framework* ora apresentado será integrado, pela equipe de desenvolvimento do projeto, à Estação Fotogramétrica Digital Livre E-Foto.

6. Referências Bibliográficas

Blanchette, Jasmin; Summerfield, Mark. **C++ GUI Programming with Qt 4**. 1. ed. Massachusetts: Prentice Hall, 2006. 537 p.

OpenGL - The Industry's Foundation for High Performance Graphics. Disponível em <http://www.opengl.org/. 31 jan. 2010>.

Simple DirectMedia Layer. Disponível em <>. 31 jan. 2010

Pazera, Ernest. **Focus on SDL**. ed. Course Technology PTR, 2002. 336p

Stroustrup, Bjarne. **A Linguagem de Programação C++**. 3. ed. Porto Alegre: Bookman, 2000. 823p.

Brito, Jorge Nunes; Coelho, Luiz. **Fotogrametria Digital**. 1. ed. Rio de Janeiro: EdUERJ, 2007. 196p.

Baker, M. Pauline; Hearn, Donald. **Computer Graphics with OpenGL**. ed. Pearson Prentice Hall, 2004. 857p.

Shreiner, David. **OpenGL Programming Guide**. ed. Addison-Wesley Professional, 2009. 936p.

Jensen, John R. **Remote Sensing of the Environment**. ed. Pearson, 2009. 592p.

Loki Software, Inc.; Hall, John R. **Building Multimedia Applications with SDL, OpenAL, and Other APIs.** São Francisco – EUA: No Starch Press, 2001.

Write, R.S.; B. Lipchal, N. Haemel, 2007. **OpenGL SuperBible: Comprehensive Tutorial and Reference.** Pearson Education Inc., 1248 p.