André Puel

COMUNICAÇÃO EM TEMPO REAL PARA PLATAFORMA MULTIMÍDIA COLABORATIVA BASEADA EM WEB

Dissertação submetida ao Programa de Pós-graduação em Ciência da Computação para a obtenção do Grau de Mestre em Ciências da Computação.
Orientador: Prof. Dr. Renato Fileto
Coorientador: Prof. Dr. Rogério de Almeida Richa

Florianópolis(SC)

2014

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Puel, André
Comunicação em Tempo Real para Plataforma Multimídia Colaborativa Baseada em Web / André Puel ; orientador, Renato Fileto ; coorientador, Rogério de Almeida Richa. - Florianópolis, SC, 2014.
132 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da Computação. 2. computação. 3. telemedicina. 4. web. 5. vídeo-chamada. I. Fileto, Renato. II. Richa, Rogério de Almeida. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

André Puel

COMUNICAÇÃO EM TEMPO REAL PARA PLATAFORMA MULTIMÍDIA COLABORATIVA BASEADA EM WEB

Este Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Ciências da Computação", e aprovada em sua forma final pelo Programa de Pós-graduação em Ciência da Computação.

Florianópolis(SC), 29 de setembro 2014.

Prof. Dr. Ronaldo dos Santos Mello
Coordenador

Banca Examinadora:

Prof. Dr. Renato Fileto
Orientador

Prof. Dr. Rogério de Almeida Richa
Coorientador

Prof. Dr. Douglas Dyllon Jeronimo de Macedo
Universidade Federal de Sergipe

Prof. Dr. Roberto Willrich
Universidade Federal de Santa Catarina

Prof. Dr. rer. nat. Eros Comunello
Universidade Federal de Santa Catarina

À minha querida família.

AGRADECIMENTOS

Gostaria de agradecer ao Prof. Dr. Renato Fileto, pela oportunidade que me foi dada de realizar este mestrado. Ao Prof. Dr. ret. nat. Aldo von Wangenheim, pela estrutura propícia a pesquisa e ao crescimento acadêmico. E ao Prof. Dr Rogério de Almeida Richa pelas inúmeras orientações que me guiaram neste trabalho. Aos colegas e amigos de trabalho, pelas experiências trocadas, que são de grande valia para o crescimento profissional, bem como por criar o melhor ambiente de trabalho. Aos meus pais, pois sem eles nada disso teria sido possível. E a minha namorada, pois sem ela eu não conseguiria ter chego até aqui.

“If you think it’s simple, then you have misunderstood the problem”

(Bjarne Stroustrup)

RESUMO

Este trabalho apresenta uma ferramenta que integra funcionalidades de visualização de imagens médicas de forma colaborativa com comunicação multimídia, vídeo chamadas, completamente baseada em padrões *web* (sem uso de *plugin* auxiliar). O objetivo deste trabalho é o desenvolvimento de uma plataforma *web* com usabilidade correspondente a plataformas *desktop* tradicionais. Os componentes necessários são discutidos individualmente e também os problemas de integração com a plataforma *web*. É utilizada uma estratégia para comparar a qualidade da plataforma apresentada com *softwares* baseados em *desktop*, a validação de usabilidade com usuários através da Escala de Usabilidade de Sistema (do inglês, System Usability Scale, SUS). Através desta validação é possível inferir que a plataforma *web-based* consegue cumprir os mesmos objetivos que uma versão do *software* em *desktop*.

Palavras-chave: telemedicina, telerradiologia, DICOM, PACS, web services, visualização, usabilidade, HTML5, WebRTC, sistemas colaborativos, vp8, h264, jpeg.

ABSTRACT

This work presents a tool that integrates collaborative medical images visualization functionalities with multimedia communication, video calls, fully based on web standards (without the use of auxiliary plugins). The objective of this project is the development of a web platform with usability matching traditional desktop's. The required components are discussed individually and also the the problems that arise with the integration in the web platform. A strategy to compare the quality of the presented platform with desktop based softwares is presented, the evaluation with users through System Usability Scale (SUS). Through this evaluation it is possible to infer that the web-based platform can achieve the same objectives than a dekstop version of the software.

Keywords: telemedicine, teleradiology, DICOM, PACS, web services, visualization, usability, HMLT5, WebRTC, collaborative system, vp8, h264, jpeg.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Os principais objetivos da telemedicina são tornar a saúde pública accessível para a população fora de grandes centros urbanos e acelerar o processo de diagnóstico. Transportar pacientes de suas cidades para hospitais mais bem equipados custa tempo e dinheiro, e sabe-se que diagnóstico tardio afeta o resultado do quadro clínico do paciente. Além disso, a telemedicina também pode ser um facilitador para o próprio paciente, que pode acessar os resultados de seus exames sem precisar se locomover até a clínica. O fácil acesso a informações, como histórico médico, é de extrema utilidade para o especialista que está realizando um diagnóstico.

Com telemedicina é possível propagar a cobertura da saúde através de tele-consultas e modernizar o acesso aos dados clínicos do paciente. Outro aspecto a ser considerado além de consulta entre paciente e médico, é que a telemedicina também pode fornecer colaboração de um ou mais médicos sob um caso. Um médico pode pedir uma segunda opinião para ajudá-lo a formar um diagnóstico de um outro especialista que não está fisicamente presente no momento ou até mesmo trabalharem juntos ao mesmo tempo.

Os especialistas podem cooperar através de marcações e desenhos feitos nas próprias imagens para destacar alguma informação relevante na imagem. Essas anotações são salvas e acessadas por outro especialistas que podem adicionar mais informações. A cooperação é melhorada quando há a possibilidade de manuseio das imagens ao mesmo tempo de forma colaborativa, isto é, as anotações feitas por um usuário são instantaneamente disponíveis para os outros e vice-versa. Na cooperação em tempo real, torna-se necessário algum meio de comunicação, como bate papo textual ou conversa por voz.

Para alcançar esses resultados, é desejável que as ferramentas de telemedicina tenham acesso ao histórico médico do paciente e possam visualizar e interagir com exames de imagens médicas. Para isso, a ferramenta tem que se comunicar transparentemente com PACS (*Picture Archiving and Communication Systems*) onde essas informações podem ficar armazenadas. A visualização das imagens do PACS depende de navegação, manipulações de brilho e contraste, medir a

distância entre dois pontos, no caso de tomografias ainda há a necessidade de a interpretação da densidade em escala de Hounsfield (HU) e a visualização tridimensionais da informação através de reconstrução.

Com uma abordagem baseada em *web*, qualquer dispositivo que tenha um navegador *web* instalado e conexão com Internet está pronto para usar essas ferramentas, isso inclui *smartphones*, *tablets*, computadores públicos, computadores pessoas e até computadores dentro do próprio ambiente de trabalho. Além da usabilidade, quando o dado está disponível na *web,* ganha se confiabilidade contra falhas, uma máquina com problemas não é impedimento para um médico acessar informações de seus pacientes. Por exemplo, o médico pode diagnosticar um caso de emergência de um cibercafé*,* onde não é possível instalar *softwares* adicionais. Técnicas de segurança garantem que o sistema não vai ser comprometido mesmo quando usado a partir de máquinas não confiáveis.

As tecnologias *web* estão ficando cada vez mais capazes, de forma que é possível construir aplicações *web* que tem mesmas funcionalidades e flexibilidade que aplicações *desktop*. Com a implantação do *HyperText Markup Language* (HTML) na versão 5, principalmente a API (Interface de Programação de Aplicação) para desenhos livres, o suporte a requisições de *JavaScript* e XML Assíncrono (AJAX) e a comunicação em tempo real, tornou-se possível uma suíte de aplicação puramente *web-based* sem usar qualquer *plugin* ou *software* externo.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Esta dissertação tem como propósito o estudo de tecnologias *web* para uma solução em *software* associada com o processo de trabalho de diagnóstico em medicina, o gerenciamento de imagens médicas para diagnóstico e apoio a decisão de forma colaborativa em ambientes de telemedicina.

Como a tecnologia de páginas *web* surgiu com um propósito muito diferente de aplicações *desktop*, aplicações *web* eram extremamente limitadas em comparação a soluções puramente em *desktop*.

Este trabalho diminui a distância entre esses dois mundos, criando uma solução *web* em que as funcionalidades e a usabilidade seja comparável a uma solução feita para *desktop*.

Além disso, os seguintes requisitos devem ser atendidos:

1. a solução deve se comunicar com PACS que sigam o padrão DICOM (*Digital Imaging and Communications in Medicine*),
2. deve suportar visualização colaborativa das imagens médicas com marcações e sincronização entre usuários,
3. deve suportar videoconferência e bate-papo textual,
4. deve usar apenas tecnologias *web* padronizadas sem o uso de *plugins* externos.

Os três primeiros requisitos são importantes para uma ferramenta de telemedicina com suporte a colaboratividade. Já o último requisito fala de aspectos técnicos de implementação e é importante para garantir que a solução seja acessível de qualquer plataforma.

1.1.2 Objetivos Específicos

Para pesquisar a viabilidade de tecnologias *web* como ferramenta cooperativa, os seguintes objetivos devem ser atingidos.

- Fazer um levantamento de todo trabalho relacionado com ferramenta *web-based* de telemedicina com as funcionalidades já descritas analisando que tipo de tecnologia e quais abordagens foram utilizadas nos trabalhos relacionados, principalmente como foram divididas as responsabilidades entre cliente (máquina do usuário) e servidor.
- Elaborar a aplicação utilizando os novos recursos disponíveis em navegadores *web* modernos, isto é, com suporte a HTML5.
- Realizar um estudo experimental das ferramentas, analisando suas funcionalidades quanto a qualidade da interação com o usuário através de validação de usabilidade de forma a fazer uma comparação com alternativas baseadas em *desktop*.

1.1.3 Estrutura da Dissertação

Este trabalho está dividido da seguinte forma. No Capítulo 2 é apresentado o levantamento de trabalhos relacionados, foi utilizado o procedimento de Revisão Sistemática, que também é brevemente descrito neste capítulo. É feito uma análise e comparada esta dissertação em relação aos pontos em que os outros trabalhos deixam em aberto. A fundamentação teórica sobre o processo de compactação de vídeo é discutido no Capítulo 3 e a comunicação em rede está descrita no Capítulo 4. Esses dois tópicos são os pilares básicos para a solução proposta, que é apresentada no Capítulo 5, são relatadas as técnicas utilizadas na implementação. . Para avaliar a qualidade do *software* proposto, foi analisado o desempenho da ferramenta de visualização e foi feita uma avaliação de usabilidade utilizando a Escala de Usabilidade de sistema, o processo e os resultados são descritos no Capítulo 6.2. Finalmente, são apresentadas as conclusões e possíveis trabalhos futuros no Capítulo 7.

2 REVISÃO DA LITERATURA

Este capítulo descreve o processo de levantamento de trabalhos relacionados desta dissertação. Foi utilizado o processo de Revisão Sistemática (KITCHENHAM; CHARTERS, 2007), a Seção Processo explica como funciona uma Revisão Sistemática e relata as diretrizes utilizadas. A Seção Protocolo apresenta o protocolo utilizado. E finalmente, a Seção Resultados contém os dados extraídos bem como a análise dos artigos selecionados.

2.1 PROCESSO

O processo de Revisão Sistemática inicia com uma questão de pesquisa que deseja-se responder, então é definido um protocolo com o objetivo de levantar toda literatura relevante a esta questão. Idealmente, como o protocolo é sistemático, qualquer pessoa seguindo-o selecionará os mesmos artigos. Isto é, o objetivo é minimizar preferências pessoais do revisor em relação a questão de pesquisa levantada, por exemplo, achar artigos que contradizem a hipótese favorável a pesquisa.

O protocolo da revisão foi elaborada seguindo as regras estabelecidas por Kitchenham e Charters (2007), com algumas mudanças para se adaptar melhor ao contexto desta dissertação de mestrado. O levantamento de bibliográfica é dividido em cinco etapas. Cada artigo selecionado passa pelas quatro primeiras etapas independentemente dos outros artigos, enquanto que a quinta etapa só é executada quando todos artigos foram selecionados.

A primeira etapa é baseada em *strings* de busca (*query*) que os motores de busca de bibliotecas *online* irão processar. A busca é totalmente automatizada pois são os motores de busca que farão o processamento utilizando algoritmos para casar metadados e conteúdo de artigos com os termos definidos. O resultado das buscas é um conjunto de artigos que será refinado por mais duas etapas, é incomum que novos artigos sejam achados nas etapas seguintes, portanto considera-se a etapa de busca como a etapa base para todos artigos. E por isso, a

busca deve ser mais abrangente do que intolerante. Nessa etapa é melhor levantar todos artigos relevantes ao custo de artigos irrelevantes também entrando para o resultado.

Os motores de busca geralmente fornecem o resultado ordenado por relevância, os artigos que o motor julga mais relevante estão no começo da lista, enquanto que os artigos que estão para o fim da lista provavelmente não são relevantes mas houve alguma correspondência parcial com o termo de busca. Portanto é comum definir um limite de profundidade na busca, são considerados apenas artigos até certa posição no resultado na busca. Nesta revisão, esse limiar é definido dinamicamente de acordo com os próprios resultados na segunda etapa da pesquisa, onde os artigos são analisados na ordem definida pelo motor de busca, se muitos resultados consecutivos não satisfazerem os critérios desta etapa, automaticamente todo os artigos subsequentes são descartados.

A busca é o processo mais rápido (por ser automatizada) e gera um volume muito grande de literatura. A segunda etapa, onde são aplicados critérios de inclusão, tem como objetivo minimizar esse volume de artigos, no entanto, essa etapa deve ser rápida para que seja praticável diante o grande volume de resultados fornecidos pela primeira etapa. Já a terceira etapa, onde aplicam-se os critérios de exclusão, tem como objetivo fazer um refinamento mais fino dos artigos levantados.

Os critérios de inclusão analisam apenas os metadados dos artigos, título, palavras-chaves e resumo. O processo é rápido pois basta um dos itens satisfazer o critério para que o artigo vá para a etapa seguinte, por exemplo, é possível que o título do artigo seja o suficiente para a segunda etapa.

Já os critérios de exclusão formam a etapa mais minuciosa, para aplicar esses critérios o artigo inteiro é utilizado. O critério de exclusão fará o contrário do critério de inclusão, buscará no artigo razões para ele não ser relevante. A análise em cima do artigo completo é feita na seguinte ordem:

1. *abstract*,
2. conclusão,
3. sessão mencionando tecnologias utilizadas (se houver alguma),
4. introdução,

5. o restante do artigo

Para um artigo ser descartado, basta um dos itens casar com os critérios de exclusão. Enquanto que para um artigo ser selecionado, ou seja, passar para etapa de extração de dados, todos os itens serão analisados. No entanto, é importante notar que uma leitura completa do artigo não é feita na etapa de exclusão, isto é feito na quarta etapa, a etapa de extração de dados. Essa ordem, também é definida visando acelerar o processo, geralmente o *abstract* é o suficiente para aplicar os critérios de exclusão.

Desta forma, as etapas são executadas numa ordem que maximiza o desempenho sem prejudicar o resultado final. A primeira etapa é muito abrangente e levanta todos artigos relevantes e muitos artigos irrelevantes. A segunda etapa aplica os critérios de inclusão e aceita qualquer artigo que case com os critérios. Enquanto que a terceira etapa aplica os critérios de exclusão e rejeita qualquer artigo que case com os critérios. Os critérios de inclusão são criados de forma a rapidamente passar artigos para a etapa seguinte, e os critérios de exclusão para rapidamente descartar artigos.

Nenhum critério de qualidade foi aplicado para excluir artigos porque o nosso objetivo é levantar todas abordagens (e tecnologias utilizadas).

A extração de dados visa sumarizar de forma sucinta as abordagens utilizadas pelo estado da arte num único lugar, geralmente os dados são sintetizados para ser apresentados em itens ou de forma tabular. O objetivo é dar uma visão geral do estado da arte e permitir comparação objetiva entre os estudos. Em contrapartida, a etapa de relatório descreve textualmente os artigos selecionados e serve como uma análise mais aprofundada de cada trabalho perante a questão de pesquisa.

2.2 PROTOCOLO

Para este levantamento de literatura, a questão de pesquisa é: “Quais tecnologias e técnicas são utilizadas para desenvolver ferramentas *web* para visualização de imagens médica com suporte a colaboratividade?”. Esta questão de pesquisa envolve quatro aspectos:

1. telemedicina
2. visualização (e manipulação) de imagens
3. baseada em tecnologias *web*
4. com suporte a colaboratividade

Esses aspectos foram utilizados para criar a *string* de busca. Os motores de buscas suportam *queries* avançadas utilizando e-lógico (*AND*) e ou-lógico (*OR*), dessa forma, a busca é constituída por quatro componentes em *AND* entre si, um para cada aspecto, sendo que cada componente é um conjunto de *OR* de forma a contemplar todas formas diferentes de referenciar ao tópico. O termo de busca genérico é:

> (“medicine” OU “medical” OU “telemedicine” OU “telerradiology” OU “radiological”) OU (“image” OU “visualization” OU “manipulation”) E (“web-based” OU “rich internet application” OU “web application” OU “web-browser” OU “web-browser”) E (“collaboratively” OU “collaboration” OU “collaborative” OU “collaborativity” OU “multi-user” OU “communication” OU “cooperation”)

O principal critério de inclusão artigos é o artigo ser a respeito de ferramentas *web* para visualização de imagens médicas, com ou sem suporte a colaboratividade em tempo real. Foram excluídos artigos que o foco não é uma ferramenta *web-based* para a visualização (e manipulação) de imagens médicas, também foi excluído qualquer solução que não rode de dentro do navegador (alguns artigos utilizam abordagem *web* em cima de uma aplicação *desktop*). Por fim, artigos que não tem sua versão integral disponível também foram descartados.

As bibliotecas digitais utilizadas são ACM Digital Library, ScienceDirect, Springer e IEEE Xplore. A escolha dessas quatro bibliotecas digitais foi feita através de pesquisas piloto utilizando o agregador Google Scholar. A Tabela 1 lista as *strings* utilizadas para cada biblioteca digital.

Tabela 1 – *Strings* de busca para cada biblioteca digital

Biblioteca	*String* de Busca	Observações
ACM DL	("medicine image" or "medical image" or "medicine visualization" or "medical visualization") and ("web-based" or "rich internet application" or "web application" or "webbrowser" or "web-browser") and ("collaboratively" or "collaboration" or "collaborative" or "collaborativity" or "multi-user" or "communication" or "cooperation")	Os componentes "*medical*" e "*image*" tiveram que ser unidos num único componente pois os resultados estavam fugindo totalmente do escopo da questão de pesquisa.
ScienceDirect	((medicine) or (medical) or (telemedicine) or (telerradiology) or (radiological)) and ((image) or (visualization) or (manipulation)) and ((web-based) or (rich internet application) or (web application) or (webbrowser) or (web-browser)) and ((collaboratively) or (collaboration) or (collaborative) or (collaborativity) or (multi-user) or (communication) or (cooperation))	
Springer	("medicine" OR "medical" OR "telemedicine" OR "telerradiology" OR "radiological") AND ("image" OR "visualization" OR "manipulation") AND ("web-based" OR "rich internet application" OR "web application" OR "webbrowser" OR "web-browser") AND ("collaboratively" OR "collaboration" OR "collaborative" OR "collaborativity" OR "multi-user" OR "communication" OR "cooperation")	Resultados encontrados apenas para "*Online Contents*"
IEEE Xplore	("medicine" OR "medical" OR "telemedicine" OR "telerradiology" OR "radiological") AND ("image" OR "visualization" OR "manipulation") AND ("web-based" OR "rich internet application" OR "web application" OR "webbrowser" OR "web-browser") AND ("collaboratively" OR "collaboration" OR "collaborative" OR "collaborativity" OR "multi-user" OR "communication" OR "cooperation")	Utilizada opção de fazer busca no artigo inteiro (ao invés de apenas metadados)

2.3 RESULTADOS

A Tabela 2 enumera a quantidade de artigos levantados com a busca em cada biblioteca digital e a quantidade remanescente após cada etapa.

Tabela 2 – Quantidade de artigos levantados em cada etapa da revisão para cada biblioteca digital

Biblioteca	Busca	Max. Profundidade	Inclusão	Exclusão
ACM DL	78	78	29	10
ScienceDirect	1.539	75	13	5
Springer	9.251	100	8	2
IEEE Xplore	1.469.579	100	10	4
Total	1.480.447	353	60	21

Dentre os trabalhos relacionados selecionados, encontram-se três tipos de abordagens. A utilização do modelo cliente-servidor básico com o *browser* apenas mostrando as imagens médicas enviadas pelo servidor, a utilização de *plugins* para aumentar a flexibilidade da aplicação, e a utilização de *streaming* de vídeo de forma que o *browser* seja apenas um terminal que recebe imagens e submete comandos.

O trabalho de Zhang et al. (2005) representa uma aplicação *web* tradicional onde o servidor envia recursos para o cliente que trata de exibi-los. Programação via *JavaScript* fornece a lógica da aplicação e da colaboratividade, dois ou mais usuários podem controlar a aplicação ao mesmo tempo. Nesta aplicação, o servidor tem responsabilidade mínima do ponto de vista de processamento, cabe ao servidor apenas a transferência de recursos.

A aplicação de Matsopoulos et al. (2004) possui um servidor mais ativo. Através de extensões ISAPI (*Internet Server Application Programming Interface*), o servidor da aplicação faz processamento nas imagens médicas a medida que o cliente as requisitas, como por exemplo janelamento de imagens tomográficas, segmentação e filtros diversos. Desta forma, a responsabilidade da aplicação fica em grande parte ao servidor *web*.

Estendendo esta ideia ao extremo, Iserhardt-Bauer et al. (2001)

colocam toda a lógica da aplicação no servidor. O cliente apenas define parâmetros da aplicação, e carrega o vídeo que é gerado pelo servidor. Constantinescu, Kim e Feng (2012) utilizam a ideia de interface gráfica sincronizável entre cliente e servidor, o cliente apenas desempenha o papel de interface gráfica do aplicação rodando no servidor. Finalmente, o CoWebViz (KASPAR; PARSAD; SILVERSTEIN, 2010) é uma aplicação de reconstrução 3D de imagens médicas onde todo o processamento gráfico e lógica da aplicação é executada no servidor, que envia imagens que representam o estado atual da aplicação e o cliente responde apenas com os comandos do usuário (manuseio do teclado). Todas estas aplicações atingem seus objetivos utilizando apenas recursos disponíveis nos navegadores, isto é, sem a utilização de *plugins*. O modelo de aplicação através *streaming* de imagens suporta colaboratividade naturalmente, pois todos os usuários conectados interagem sobre a mesma instância de aplicação no servidor, operações que são feitas por um usuário são imediatamente vistas por outro usuários.

Em contrapartida, outras abordagens utilizam *plugins* para que mais coisas possam ser feitas no lado cliente de forma a melhorar o tempo de resposta da aplicação e diminuir a carga do servidor. Por fornecerem a mesma flexibilidade que uma aplicação *desktop*, Java *Applet* é uma abordagem bastante utilizada dentro os artigos levantados (KIM et al., 2001; KIM; FENG; CAI, 2001; ABRARDO; CASINI, 1998; MATA et al., 2012; LIM; FENG; CAI, 2001). Shen et al. (2014) utilizam o *plugin* Adobe Flash, Hsiao et al. (2011) utilizam Microsoft Silverlight. O suporte a vídeo conferência de Sung et al. (2000) é fornecido por componentes ActiveX, enquanto que Chan, Lim e Feng (2002) e Sánchez, Triana e Romero (2008) utilizam *applet* Java para dar suporte a vídeo conferência. Por fim, o *framework* Adobe Flex é utilizado por Shen et al. (2012), também com suporte a vídeo conferência.

No entanto, *plugins* não são tecnologias padronizadas, isto é, não há garantia que um navegador *web* terá suporte a esse tipo de aplicação. Além disso, muitos *plugins* não são totalmente multiplataforma, principalmente quando se trata de dispositivos móveis (*smartphones* e *tablets*).

Já a utilização de tecnologias HTML puras garante a funcionalidade da aplicação em qualquer plataforma. A aplicação de Wang,

Rabsch e Liu (2008) utilizam o suporte a *Scalable Vector Graphics* (SVG) de *browsers* para criar um sistema de anotações gráficas em imagens sem a utilização de *plugins*. Enquanto que Mahmoudi et al. (2010) propõem uma arquitetura multi-camada e *multiplugin*, a aplicação pode ser vista como uma abordagem híbrida em relação as outros trabalhos apresentados. Parte do processamento de imagens roda no servidor, a visualização das imagens médicas é implementada através de HTML puro, as filtragens de imagens são feitas utilizando *applet* Java, enquanto que a visualização 3D é feita utilizando VRML (*Virtual Reality Markup Language*). Se algum *plugin* não estiver disponível na máquina do usuário, a funcionalidade é simplesmente desabilitada sem afetar o restante da aplicação.

A Tabela 3 contém os dados extraídos dos artigos selecionados com relação a recursos de colaboratividade, comunicação, validação de usabilidade, quais tecnologias foram utilizadas e quais técnicas são propostas. Grande parte dos trabalhos relacionados em que foi feita alguma avaliação de usabilidade, considerou-se o resultado pelo menos aceitável, porém não há nenhum que proponha uma solução que não utilize *plugins* mas que ao mesmo tempo forneça todos recursos apresentados no Capítulo 1. Esta dissertação propõe uma solução com suporte a colaboratividade, comunicação e usabilidade utilizando a tecnologia HTML5 com a carga de processamento voltada para o lado cliente da aplicação de forma a preencher esta lacuna observada nos trabalhos relacionados.

Tabela 3: Dados extraídos dos artigos selecionados

Referência	Colaboratividade	Comunicação	Usabilidade	Tecnologias	Técnicas
(ISERHARDT-BAUER et al., 2001)	Nenhuma	Nenhuma	Baixa	Java Server Pages, HTML puro	Renderização 3D de um vídeo que é mandado ao cliente.
(HAMZA-LUP; DAVIS; ZEIDAN, 2006)	Nenhuma	Nenhuma	Não disponível	Java Server Pages, HTML	Renderização 3D *client-side* utilizando VRML.

Dados extraídos dos artigos selecionados (continuação)

Referência	Colaboratividade	Comunicação	Usabilidade	Tecnologias	Técnicas
(KIM et al., 2001)	*Offline*	Nenhuma	Boa	Java Server Pages, Java Applet	Servidor fornece imagens que são visualizadas utilizando *applets*.
(KIM; FENG; CAI, 2001)	Nenhuma	Nenhuma	Boa	Python (server side), Java Applet	Servidor fornece imagens que são visualizadas utilizando *applets*.
(LIM; FENG; CAI, 2001)	Manipulação colaborativa de imagens	*Chat* textual	Boa	CGI (server side), Java Applet	Servidor fornece imagens que são visualizadas utilizando *applets*. Colaboratividade intermediada pelo servidor central.

Dados extraídos dos artigos selecionados (continuação)

Referência	Colaboratividade	Comunicação	Usabilidade	Tecnologias	Técnicas
(CHAN; LIM; FENG, 2002)	Manipulação colaborativa de imagens	*Chat* textual e vídeo chamadas	Boa	CGI (server side), Java Applet	Servidor fornece imagens que são visualizadas utilizando *applets*, conexão direta entre clientes para a vídeo chamada.
(SáNCHEZ; TRIANA; ROMERO, 2008)	*Offline*	Vídeo chamadas	Boa	Java Server Pages, e HTML com alguns componentes em Java Applet	Imagens são visualizadas utilizando Java Applet, outros recursos são implementa-dos em HTML.
(KASPAR; PARSAD; SILVERS-TEIN, 2010)	Controle simultâneo na visualização de imagem	Nenhuma	Razoável	Server próprio em C++, HTML	*Streaming* de vídeo para fornecer visualização 3D no *browser* sem utilização de recursos da máquina do usuário.

Dados extraídos dos artigos selecionados (continuação)

Referência	Colaboratividade	Comunicação	Usabilidade	Tecnologias	Técnicas
(MATSOPOULOS et al., 2004)	*Offline*	Nenhuma	Boa	Server com módulos ISAPI e ASP, HTML	Servidor fornece imagens que são visualizadas utilizando HTML e *JavaScript*.
(ZHANG et al., 2005)	Manipulação colaborativa de imagens com *dual-cursor*	Nenhuma	Boa	Server Microsoft IIS, HTML	Servidor fornece imagens que são visualizadas utilizando HTML e *JavaScript*. Colaboratividade intermediada pelo servidor central.
(MAHMOUDI et al., 2010)	Nenhuma	Nenhuma	Boa	Componentes em C++, ASP, HTML, VRML, Java Applet	Sistema multi-camada que usa os *plugins* disponíveis no *browser* mas mantém funcionalidade básica se não houver.

Dados extraídos dos artigos selecionados (continuação)

Referência	Colaboratividade	Comunicação	Usabilidade	Tecnologias	Técnicas
(SHEN et al., 2014)	Nenhuma	Nenhuma	Boa	Java (server side), Adobe Flash	Servidor fornece parte das imagens sob demanda que são visualizadas utilizando Flash.
(SUNG et al., 2000)	Marcação colaborativa sob imagens	*Chat* textual e vídeo chamadas	Razoável	CORBA, Java, Active X e Java *Applet*	Servidor fornece imagens que são visualizadas utilizando *applets*. Colaboratividade intermediada pelo servidor central. Vídeo chamada utilizando ActiveX.
(HSIAO et al., 2011)	Nenhuma	Nenhuma	Razoável	DICOM WADO, Microsoft Silverlight	O *download* de imagens médicas é feita diretamente do servidor DICOM através do protocolo WADO.

Dados extraídos dos artigos selecionados (continuação)

Referência	Colaboratividade	Comunicação	Usabilidade	Tecnologias	Técnicas
(SHEN et al., 2012)	Marcação colaborativa sob imagens	Chat textual e vídeo chamadas	Boa	Flex, Adobe Flash	O servidor converte imagens DICOM para JPEG e o cliente visualiza com Flash.
(CONSTANTINESCU; KIM; FENG, 2012)	Controle simultâneo da visualização	Nenhuma	Boa	Java em server side, e Flash iOS nativo em client side	O servidor faz *streaming* de imagens e qualquer dispositivo pode visualizar e controlar
(WANG; RABSCH; LIU, 2008)	Marcação colaborativa sobre imagens médicas	Nenhuma	Boa	SVG e AJAX	O padrão SVG (suportado nativamente em *browsers* modernos) é utilizado para fazer anotações sobre as imagens, AJAX é utilizado para fornecer colaboratividade em tempo real

Dados extraídos dos artigos selecionados (continuação)

Referência	Colaboratividade	Comunicação	Usabilidade	Tecnologias	Técnicas
(ABRARDO; CASINI, 1998)	Nenhuma	Nenhuma	Boa	Java Server Pages, Java Applet	Servidor fornece imagens que são visualizadas utilizando *applets*.
(MATA et al., 2012)	*Offline*	Nenhuma	Boa	Java Applet	Servidor se comunica com o PACS e converte imagens DICOM que são visualizadas utilizando *applets*.

3 FORMATOS DE VÍDEO DIGITAL

Neste capítulo, serão apresentadas as principais técnicas de compactação de vídeo. A abordagem utilizada nesta explicação é de baixo para cima. Primeiro serão explicados conceitos básico e definições sobre cores, como cores formam uma imagem, como uma imagem pode ser compactada, como uma sequência de imagens formam um vídeo, e como vídeos são compactados. No final desta capítulo são apresentados os compactadores de vídeo com importância comercial no contexto atual.

3.1 COR E LUMINÂNCIA

Cor é um fenômeno baseado na percepção do observador, a definição de cor não se baseia puramente em fenômenos físicos, mas na reação da luz em seu espectro visível incidindo sobre a retina. Espectro visível é a denominação dada a luz cujo comprimento de onde está na faixa de 400nm até 700nm, fora desta faixa, um ser humano não consegue perceber a luz. Exemplos notáveis são a radiação infravermelho e ultravioleta. A luz pode ser representada por uma distribuição de potência espectral (do inglês, *spectral power distribution,* SPD), comumente dividida em partes de 10nm (POYNTON, 2006).

A utilização de SPD para representar a luz visível fornece um sistema que usa 30 componentes numéricos para representar uma única informação de cor. No entanto, o sistema visual humano é composto por três tipos células cones, cada uma com sensibilidade diferente para diferentes comprimentos de onda, desta forma, qualquer cor – isto é, percepção da distribuição espectral – pode ser representada apenas usando três componentes. A função que transforma a SPD em estímulos tricromáticos leva em consideração a área da curva de sensibilidade das células cones. Estas curvas tem picos bem definidos, porém há sobreposição, desta forma, existem combinações que nunca aconteceriam na prática, como por exemplo o estímulo em apenas um dos tipos das células, com as outras totalmente desestimuladas.

A Figura 1a demonstra o espaço de cores LMS (FAIRCHILD,

2013), o nome LMS vem do inglês *long, medium* e *short* e se refere aos comprimentos de onda em que cada célula cone tem seu pico de sensibilidade, respectivamente comprido, médio e curto. O espaço de cores CIE XYZ *(SMITH; GUILD, 1931)* foi criado a partir de experimentos com pessoas e tem a característica de ser capaz de representar qualquer cor (POYNTON, 2006), a Figura 1b demonstra a função de conversão de espectro para XYZ, tendo Y como valor relacionado ao brilho e XZ relacionado a cor. Finalmente, a Figura 1c representa o espaço de cores RGB, do inglês *Red, Green, Blue,* vermelho, verde e azul respectivamente, segundo a recomendação BT.709 (ITU, 2002).

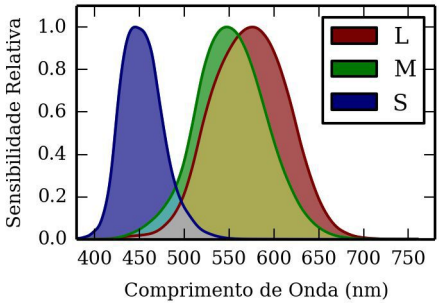
Naturalmente, a percepção humana em relação ao brilho de um objeto não pode ser medida de forma exata. No entanto, sabe-se que o brilho não é percebido de forma linear em relação a luminância do objeto. Por exemplo, uma pessoa percebe um objeto que tem metade da luminância da cor branca (luminância máxima no sistema de cores) como apenas $1/4$ a menos de brilho. A sensação de metade do brilho só é alcançada quando a luminância cai para 18% em relação a cor branca (POYNTON, 2002). O gráfico na Figura 2 demonstra essa relação entre brilho percebido e luminância do objeto.

Entender a forma como o ser humano percebe a luz é importante para que seja possível codificar vídeos de forma eficiente. Uma codificação de cor trivial, que não leva em consideração a relação do brilho percebido com a luminância, gastaria uma faixa de valores grande demais para representar cores claras indistinguíveis enquanto que a faixa de valores que representa cores escuras não possui valores o suficiente para ser distinguível. Poynton (2012) aponta que 12 *bits*, 4096 valores, são necessários para representar satisfatoriamente luminância, enquanto que se a luminância for convertida para luminosidade (a curva da Figura 2), bastam 8 *bits*, 256 valores. Quando um incremento ou decremento no valor codificado causa a mesma quantidade de diferença percebida independente se a mudança ocorre no topo ou na base do intervalo de valores codificados, a codificação é dita perceptualmente uniforme. Neste caso, a codificação de 50% teria metade da luminosidade da codificação de 100%. Desta forma, o sinal codificado é mais tolerante a erros, tanto os ruídos provenientes da captura quanto artefatos gerados por algoritmos de compactação.

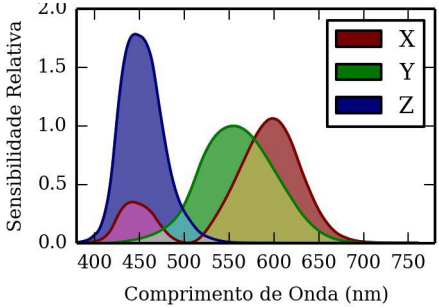
Outro aspecto importante da percepção humana é que a nitidez de imagens em tons de cinza é maior do que a nitidez de imagens

Figura 1 – Espaço de Cores em função do comprimento de onda do feixe de luz. O valor tricromático é calculado como o somatório da intensidade de cada feixe de luz ponderado pela sensibilidade relativa de acordo com este gráfico

(a) Espaço de Cores LMS, cada curva está relacionado a um tipo de célula cone



(b) Espaço de Cores CIE XYZ



(c) Espaço de cores RGB BT.709

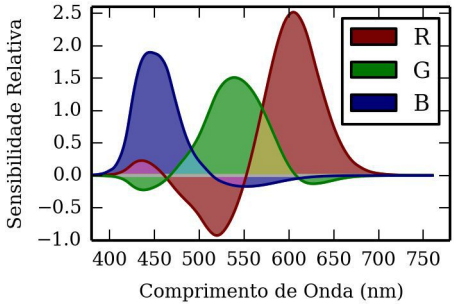
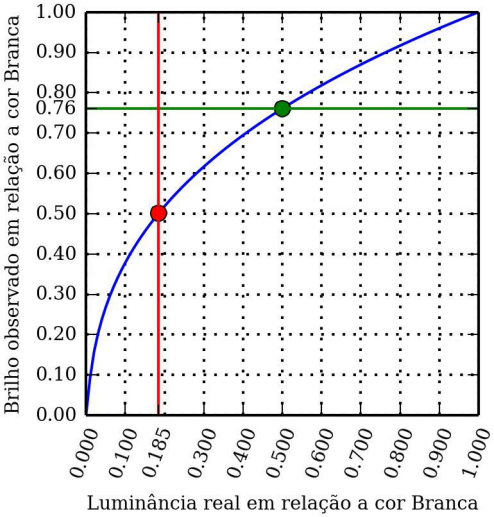


Figura 2 – Luminosidade: Relação entre luminância e brilho percebido, em destaque o quanto que influência diminuir a luminância pela metade e onde que um observador tem a impressão de que o brilho caiu pela metade



com apenas informação de cor (luminância constante). A Figura 3 demonstra esse efeito utilizando um degradê com 256 tons de cinza e 256 tons de cor, embora ambos tenham mesma quantidade de cores, a parte colorida parece variar menos.

3.2 CODIFICAÇÃO

Uma imagem é representada como uma matriz de *pixels*. Cada *pixel* representa uma cor dentro do sistema utilizado pela imagem. Imagens geralmente são codificadas em escala de cinza com um componente de 8 *bits* representando a intensidade de luz, e em RGB *true-color* com cada *pixel* carregando três componentes, vermelho, verde e azul, utilizando 24 *bits*. Um vídeo consiste de uma sequência de imagens em movimento. No contexto de vídeo, cada uma das imagens é denominada quadro.

A codificação de cor Y'CbCr separa a cor numa componente de luminância (Y') e em dois componentes de crominância (CbCr). O componente de crominância, como é menos nítido para a visão, pode ser codificado com menos informação. Tipicamente o componente de crominância é sub-amostrado, isto é, o tamanho da imagem é diminuído na largura ou na largura e altura, por um fator de divisão que é uma potência de 2. Supondo uma imagem tradicional codificada em Y'CbCr, são necessários 3 valores por cada *pixel*, se for aplicada uma sub-amostragem de fator 4, isto é a largura e a altura dos componentes de crominância foram divididas por 2, cada cada 4 *pixels* ocuparão 4 valores de Y', 1 valor de Cb e 1 valor de Cr. Sem a sub-amostragem, cada 4 *pixels* ocupam 12 valores. A sub-amostragem com fator 4 diminui a quantidade de espaço pela metade.

Existe uma notação numérica para representar os tipos de sub-amostragem em relação ao fator de divisão. A notação consiste de três números, representando respectivamente, o fator do componente de luminância (fator de referência), o fator horizontal dos componentes de crominância e o fator horizontal dos componentes de crominância em linhas ímpares. Por exemplo, a notação "4:2:0" significa que para cada linha, para cada 4 *pixels* do componente de luminância, haverão 2 *pixels* na componente de crominância, em linhas ímpares não haverá *pixel* nenhum na componente de crominância, ou seja, a sub-

Figura 3 – Comparação da nitidez do componente de luminância com a nitidez no componente de crominância. A impressão é que o degradê da faixa inferior muda menos que o da faixa superior.

amostragem divide o tamanho das componentes de crominância pela metade tanto horizontalmente quando verticalmente. Na prática, o terceiro fator sempre será igual ao segundo ou igual a zero. Quando igual a zero significa que há metade do tamanho verticalmente, caso contrário, a crominância não é afetada verticalmente.

A Figura 4a demonstra a sub-amostragem com vários fatores, começando com fator 1 até o fator 256 na última imagem, onde a mesma crominância é utilizada na imagem inteira. Mesmo assim, é possível reconhecer os pássaros na imagem. Enquanto que a Figura 4b demonstra uma sub-amostragem hipotética na componente de luma, é possível perceber que sub-amostragem no componente de luma é mais nocivo para a percepção visual da imagem do que no componente de crominância. Já as Figuras 4c, 4d e 4e exemplificam a sub-amostragem utilizando o espaço de cores RGB BT.709 (ITU, 2002) para demonstrar como este espaço de cores é inadequado para a aplicação de alguma sub-amostragem.

Espaço de cores podem ser convertidos de um para o outro através de uma combinação dos componentes do espaço original, cada componente do espaço destino é calculado como uma soma com pesos dos três componentes da origem, geralmente se representa esse tipo de conversão como multiplicação matricial. Dessa forma, o espaço de cores Y'CbCr pode ser obtido através do RGB tradicional como mostram as Equações (3.1) e (3.2) (FAIRCHILD, 2013).

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.2126 & 0.7152 & 0.0722 \\ -0.1146 & -0.3854 & 0.5 \\ 0.5 & -0.4541 & -0.0459 \end{bmatrix} \cdot \begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} \quad (3.1)$$

(a) Sub-amostragem no componente de crominância

(b) Sub-amostragem no componente de luma

(c) Sub-amostragem no componente vermelho

(d) Sub-amostragem no componente verde

(e) Sub-amostragem no componente azul

Figura 4 – Exemplos de tipos diferentes de sub-amostragem com fatores de 1, 2, 4, 8, 16, 32, 64, 128 e 256 sobre uma imagem de tamanho 256x256.

$$\begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} = \begin{bmatrix} 1 & 0.0002 & 1.5748 \\ 1 & -0.1874 & -0.4681 \\ 1 & 1.8556 & 0.0001 \end{bmatrix} \cdot \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} \quad (3.2)$$

3.2.1 Compactação

Uma imagem em alta definição (resolução de 1080 linhas por 1920 colunas) representada com sub-amostragem de crominância 4:2:0 ocupa 3MB de armazenamento, um vídeo com mesmo tamanho de imagens e frequência de 25 imagens por segundo precisaria de taxa de transferência de aproximadamente 75MB/s.

Compactadores de dados armazenam a informação de forma mais compacta do que ela é originalmente. O processo de descompressão recupera o dado original exatamente como é. Compactadores de dados tiram proveito de redundâncias no dado original para diminuir o tamanho do arquivo.

O sistema *Run-Length Encoding* (RLE) representa eficientemente dados repetidos, cada valor é acompanhado por um contador de repetições que indica quantas vezes o valor aparece logo em seguida (SALOMON, 2004). Se fosse usado, por exemplo, para representar uma imagem totalmente branca, seria muito eficiente dado que é um único valor repetido por toda a extensão do arquivo. Por outro lado, se fosse usado para representar uma fotografia, onde raramente dois *pixels* consecutivos tem o mesmo valor, seria altamente ineficiente pois todos os valores do dado original seriam representados seguindo por um contador de repetições com o valor 1.

A codificação de Huffman (HUFFMAN, 1952) utiliza uma tabela de frequência sobre o dado original para criar uma representação que utilize menos *bits* nos valores que são mais comuns, é chamado de codificação de comprimento variável pois nem todos os valores são representados com o mesmo comprimento de *bits*. Como os valores mais frequentes utilizam menos espaço para ser representados, ocorre uma diminuição no tamanho final do arquivo. A eficiência da codificação de Huffman depende da precisão da tabela de frequência utilizada, se o dado a ser codificado não casa com a tabela usada, o arquivo final pode ser maior do que o arquivo original. Este tipo de codificação é bastante eficiente quando um valor específico acontece muito mais ve-

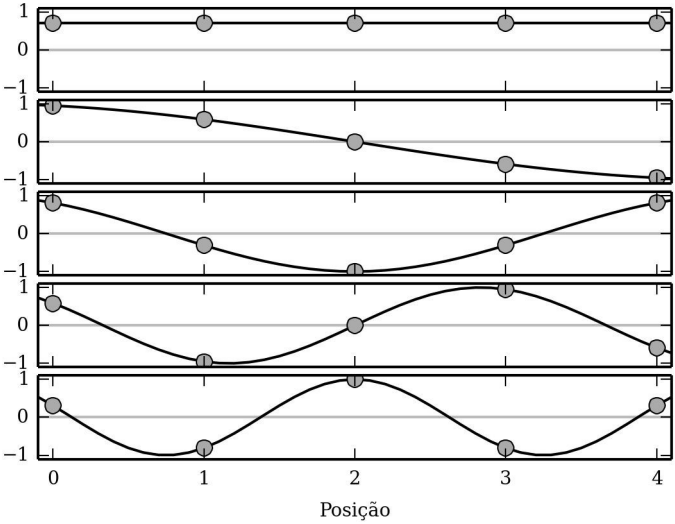
zes do que os outros valores, como por exemplo um desenho sob um fundo branco, como a cor branca é predominante, ela receberá a codificação de tamanho mínimo, quando a imagem é codificada, a maior parte dos *pixels* será representado com poucos *bits*. O Apêndice (B) contém um exemplo em que a codificação de Huffman é utilizada para reduzir um arquivo de texto para a metade do tamanho original.

Um compactador de imagem consegue ser mais eficiente do que um compactador genérico pois tira proveito do domínio de imagens, principalmente a redundância espacial, como por exemplo a tendência de *pixels* próximos terem valores semelhantes em uma fotografia. A compactação é melhorada se não houver a necessidade de representar a imagem sem perdas, isto é, se a imagem original não for exatamente igual a imagem resultante do processo de compactação seguido pelo processo de descompactação. E geralmente não há essa necessidade, é possível representar a mesma imagem com certo nível de erro que não é relevante para a percepção do observador, a liberdade que a tolerância a erros fornece aumenta significativamente a eficiência do compactador. A utilização de codificação que leva em consideração a percepção humana é importante para melhorar a tolerância a erros causados pelo algoritmo de compactação. Dessa forma, a presença de erros de baixa intensidade afeta o resultado visual apenas em baixa intensidade.

Além da transformação do espaço de cores, existem também transformações que expressam a imagem original no domínio da frequência. A Transformada Discreta de Cosseno representa o dado original como uma soma de cossenos com amplitude e frequência variada, é comumente denominado pela sigla DCT, da expressão em inglês *Discrete Cosine Transform*. O resultado da transformação é uma sequência de amplitudes do mesmo tamanho, chamadas de coeficientes, as frequências de cada curva cosseno são fixas e dependem do tamanho da sequência de dados originais. Com os coeficientes calculados, é possível recuperar os dados originais utilizando o DCT inverso (IDCT) (AHMED; NATARAJAN; RAO, 1974).

Por exemplo, ao aplicar o DCT em um vetor qualquer de 5 valores, são obtidos 5 coeficientes. Esses coeficientes representam a amplitude do cosseno de frequência zero (fator constante), de meia, uma, uma e meia, e duas oscilações dentro do intervalo $[0,5)$. Isto é, o primeiro cosseno tem comprimento infinito enquanto que o último

Figura 5 – Interpretação geométrica do *Discrete Cosine Transform*. Qualquer sequência de 5 números pode ser representado como uma soma ponderada destes cossenos.



cosseno tem comprimento $4/2$. A Figura 5 demonstra esse exemplo.

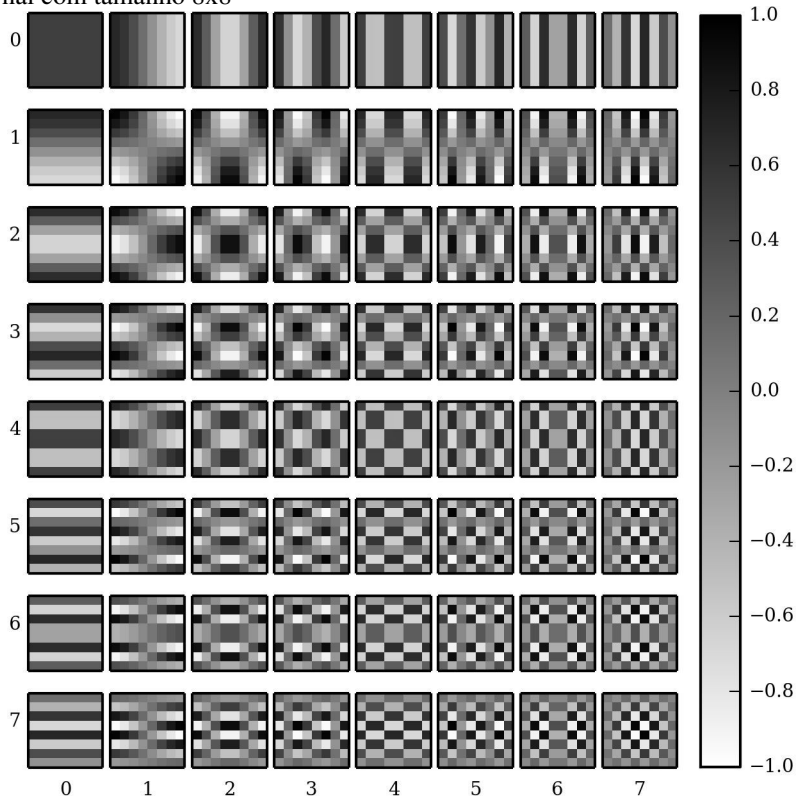
A transformação DCT tipicamente contém valores altos nos primeiros coeficientes, que são os coeficientes de baixa frequência, e contém valores mais baixos nos coeficientes de alta frequência. No dado original, quanto mais suave for a transição de valores em posições sequencias, menos importante serão os coeficientes de alta frequência. Ou seja, o domínio do DCT representa bem redundâncias que existem na imagem, dessa forma, o algoritmo de compactação pode escolher representar com menos precisão dados que tendem a não ser representativos no vetor original.

Os coeficientes do DCT, que são números reais, podem ser passados para uma representação limitada, como por exemplo arredondar um número para que ele seja representado utilizando o conjunto dos inteiros. Um número inteiro pode ser visto como tendo precisão de tamanho 1, ou seja, não haverá dois valores que a distância entre eles é menor que 1. O tamanho da precisão pode ser alterado para um valor qualquer através de uma divisão seguida por um arredondamento. É isso que faz o processo de quantização, o vetor de quantização contém a quantidade de precisão de cada coluna e deve ser conhecido pelo decodificador pois será necessário fazer uma multiplicação para recuperar a escala original do valor.

Para trabalhar com dados multidimensionais, o DCT deve ser aplicado isoladamente em um eixo de cada vez. Para matrizes (e imagens), o DCT é primeiramente aplicado para cada uma das linhas separadamente, com a matriz resultante desta operação, o DCT é aplicado uma segunda vez em cada uma das colunas separadamente. Para recuperar os valores originais, o DCT inverso é aplicado em ordem inversa.

A reconstrução de uma matriz utilizando o IDCT pode ser vista como uma soma ponderada das matrizes fatores do IDCT. Cada fator é multiplicado pelo seu respectivo coeficiente e somado na matriz resultante, que inicialmente tem valor zero. Tomando como exemplo uma matriz 8x8, existem 64 fatores que serão multiplicados pelos 64 coeficientes resultantes do DCT e somados na matriz resultante. A Figura 6 contém os fatores do IDCT antes de passarem pela multiplicação, cada fator representa uma combinação de frequência horizontal e frequência vertical. Qualquer matriz 8x8 pode ser representa como uma soma ponderada das 64 matrizes da figura.

Figura 6 – Fatores do *Discrete Cosine Transform* inverso bidimensional com tamanho 8x8



Novamente a quantização pode ser aplicada, dessa vez utilizando uma matriz para representar a precisão desejada em cada elemento. O valor do coeficiente DCT representa a influência máxima que cada uma das curvas de cosseno terá na reconstrução. Desta forma, é possível prever o pior caso na hora de quantizar um valor. A Figura (7) demonstra os efeitos da quantização sobre uma foto.

Em imagens em que há suavidade na mudança de cor dos *pixels*, por exemplo fotos, a tendência a acumular valores maiores continua existindo para as primeiras linhas e colunas da matriz de coeficientes. A medida que o índice do elemento da matriz aumenta, os valores tendem a ter valores cada vez mais próximos de zero. Dessa forma, a quantização pode ser mais grosseira em valores de alta frequência. O Apêndice (C) contém uma série de análises da eficiência do DCT como forma de compactar dados através da diminuição da precisão dos coeficientes resultantes.

Um aspecto importante a ser notado no DCT é que, dentro de um bloco, um coeficiente representa parcialmente todos os *pixels* codificados. Quando o processo de compactação resolve descartar um um dos elementos da matriz de coeficientes, esse descarta afeta parcialmente todos os *pixels* do bloco original ao invés de afetar inteiramente um único *pixel*. Desta forma, como o erro criado pelo descarte é dividido, ele se torna mais tolerável para o observador.

3.2.2 JPEG

JPEG é o padrão de codificação e compactação de imagens definido pelo *Joint Photographic Experts Group (IEC, 1994).* O padrão conta com diferentes *perfis* de codificação, cada um com parâmetros e algoritmos diferentes. O *perfil* básico (*baseline*) é o mais importante comercialmente (POYNTON, 2012). Apenas o perfil *baseline* será descrito a seguir.

O primeiro passo é converter a imagem para o formato Y'CbCr com sub amostragem 4:2:0, o que já diminuiu a quantidade de espaço necessário para representar a imagem para metade do tamanho original. Após a conversão, os planos Y', Cb *e* Cr serão tratados individualmente com pela mesma sequencia de passos, variando apenas os parâmetros que são utilizados.

Figura 7 – Quantização dos coeficientes de DCT em fotos utilizando blocos de 8x8

(a) Quantização com boa precisão no centro da imagem, pontos distantes do centro de quantização mais brusca

(b) Quantização aplicada nos cantos da imagem

A unidade mínima de codificação no JPEG consiste de 4 blocos de *pixels* em forma de quadrado da imagem original. Com a conversão para Y'CbCr *4:2:0* isso significa dois blocos de crominância (Cb e Cr) e os respectivos quatro blocos de luminância. Cada bloco possui tamanho 8x8, que é um tamanho definido para equilibrar eficiência da compactação e desempenho do algoritmo. Cada bloco é convertido para o espaço de frequência utilizando DCT.

Até este ponto, todo o processo é sem perda – a troca do sistema de cores não é considerado como perda, apenas como representação alternativa. Com os coeficientes do DCT é possível restaurar os dados originais de cada bloco, a adição de erro ocorre no processo de quantização. A matriz de quantização que é responsável por regular a fidelidade da compressão a imagem original, valores alto na matriz de quantização diminuem a necessidade de espaço para representar a imagem porém degradam muito a qualidade visual. Como a matriz de quantização é variável, o codificador inclui no arquivo qual matriz foi utilizada. A matriz de quantização tem valores maiores em coeficientes de alta frequência pois eles são mais difíceis de serem notados pela visão humana (POYNTON, 2012).

Geralmente, a representação serial de matrizes se faz representando uma linha por vez, ou representando uma coluna por vez . No caso do da matrizes de coeficientes do DCT, é utilizada uma ordem modificada em que valores mais próximos do canto superior esquerdo sejam tendem a ser representados primeiro, esta ordem utiliza ziguezague de diagonais. O primeiro elemento a ser representado é o canto superior esquerdo, em seguida é representado o elemento a direita e desce na diagonal. Ao fim de uma descida pela diagonal, é representado uma linha e em seguida subindo pela diagonal. Este processo de subida pela diagonal seguida de descida pela diagonal se repete até que o último elemento seja representado, que fica no canto inferior direito. A Figura 8 representa a ordem em que os elementos são serializados numa matriz 8x8.

Esta ordenação tem como objetivo concentrar na mesma região valores não nulos consecutivos, e concentrar para o fim da sequência os valores nulos. Com isso, é utilizado uma versão modificada do RLE para representar de forma compacta zeros consecutivos na sequência. Cada elemento é representado por um par, com o primeiro valor indicando quantos zeros existem antes do elemento, e o segundo valor o

Figura 8 – Ordem em que os elementos de uma matriz 8x8 são serializados

(a) Ordem por linha

1º	2º	3º	4º	5º	6º	7º	8º
9º	10º	11º	12º	13º	14º	15º	16º
17º	18º	19º	20º	21º	22º	23º	24º
25º	26º	27º	28º	29º	30º	31º	32º
33º	34º	35º	36º	37º	38º	39º	40º
41º	42º	43º	44º	45º	46º	47º	48º
49º	50º	51º	52º	53º	54º	55º	56º
57º	58º	59º	60º	61º	62º	63º	64º

(b) Ordem por coluna

1º	9º	17º	25º	33º	41º	49º	57º
2º	10º	18º	26º	34º	42º	50º	58º
3º	11º	19º	27º	35º	43º	51º	59º
4º	12º	20º	28º	36º	44º	52º	60º
5º	13º	21º	29º	37º	45º	53º	61º
6º	14º	22º	30º	38º	46º	54º	62º
7º	15º	23º	31º	39º	47º	55º	63º
8º	16º	24º	32º	40º	48º	56º	64º

(c) Ordem zigue-zague DCT no JPEG

1º	2º	6º	7º	15º	16º	28º	29º
3º	5º	8º	14º	17º	27º	30º	43º
4º	9º	13º	18º	26º	31º	42º	44º
10º	12º	19º	25º	32º	41º	45º	54º
11º	20º	24º	33º	40º	46º	53º	55º
21º	23º	34º	39º	47º	52º	56º	61º
22º	35º	38º	48º	51º	57º	60º	62º
36º	37º	49º	50º	58º	59º	63º	64º

elemento em si. No momento em que a sequência não tem mais valores se não zeros, é afixado o valor *End-of-Block* que indica para o decodificador que o restante daquele bloco tem valor zero.

O primeiro valor do bloco de coeficientes, que representa a frequência nula, é representado de uma forma excepcional. No primeiro bloco da imagem ele é representado diretamente, nos blocos seguintes, este valor é representado como a diferença da frequência nula do bloco anterior para o bloco atual. Este é o único momento em que o algoritmo do JPEG tenta fazer predição de um valor.

Finalmente, o bloco é codificado utilizando Huffman para diminuir mais ainda o tamanho necessário para a representação de cada bloco. Enquanto que a decodificação consiste de aplicar o inverso de cada passo na ordem inversa, é feita a decodificação de Huffman, os zeros do bloco são preenchidos de acordo com o RLE, a de-quantização é aplicada e o IDCT é calculado para recuperar o bloco original.

MOTION JPEG

Motion JPEG consiste de um vídeo codificado como uma sequência de quadrados codificados cada um independente do outro como

uma imagem no formato JPEG. As vantagens desse formato é que é possível acessar e até mesmo modificar um quadro sem interferência de nenhum outro quadro, porém, nenhuma redundância entre imagens é utilizada para melhor a compactação, tornando o arquivo final maior do que outros codificadores de vídeo. Quando um dispositivo já tem suporte em *hardware* para codificar JPEG é possível utiliza-lo para criar vídeos em *Motion* JPEG sem necessidade de *hardware* extra ou algoritmos de codificação de vídeo rodando em *software*.

3.2.3 MPEG-2

MPEG-2 Part 2 é o padrão de codificação de vídeo desenvolvido conjuntamente pelo *ITU-T Video Coding Experts Group* e *ISO/IEC Moving Picture Experts Group* (ITU, 2000). O padrão conta com recursos comparáveis ao JPEG e com compactação inter-quadros, isto é, que utiliza informação de redundância de outros quadros para codificar o quadro atual.

O vídeo MPEG possui o conceito de tipo de figura. O tipo *I* (*Intra*) consiste de uma imagem que é codificada independentemente de outros quadros como no JPEG. O tipo *P* (*Predicted*) consiste de um quadro que utiliza o quadro anterior como forma de fazer predições, apenas as diferenças são codificadas. Além disso, existe figuras do tipo *B* (*Bidirectional*) que utilizam tanto o quadro anterior quanto o quadro seguinte para fazer predição. Como figuras do tipo *B* utilizam o quadro futuro para fazer sua predição, elas não podem ser utilizadas como fonte de predição, quando o decodificador desenha uma figura do tipo *B* ela é descartada logo em seguida.

As figuras do tipo *I* definem o inicio de cada Grupo de Figuras (*Group of Pictures, GoP*), e o fim do grupo anterior. Os grupos são independentes entre si, enquanto que dentro de um grupo, toda imagem além da primeira depende de alguma outra. Desta forma, para editar um vídeo MPEG-2*,* é preciso codificar de novo todo o grupo aonde houve modificações. A sequência de imagens dentro do *GoP* fica a critério do codificador, não há obrigações de tamanho ou quais tipos de figura devem ser usados. Um vídeo que contém apenas imagens do tipo *I,* grupos de tamanho unitário, é essencialmente igual o *Motion*-JPEG.

O decodificador do MPEG-2 também calcula compensação de movimento. Grande parte das mudanças de cores entre quadros de um vídeo se dão pelo movimento de *pixels*. A primeira etapa da predição de um quadro é aplicar vetores de movimentos fornecidos pelo codificador, com o movimento compensado, a diferença entre a imagem predita e a imagem real deve ser codificada. A imagem de resíduos, isto é, diferença da predição para a imagem real, é compactada da mesma forma que uma figura-P.

Como o processo de codificação do vídeo gera erros na imagem resultante, o codificador precisa decodificar a imagem para garantir que usará como referência de codificação de imagens futuras a mesma imagem que o decodificador receberá, evitando assim que os erros se acumulem.

Na verdade, o padrão define apenas o decodificador. Qualquer fluxo de bytes que faça sentido para o decodificador MPEG-2 é dito como um vídeo válido. O codificador tem flexibilidade para criar esse fluxo válido de qualquer forma, por exemplo, para aplicações de tempo real o codificador pode evitar perder tempo calculando a melhor combinação de parâmetros enquanto que a codificação para distribuição pode tentar várias técnicas diferentes até que acha o menor tamanho possível.

3.2.4 H.264

Também denominado como *MPEG-4 Part 10* ou *Advanced Video Coding* (AVC), foi desenvolvido pela *Joint Video Team* (JVT, 2003). O H.264 é similar ao MPEG-2, porém os perfis de decodificação contam com uma variedade maior de ferramentas e flexibilidade a disposição do codificador. Poynton (2012) usa a expressão “fruto baixo” para explicar que várias técnicas simples foram implementadas para em conjunto alcançar até o dobro da eficiência do *MPEG-2*. É um dos codificadores utilizados no WebRTC.

3.2.5 VP8

O VP8 consiste de um esforço da Google para fornecer uma alternativa ao H.264 de código aberto e sem custos para o uso, isto é, sem o problema de patentes. A intenção é que o formato seja livre para ser incorporado em navegadores para evoluir o suporte a vídeo, atualmente também é utilizado no WebRTC. O projeto consiste do padrão do decodificador (BANKOSKI et al., 2011), e de uma biblioteca de implementação de referência chamada *libvpx*.

4 PROTOCOLOS DE COMUNICAÇÃO EM TEMPO REAL

Este capítulo apresenta como os protocolos em redes são utilizados para formar um sistema de comunicação multimídia em tempo real. Novamente, a descrição se dá de forma de baixo para cima, primeiro serão apresentados os protocolos Internet (UDP e TCP), os protocolos de comunicação em tempo real, e por fim o WebRTC e os esforços necessários para implantar vídeo conferência em *web*.

O projeto de uma rede de computadores pode ser visto como uma arquitetura baseada em camadas onde cada camada se preocupa com uma questão específica e fornece funcionalidades para a camada superior (ISO/IEC, 1994). O protocolo de Internet (*Internet Protocol* - IP) trabalha com quatro camadas, sendo que a última camada é a própria aplicação (BRADEN et al., 1989; BRADEN, 1989). Uma aplicação tem dois protocolos de transporte a sua disposição, o *Transmission Control Protocol* (TCP) *(POSTEL, 1981)* que é baseado em conexão e em fluxo de dados, e o *User Datagram Protocol* (UDP) *(POSTEL, 1980)* que é baseado em envio de *datagramas* sem o estabelecimento de conexões.

Uma conexão TCP estabelece um túnel bidirecional com garantia de entrega, integridade, ordem dos dados, e controle de velocidade. Isto é, a aplicação faz o envio de dados sem nenhum tipo de separação de mensagens, a ordem em que os dados são escritos é a ordem em que os dados são entregues na aplicação destino. Tudo que é escrito chega no destino, e a aplicação não consegue enviar mais dados do que a rede consegue transmitir.

Graças a sua natureza confiável, o TCP é utilizado na maioria das aplicações. Esta a confiabilidade do TCP vem a um custo. Por exemplo, para garantir que todo o dado escrito chegue no emissor, o TCP precisa reenviar pacotes perdidos, para garantir ordem, o receptor tem que guardar pacotes adiantados em um *buffer* para que só sejam entregues quando os pacotes anteriores cheguem.

Há um conjunto de aplicações em que essa confiança não é necessária e que o esses custos são nocivos. O melhor exemplo desse tipo de aplicação é a comunicação multimídia em tempo real. Dados multimídias são propriamente toleráveis, não há a necessidade de

transmissão e/ou representação exata da informação. Por exemplo, dado uma comunicação por vídeo conferência que tenha uma taxa de 25 quadros por segundo, cada quadro fica visível para o usuário por 40 milissegundos. Se houver o descarte de um desses quadros, o decodificador que está rodando no receptor pode estender a exibição do quadro anterior por mais 40 milissegundos. A falha de comunicação no vídeo se torna imperceptível ou ao menos não afetará a experiência de usuário. É importante notar que é possível que a decodificação de um quadro dependa de informações do dado anterior, então além do atraso, a perda de um pacote afeta a reconstrução dos pacotes seguintes.

Em aplicações de tempo real, é preferível manter o fluxo de dados ativo e com *delay* mínimo a atrasar o envio de pacotes mais recentes para garantir a entrega de um pacote antigo. Em uma aplicação de *voz-sobre-ip*, atrasos para que a voz chegue no outro lado atrapalham mais a conversação do que uma distorção momentânea do áudio. É possível reconhecer palavras faladas mesmo que fragmentos do som estejam faltando.

Como a reprodução do dado ocorre em tempo real, a sincronização da mídia faz com que cada pacote tenha um momento certo em que deve ser reproduzido. Esse momento certo é o mais cedo possível, porém com alguma folga para compensar o fato de que o tempo que um pacote leva para chegar no destinatário não é constante. Em rede de computadores, é comum chamar de *jitter* a variação do atraso de transmissão de um pacote, porém é um termo ambíguo, a expressão variação do atraso é mais recomendada (STEPHAN, 2005). Se a variação do atraso não for levada em consideração corretamente, vários pacotes seriam descartados pelo simples fato de ter sofrido um atraso maior do que a média naquela comunicação.

É empregado um *buffer* com um parâmetro que define um atraso fixo e constante em todos os pacotes recebidos, como consequência, a mídia pode ser reproduzida de forma contínua. Idealmente, esse parâmetro é grande o suficiente de forma que todo pacotes recebidos levem menos tempo para chegar do que o estipulado, porém não pode ser muito grande de forma a criar atrasos desnecessários na reprodução da mídia. Portanto, se uma sincronização rígida na reprodução da mídia for aplicada, o reenvio de pacotes torna-se inútil pois na maioria dos casos, quando o pacote reenviado chegar no receptor ele será

automaticamente descartado por estar mais atrasado do que o valor máximo tolerado.

O UDP é uma alternativa mais leve do ponto de vista de recursos. Ao invés de estabelecimento de conexão entre dois usuários, é feita apenas o envio de um datagrama a partir de um emissor (endereço IP e número da porta) para um receptor (endereço IP e número da porta). A integridade da mensagem dentro do datagrama é garantida através de um *checksum*, porém a aplicação não pode contar com a ordem em que os datagramas chegam no emissor e nem se eles de fato chegarão.

Como a lógica do UDP é simples, ele se torna muito flexível. A aplicação pode construir lógicas de confiabilidade a medida que for necessário. Se for tomado como exemplo uma aplicação que transfere arquivos, quebra o arquivo em vários segmentos e cada pedaço é enviado através de um datagrama contendo indicação da posição deste segmento dentro do arquivo. O receptor aloca previamente o espaço para o arquivo a ser recebido e preenche esse espaço com os segmentos recebidos. Esse é um exemplo de aplicação que precisa que os pacotes garantidamente cheguem em seu destinatário, porém não faz diferença em qual ordem os pacotes chegam, no final da transferência, o arquivo vai ser reconstruído de mesma forma.

A segurança, isto é, autenticidade, integridade e privacidade, pode ser alcançada utilizando o *Transport Layer Security* (TLS) (DIERKS; RESCORLA, 2008). O protocolo funciona utilizando certificado digital para identificar as partes e criptografia simétrica para cifrar os dados transferidos utilizando uma nova chave para cada conexão. O TLS depende de uma maneira de troca de fluxo de bytes confiável. Uma versão alternativa do protocolo, *Datagram Transport Layer Security* (DTLS) fornece a mesma segurança para comunicações baseadas em datagrama.

Para a comunicação multimídia, áudio e vídeo, existe o protocolo *Real-time Transport Protocol* (RTP) junto com o protocolo de controle *Real Time Control Protocol* (RTCP) (SCHULZRINNE et al., 2003). Tipicamente, o UDP é utilizada embaixo do RTP mas qualquer meio de comunicação pode ser utilizado com o RTP, e da mesma forma, embora o protocolo contém informação de ordenação dos pacotes, cabe a camada acima decidir se irá ordenar a informação ou não. O RTP possui perfis que definem extensões ao cabeçalho, um

deles é o *Secure Real-time Transport Protocol* (SRTP) em conjunto com o *Secure Real Time Control Protocol* (SRTCP) (BAUGHER et al., 2004).

4.1 TELECOMUNICAÇÃO BASEADA EM *WEB*

Nesta seção, são descritos os desafios para estabelecer telecomunicação (vídeo e áudio) inteiramente baseada em *web*. Foi escolhido o WebRTC como tecnologia base para vídeo conferência, os principais motivos são o suporte embutido nativamente em navegadores *web*, e o suporte a codificadores avançados de vídeo (VP8 e H264). Também é considerada a questão de conferência com mais de duas pessoas, que é um desafio pois a quantidade de recursos utilizados cresce quadraticamente em função da quantidade de pessoas na sala.

Web Real-Time Communication é um padrão ainda sob desenvolvimento que objetiva comunicação em tempo real dentro do navegador *web* (BURNETT et al., 2013; WEBRTC, 2013). O padrão é relativamente novo, sendo que o primeiro rascunho foi publicado pelo *World Wide Web Consortium* (W3C) em 27 de outubro de 2011. O primeiro navegador a adicionar suporte ao WebRTC foi o Google Chrome 23 em versão *beta*, sendo que em novembro de 2013, foi lançada a versão 23 estável. O navegador Mozilla Firefox na versão 22, em junho de 2013, adicionou o seu primeiro suporte estável ao WebRTC (MOZILLA, 2013). Finalmente, em novembro de 2013, o navegador Opera foi disponibilizado na versão 18 com suporte estável ao WebRTC (OPERA, 2013).

O WebRTC está sendo projetado visando tanto comunicação em tempo real de dados genéricos quanto comunicação via vídeo e voz, sendo que a base de código do projeto possui codificadores de vídeo e áudio modernos livres de *royalties*. Todo o código é disponibilizado através da licença de *software* BSD (CALIFORNIA, 1998). O desenvolvedor tem acesso ao WebRTC utilizando uma API em *JavaScript*. A API também conta com funções para obter a *webcam* e o microfone do usuário. Para a reprodução da mídia, são utilizados elementos do HTML5.

WebSocket (HICKSON, 2012; FETTE; MELNIKOV, 2011) é

um padrão mais maduro que fornece comunicação *full duplex*, confiável, ordenada e livre de erros com o servidor *web*. O WebRTC pode ser visto como uma tecnologia que complementa o *WebSocket*, enquanto que o *WebSocket* faz comunicação baseada em cliente-servidor e utilizando o protocolo TCP, o WebRTC tem o enfoque de comunicação em tempo real de usuário para usuário, *peer to peer* (P2P) utilizando os protocolos descritos na seção anterior. O SRTP é utilizado para a transmissão da mídia, o DTLS faz a troca de chaves do SRTP de forma segura, ambos utilizando o UDP como meio de transporte. Porém, uma aplicação de vídeo conferência sempre terá um componente de controle da chamada que comumente é feito utilizando WebSocket (ou outra alternativa de comunicação confiável).

O esforço da comunidade de desenvolvedores de navegadores *web* para criar o WebRTC também se preocupou com os codificadores (*codec*) de áudio e vídeo. Segundo a página oficial do projeto, os *codecs* de áudio que atualmente fazem parte do projeto são G.711, G.722, iLBC e iSAC, enquanto que o suporte a codificação de vídeo é feito através do VP8. Todos são livres de *royalties* ao serem utilizado junto com o WebRTC, que por sua vez tem o código fonte livre para ser utilizado e modificado sem restrições.

4.1.1 *Network Address Translation*

Nesse contexto de conexão usuário para usuário, o primeiro aspecto que a implementação do WebRTC deve considerar é a conectividade, pois a questão é mais complexa do que conexão com um servidor. Como o usuário pode possuir algum *software* de segurança (filtro de pacotes) ou pode ter seu dispositivo escondido atrás de um sistema de *Network Address Translation* (NAT) (EGEVANG; FRANCIS, 1994), a conectividade pode ser limitada ou estar totalmente bloqueada.

Uma das formas de aumentar a segurança do usuário em Internet é bloquear qualquer abertura de conexão iniciada por um dispositivo externo - enquanto que se for o próprio sistema que inicializou a conexão ela é liberada. Por exemplo, para ler e-mail, o computador do usuário requisita uma conexão com um servidor de e-mail (uma conexão inicializada pela própria máquina), enquanto que se algum

software malicioso abrir uma porta de conexão para atacantes acessarem o computador, esta seria bloqueada. A ideia por trás desse tipo de bloqueio é que, para o usuário comum, o uso da rede de computadores é feito sempre através de requisições iniciadas pelo próprio usuário, nunca uma conexão iniciada por um sistema externo.

Quando uma aplicação se caracteriza em conexões em forma de P2P, isto se torna um problema, pois obrigatoriamente um dos usuários terá que receber uma conexão externa. Com um filtro de pacotes, caberá ao usuário (ou administrador da rede) configurá-lo para permitir as devidas conexões, não há nada que a aplicação possa fazer uma vez que o objetivo do filtro de pacotes é manter a segurança, isto é, se aplicações pudessem ultrapassar o filtro ele não estaria servindo seu propósito.

Já o NAT[1] foi criado com o objetivo de resolver o problema do crescente uso de endereços IP que levaria a uma exaustação dos endereços disponíveis (HUSTON, 2004) - o protocolo de Internet na versão 4 (IPv4) utiliza endereços de tamanho fixo de 4 *bytes*, o que contabiliza aproximadamente 4.3 bilhões de endereços disponíveis. A solução consiste de mascarar vários dispositivos numa rede privada (REKHTER et al., 1996) com o acesso a Internet através de um *gateway* operando o protocolo NAT de forma que apenas um endereço do protocolo de Internet é alocado.

Comunicações acontecendo dentro da rede privada seguem os protocolos de Internet normalmente, porém para que os vários dispositivos possam fazer acessar dispositivos externos utilizando um único endereço precisa-se fazer uma tradução de endereço externo (endereço IP global e porta no *gateway*) para um endereço interno (endereço IP na rede privada e porta no dispositivo). Quando a comunicação é iniciada por um dispositivo da rede privada, o *gateway* irá alocar uma porta para representar aquele dispositivo específico e irá redirecionar todo pacote recebido nesta porta para o dispositivo, mas quando tenta-se iniciar a comunicação externamente, o *gateway* não saberá qual dispositivo na rede privada deverá receber o pacote.

Novamente, o comportamento de um usuário comum (ler e es-

[1] alguns autores diferenciam NAT e NAPT (*Network Address and Port Translation*). As primeiras implementações de NAT lidavam apenas com tradução a nível de endereço IP, enquanto que o NAPT lidava tanto com o endereço IP quanto com a porta de conexão utilizada. Hoje é comum utilizar o termo NAT para ambas caracterizações do sistema.

crever e-mail, navegar na Internet, entre outros) não é afetada pela camada NAT, inclusive algum nível de segurança é oferecido uma vez que não é possível abrir uma porta de conexão nos dispositivos da rede privada. Como o sistema NAT foi desenvolvido com a intenção de ser transparente, não espera-se que o usuário configure seu *gateway* para expor sua máquina externamente, embora seja possível através de mapeamento estático de portas *(port forwarding*). Além disso, implementações NAT podem causar outras complicações, como demonstra o RFC 4787 (AUDET; JENNINGS, 2007).

É importante notar, que a disseminação do NAT se deu principalmente pelo fato de que nenhuma mudança na estrutura da Internet é necessária, o NAT não define um protocolo novo e nem modifica o protocolo de Internet. Dessa forma, os problemas que surgiram se dão ao fato de que o NAT é construído para se encaixar transparentemente nas redes de computadores sendo invisível para os dispositivos, o protocolo da Internet e para as aplicações desenvolvidas.

No protocolo TCP, o comportamento do NAT tende a ser determinístico, para cada conexão é alocada uma porta externa (mapeada a porta e endereço IP do dispositivo na rede privada) que será utilizada na troca de dados entre o dispositivo externo e o dispositivo interno. Como o protocolo é baseado em conexão, apenas os dois dispositivos conectados usarão a porta alocada. Já o protocolo UDP se comporta diferente pois cada porta alocada pode ser utilizada para a comunicação com vários dispositivos, pois o protocolo não é baseado em conexões, mas em troca de data-gramas.

Como nunca houve um processo de padronização do NAT, inclusive não foi planejado que esta solução durasse tanto tempo, cada implementação de NAT tem um comportamento diferente, principalmente no protocolo UDP. Quando um pacote sai de um dispositivo da rede interna, tem-se: o endereço IP do dispositivo interno que iniciou a comunicação; a porta do dispositivo utilizada; a porta externa alocada para a comunicação; o endereço IP de destino; e a porta no dispositivo de destino.

As implementações NAT podem ser categorizadas de acordo com quais restrições são impostas na utilização do mapeamento criado em relação ao protocolo UDP. *Symmetric* é a forma mais restrita, quando um mapeamento UDP é criado, a porta alocada só pode ser utilizada para receber mensagens do endereço IP destino e da porta

Tabela 4 – Categorização de NATs em função da filtragem de pacotes de acordo com o endereço IP e porta de origem

Tipo	Filtragem ao Receber Pacotes	
	Endereço IP Origem	Porta Origem
symmetric	Mesmo do mapeamento	Mesmo do mapeamento
restricted-cone	Mesmo do mapeamento	Qualquer
port-restricted-cone	Qualquer	Mesmo do mapeamento
full-cone	Qualquer	Qualquer

destino. Numa comunicação P2P, se ambos usuários estiverem utilizando NAT do tipo *Symmetric* não será possível estabelecer qualquer tipo de comunicação. O *full-cone*, por sua vez, é o tipo menos restrito. Quando uma porta é alocada, este mapeamento fica disponível para receber mensagens de qualquer dispositivo externo, mesmo que não seja o dispositivo envolvido na criação do mapeamento. Também há categorização para tipos intermediários, como *restricted-cone* onde o mapeamento é restrito apenas ao endereço IP de destino (podendo este usar qualquer porta) e *port-restricted-cone* onde a restrição é apenas que a porta utilizada para enviar pacotes seja a mesma utilizada no dispositivo de destino da mensagem que criou o mapeamento.

Basicamente, essa forma de categorizar NATs leva em consideração se o endereço IP e porta destino que criaram o mapeamento no NAT são utilizados para filtrar novos pacotes que são recebidos. A Tabela 4 enumera os tipos de NAT em função de qual tipo de filtragem é aplicada ao receber novos pacotes.

Na verdade, existem vários critérios para categorizar NATs, mas para nosso problema, essas 4 categorias são o suficiente. As categorias *symmetric* e *restricted-cone* são as problemáticas pois com elas não é possível estabelecer conexão com outro usuário atrás de um NATs. Já com *full-cone* ou *port-restricted-cone* é possível, com o auxílio de um terceiro dispositivo que seja acessível, estabelecer uma conexão P2P mesmo com usuários que estão inacessíveis atrás de um NAT. A técnica consiste de utilizar o terceiro dispositivo para criar um mapeamento no NAT que será utilizada pelo segundo usuário para enviar pacotes a primeira máquina, isso funciona porque o mapeamento criado pode ser utilizado por qualquer dispositivo.

4.1.2 Ferramentas para Atravessar NAT

O servidor auxiliar utilizado para criar o mapeamento no NAT é um conjunto de métodos e protocolos padronizados chamada *Session Traversal Utilities for NAT* (STUN) (ROSENBERG et al., 2008). O objetivo dessa ferramenta é permitir que dispositivos descubram se estão atrás de um NAT e de qual tipo. O servidor STUN responde a requisições por meios configuráveis pelo cliente, quando a resposta será enviada e partir de qual endereço IP e porta. Dessa forma, o dispositivo atrás de NAT poderá fazer uma sequencia de testes utilizando respostas através de endereço IP e/ou porta diferentes para ver o quão restrito seu NAT. Além disso, o serviço STUN também informa de qual porta e endereço IP foi originada a requisição, de forma que o dispositivo possa comparar o seu endereço e porta local com o recebido pelo STUN. Por exemplo, se o endereço recebido pelo STUN for o mesmo que o dispositivo, isso significa que o dispositivo não está numa rede privada.

Uma forma básica de testar o tipo de NAT pode ser dividida em três testes. O primeiro teste apenas requisita qual porta e endereço IP o servidor STUN recebeu, para identificar a presença ou não de um NAT. Caso haja um NAT, o segundo teste requisita que a resposta seja dada através de um endereço IP diferente, para verificar se o mapeamento permite acesso a partir de outros endereços. E no terceiro teste é requisitado que uma porta diferente seja utilizada para a resposta. Ao fim dos três testes é possível verificar se o dispositivo está atrás de algum NAT e qual restrição é imposta sobre os mapeamentos criados, isto é, se o tipo do NAT é *symmetric, full-cone, restricted-cone* ou *port-restricted-cone.*

No fim das contas, o STUN é uma ferramenta que responde as requisições feitas pelo dispositivo. Quais requisições devem ser feitas para verificar o tipo de NAT e para atravessar a barreira não fazem parte do objetivo do STUN. Este serviço deve ser utilizado por outros protocolos para efetuar de fato o atravessamento de NAT, como por exemplo o protocolo *Interactive Connectivity Establishment* (ICE) (ROSENBERG, 2010) e *Session Initiation Protocol* (JENNINGS; MAHY; AUDET, 2009).

A técnica ICE em particular é a utilizada pelo padrão WebRTC.

Ela é uma evolução sobre o protocolo *Session Description Protocol* (SDP), onde a comunicação consiste apenas de uma *oferta* vinda de um dispositivo e de uma *resposta* do segundo dispositivo. Se houver alguma dificuldade na conexão, essas duas mensagens não são o suficiente para ultrapassar o problema e tentar outras maneiras de conexão. Já o protocolo ICE consiste de uma troca interativa de mensagens até que a conexão seja estabelecida.

Toda a comunicação auxiliar utilizada para estabelecer a conexão P2P acontece através de algum servidor central seguindo o modelo cliente-servidor tradicional, em que filtro de pacotes e NAT não atrapalham. É utilizado o termo *signaling* para representar esta comunicação utilizada para controlar a comunicação multimídia em si. O protocolo ICE não define que tipo de canal é utilizado para fazer o *signaling* ficando livre à aplicação escolher a forma mais adequada. Em telecomunicação em *web* é comum utilizar o servidor de *Hypertext Transfer Protocol* (HTTP) de onde é feito o *download* da página web também para a comunicação do *signaling*, utilizando o próprio protocolo HTTP via AJAX ou via *websockets*, ambas técnicas permitem comunicação com o *signaling server* utilizando recursos de programação do navegador *web*.

A abordagem do ICE consiste de levantar uma série de candidatos de conexão, endereço IP e porta, dando uma ordem de prioridade. Ambos os dispositivos fazem esse levantamento e fazem a troca de candidatos, estes são pareados respeitando a ordem de prioridade para tentar estabelecer uma comunicação. A priorização de candidatos dá preferência primeiro ao endereço informado pela interface de rede local, que no caso de uma rede privada é acessível apenas pela rede local, e no caso do dispositivo estar na Internet sem NAT já é o endereço acessível de qualquer lugar. Em seguida, são levantados os candidatos utilizando o endereço global obtido através do STUN, e por fim, se estiver disponível, são utilizados servidores auxiliar de retransmissão. Com esta ordem, a tentativa de conexão é feita primeiro com as opções mais eficientes e só no caso de falha as opções mais custosas são utilizadas, por exemplo, se os dois dispositivos estão na mesma rede privada, eles podem se conectar diretamente utilizando o endereço IP local, enquanto que a utilização de servidor de retransmissão só acontece em última instância.

4.1.3 Travessia Através de Servidor Intermediário

Servidores de retransmissão são implementados como uma extensão ao protocolo STUN chamada *Traversal Using Relays around NAT* (TURN) (MATTHEWS; ROSENBURG; MAHY, 2010). O servidor TURN, assim como qualquer servidor STUN, deve ter endereço de IP acessível de qualquer lugar, desta forma é garantido que será possível estabelecer a comunicação P2P mesmo nos casos em que o NAT é restritivo. No entanto, como o TURN agirá como um intermediário da comunicação, retransmitindo todos pacotes recebidos para a outra ponta, perde-se as vantagens de desempenho fornecidas pelo P2P, a comunicação deixa de ser direta e muitos usuários utilizando ao mesmo sobrecarrega o servidor. Por isso, a solução TURN é utilizada somente em última instância ou as vezes nem está disponível.

O padrão WebRTC utiliza o protocolo ICE, junto com o serviço STUN e o desenvolvedor da aplicação pode configurar algum servidor TURN para ser utilizado. Como o serviço STUN é leve, não demanda muitos requisitos computacionais e de banda de rede, existem servidores disponíveis na Internet para uso de qualquer aplicação. Já o TURN *server*, como demanda muito recursos de rede, é impraticável a disponibilização de servidores para o público em geral. É comum que a implantação do TURN fique restrita ao ambiente da aplicação desenvolvida.

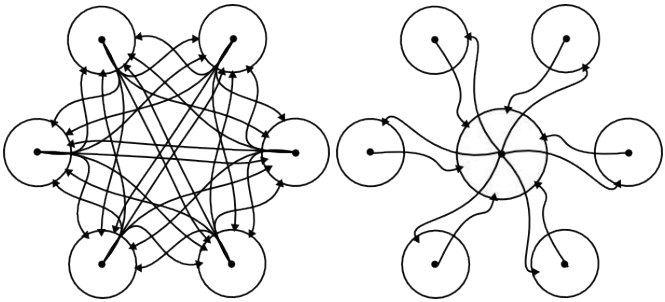
4.1.4 Sala de Conferência

Sala de Conferência consiste de uma vídeo chamada que envolve mais do que dois participantes, uma sala de reunião virtual. O desafio de uma vídeo chamada com múltiplos participantes é o uso de recursos de rede que cresce quadraticamente. As duas abordagens mais simples para enfrentar esse problema é a utilização de servidor centralizado ou conexão todos-para-todos. A Figura 9 ilustra essas duas topologias.

Conexão todos-para-todos consiste de toda combinação de par de usuários estabelecer uma conexão 1 para 1 entre si, desta forma consegue-se efetuar uma conferência com a vantagem de utilizar ape-

Figura 9 – Topologias de conexão para vídeo conferência com múltiplos usuários

(a) Conexão todos-com-todos (b) Conexão com servidor central



nas recursos dos usuários, sem a necessidade de nenhum servidor. A grande desvantagem é a grande quantidade de conexões que cada usuário tem que fazer, em uma conferência com n usuários – contabilizando conexão de transmissão separada da conexão de recebimento – cada usuário fará $2 \cdot (n-1)$ conexões e a quantidade total de conexões é de $n \cdot (n-1)$. Além disso, a mesma informação é replicada para cada conexão. Embora, do ponto de vista do desenvolvedor da aplicação, seja uma solução de baixo custo, é uma abordagem que consome muitos recursos de banda da rede como um todo e do usuário, sendo possível que uma conexão doméstica não seja o suficiente para manter tantos dados sendo transmitidos ao mesmo tempo.

A utilização de servidor central tira proveito da redundância de informação sendo transferida para reduzir a quantidade de conexões. Cada usuário precisará fazer apenas uma conexão ao servidor central para transmissão e recebimento do áudio e vídeo da vídeo conferência. Numa conferência com n participantes, a conexão de cada usuário terá a carga de n fluxos ($n-1$ recebimentos e 1 transmissão). A complexidade se concentra no servidor central, que por sua vez receberá n fluxos e transmitirá $n \cdot (n-1)$ fluxos. A desvantagem dessa abordagem, é que para aplicações que tem muito usuários (mesmo não estando na mesma conferência) a quantidade de banda concentrada em um único ponto torna a abordagem inviável, além disso, a abordagem

obriga toda a comunicação a passar pelo servidor central, desconsiderando afinidade entre usuários. Por exemplo, se dois usuários estão na mesma rede local, é mais vantajoso que a transmissão seja direta entre eles do que passando por um servidor localizado em uma rede externa.

Existem topologias de conexão mais inteligentes que tentam utilizar o melhor de cada abordagem, mas elas envolvem retransmissão (transmitir um fluxo recebido de um usuário para outro usuário) que ainda não é suportado no padrão WebRTC, portanto ainda não é possível utilizar tal abordagem.

5 AMBIENTE

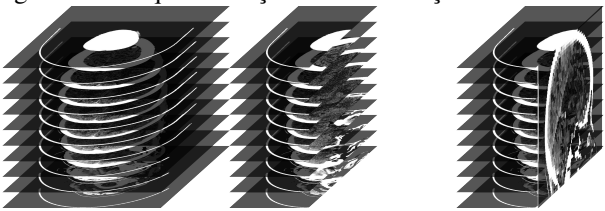
A aplicação segue uma abordagem modular utilizando o padrão HTML5. Ela é focada no gerenciamento de imagens médicas, no apoio a decisão no campo de doenças buco-maxilofacial, e em objetivos educacionais. A aplicação é capaz de carregar, visualizar, manipular e fazer marcações simples numa série de imagens de propósito diagnóstico (imagens clínicas, radiológicas e histopatológicas). O módulo de visualização também possui funcionalidades de reconstrução 3D através de Reconstrução Multi-planar, Reconstrução Panorâmica e Reconstrução Oblíqua e funcionalidades de visualização de imagens grandes sob demanda. Para suportar colaboratividade, todas as ações de um usuário são propagadas para os outros usuários visualizando o mesmo exame, além disso, os usuários podem ativar o módulo de vídeo chamada para comunicação em tempo real.

Todos recursos são montados em cima do módulo primário, que implementa funcionalidade PACS. Como navegadores *web* não podem abrir arquivos DICOM diretamente, esse módulo serve uma cópia especial do arquivo DICOM em um subformato de 16-bits em tons de cinza do *Portable Network Graphics* (PNG). Quando o cliente *web* requisita uma imagem, a versão PNG, que usa uma tabela de cores especial de 16 bits em tons de cinza que reflete os valores de densidades radiológica (HU) da imagem original. Imagens que não são do padrão DICOM também podem ser importadas para o PACS: O usuário fornece os metadados desejados e usa ferramentas de medição para dar informação de escala para os *pixels* da imagem importada.

5.1 FERRAMENTAS DE VISUALIZAÇÃO

Para tornar os módulos de visualização e manipulação de imagens médicas mais amigável para o usuário, todas ferramentas são manuseadas através de gestos do mouse. As ferramentas são categorizadas em ferramentas de visualização, que mudam aspectos visuais da imagem, e ferramentas de desenho/mensuração, que marcam a imagem e mostram a medida da forma desenhada. A Tabela 5 enumera

Figura 10 – Esquematização da reconstrução MPR no eixo sagital



todas as ferramentas disponíveis do módulo de visualização.

5.1.1 Reconstruções Tridimensionais

O modulo de visualização suporta reconstruções tridimensionais sobre a série de imagens médicas. O primeiro tipo de construção suportada é a Reconstrução Multi-planar (MPR), consiste de disponibilizar imagens nos três eixos, axial, coronal e sagital, a partir da imagem original, que geralmente é uma imagem axial. A Figura 12 demonstra uma captura de tela da reconstrução multi-planar em progresso. Para contornar a limitação de que o desenho tem que ser feito apenas em duas dimensões, é feito um mapeamento de *pixels.* A sequência de imagens é empilhada para criar a dimensão z que representa qual imagem da sequência é utilizada, desta forma, uma posição dentro do volume pode ser representada como (x,y,z). Uma imagem nova $(x',y')_s$ – sendo s o corte atual que o usuário selecionou – é reconstruída através do mapeamento da Equação 5.1para reconstrução axial, Equação 5.2 para reconstrução coronal e Equação 5.3 para reconstrução sagital. A Figura 10 demonstra esse processo para a reconstrução sagital.

$$f(x',y')_s = (x',y',s) \tag{5.1}$$
$$f(x',y')_s = (x',s,y') \tag{5.2}$$
$$f(x',y')_s = (s,x',y') \tag{5.3}$$

Também há ferramenta para reconstrução panorâmica e reconstrução oblíqua. Com a reconstrução panorâmica é possível ver uma

Tabela 5 – Ferramentas disponíveis no módulo de visualização de imagens

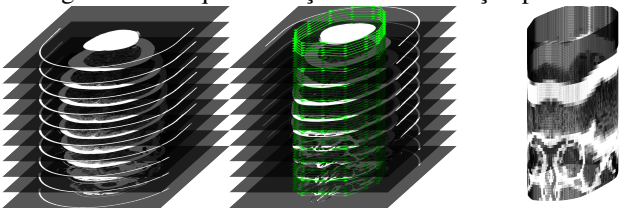
(a) Ferramentas de Desenho e Mensuração

Forma	Medição
Flecha	Comprimento
Cruz	Valor HU na posição
Linha	Comprimento
Arco	Ângulo
Retângulo	Área
Elipse	Área
Polígono	Área

(b) Ferramentas de Visualização

Nome	Descrição
Lupa	Amplia a região da imagem embaixo do cursor
Janelamento	Configura qual intervalo da escala de *Hounsfield* (HU) será visível sob os 256 tons de cinzas da cor do *pixel*, arraste vertical com o cursor muda o centro e arraste horizontal muda a largura
Zoom	Muda o *zoom* da imagem inteira
Pan	Move a imagem para tornar outras porções visíveis
Rolagem	Muda qual fatia, se mais de uma disponível, da série de imagens será mostrada
MPR	Reconstrói o volume através de Reconstrução Multi-Planar
no-MPR	Esconde a Reconstrução Multi-Planar
Resetar	Reinicia a visão para o estado inicial, os desenhos também são apagados
Panoramic	Inicia a reconstrução panorâmica, essa ferramenta é utilizada para desenhar a superfície panorâmica

Figura 11 – Esquematização da reconstrução panorâmica



superfície curva dentro da imagem médica como uma única imagem bidimensional, por exemplo a arcada dentária inteira. O processo consiste de desenhar uma curva sobre o plano axial, que o usuário define através de pontos de controle que o *software* interpola para formar uma curva suave. A curva é definida como $p(t) = (x, y)$, e é feito o mapeamento de *pixels* utilizando Equação 5.4 de forma que ao longo do eixo x da imagem gerada percorre-se pela curva, a Figura esquematiza esse processo. A reconstrução oblíqua envolve gerar várias imagens ao longo de linhas perpendiculares à curva panorâmica. A Figura 13 mostra essas reconstruções.

$$f(x', y') = \left(p\left(x'\right)_x, p\left(x'\right)_y, y' \right) \tag{5.4}$$

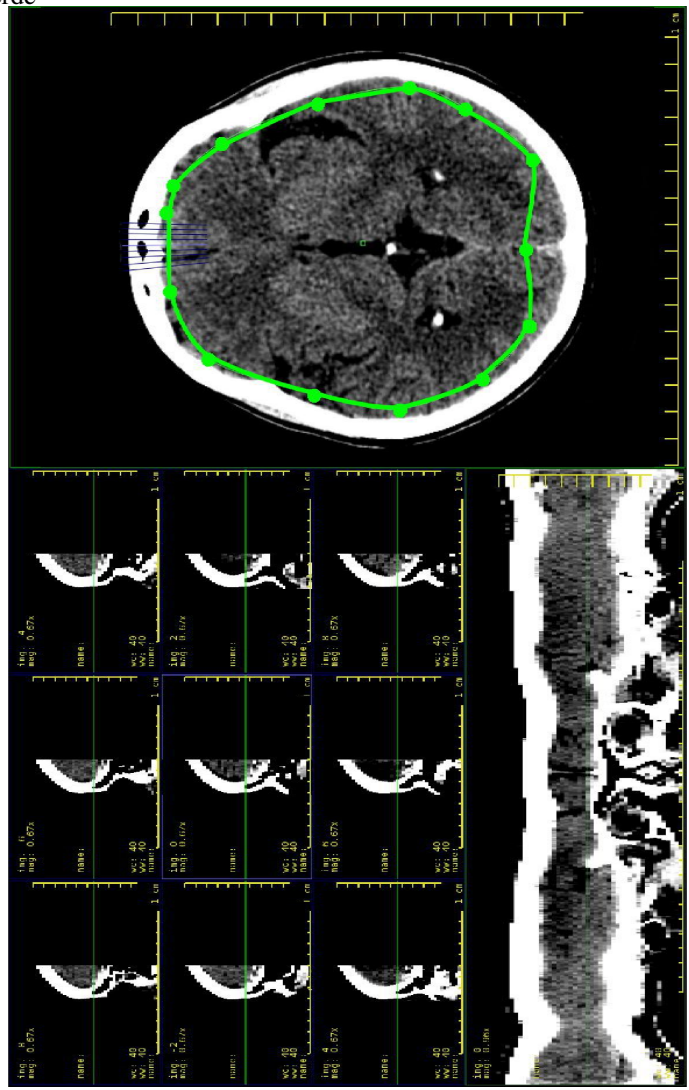
5.1.2 Visualização Sob Demanda de Imagens Grandes

O processo de diagnóstico de histopatologia depende da análise de amostras de lâminas de microscópio, onde as análises são feitas através de microscópio óptico. Usando um sistema digitalizador de lâminas, a lâmina de microscópio inteira pode ser digitalizada e enviada ao PACS, então o processo de análise pode ser feito a partir da aplicação *web*. Como geralmente a imagem final é muito grande, de forma que é inviável o *download* para todo usuário que tenta acessar a imagem, uma abordagem baseada em *tiles* é utilizada para minimizar o tempo esperado para a visualização.

O processamento do módulo de visualização de imagens grandes é feito de forma *offline*, isto quer dizer, o processamento é feito

Figura 12 – Reconstrução multi-planar, na esquerda o eixo axial, que é a imagem no formato original, e na direita as reconstruções no eixo coronal em cima e no eixo sagital em baixo

Figura 13 – Ferramenta de Reconstrução Panorâmica e Reconstrução Oblíqua, no lado direito um corte axial da série de tomografias, no lado esquerdo superior a reconstrução oblíqua na parte anterior do crânio, no lado esquerdo inferior a reconstrução panorâmica seguindo a curva verde



uma única vez antes do processo de visualização. Ao receber uma nova imagem histopatológica, o servidor criará uma versão em formato piramidal constituída por *tiles* de tamanho fixo. Cada nível da pirâmide contém a imagem inteira, o que faz com que níveis maiores da pirâmide tenham melhor resolução é que eles possuem uma quantidade maior de *tiles.*

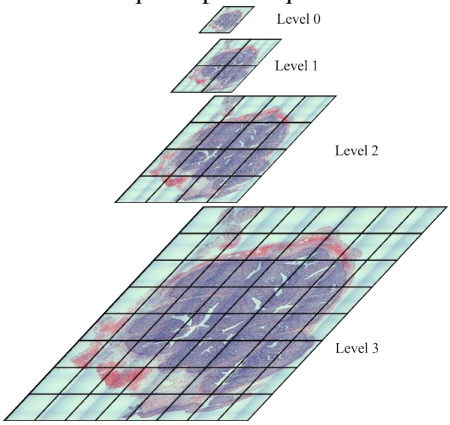
Iniciando no nível zero que contém apenas um *tile*. O processo de construção do restante da pirâmide é criar um nível novo subdividindo um *tile* em quatro novos *tiles*, de forma que cada nível novo duplica a resolução horizontal e a resolução vertical. O processo termina quando chega-se no nível que tem a mesma resolução que a imagem original, o tamanho do *tile* é ajustado para que o último nível seja exatamente o tamanho da imagem original. Cada nível divide cada *tile* do nível anterior em quatro novos *tiles*, portanto aumentando a resolução quatro vezes. A Figura 14 ilustra a representação em forma de pirâmide.

Levando em consideração a quantidade de *pixels* na imagem, o tamanho da pirâmide formada nunca será maior que a série $\sum_{i=0}^{\infty}\left(\frac{1}{4^i}\right) \cdot K = \frac{4K}{3}$, sendo K o tamanho da imagem original, isto é, um terço de aumento no tamanho da imagem. Na prática, esse valor pode chegar até 50% pois cada um dos *tiles* gerados será armazenado em um arquivo separado, e isso diminui desempenho do algoritmo de compactação de imagem em comparação a salvar apenas uma imagem inteira.

Uma vez que a pirâmide foi gerada, mais nenhum cálculo no lado servidor será feito, toda visualização é feita com código no lado cliente, enquanto que o servidor é responsável apenas por fornecer as imagens. O cliente implementa uma heurística que tenta minimizar o tempo de espera enquanto maximiza a qualidade da imagem. Esta heurística foi escolhida após experimentações exaustivas com conexões com largura de banda limitada. A Figura 15 contém uma captura de tela no exato momento em que as imagens de alta qualidade são baixadas do servidor, a parte borrada representa região da imagem que está sendo desenhada com *tiles* de níveis inferiores.

Quando o usuário navega para uma sub região da imagem, *tiles* de três níveis da pirâmide serão obtidos, o nível 0 sempre é desenhado no plano de fundo, para garantir que pelo menos algo é visível, o nível que tem resolução adequada para o zoom escolhido, e o nível anterior

Figura 14 – Pirâmide em *tiles* para o módulo de visualização de imagens grandes. Cada nível quadruplica a quantidade de *tiles*.

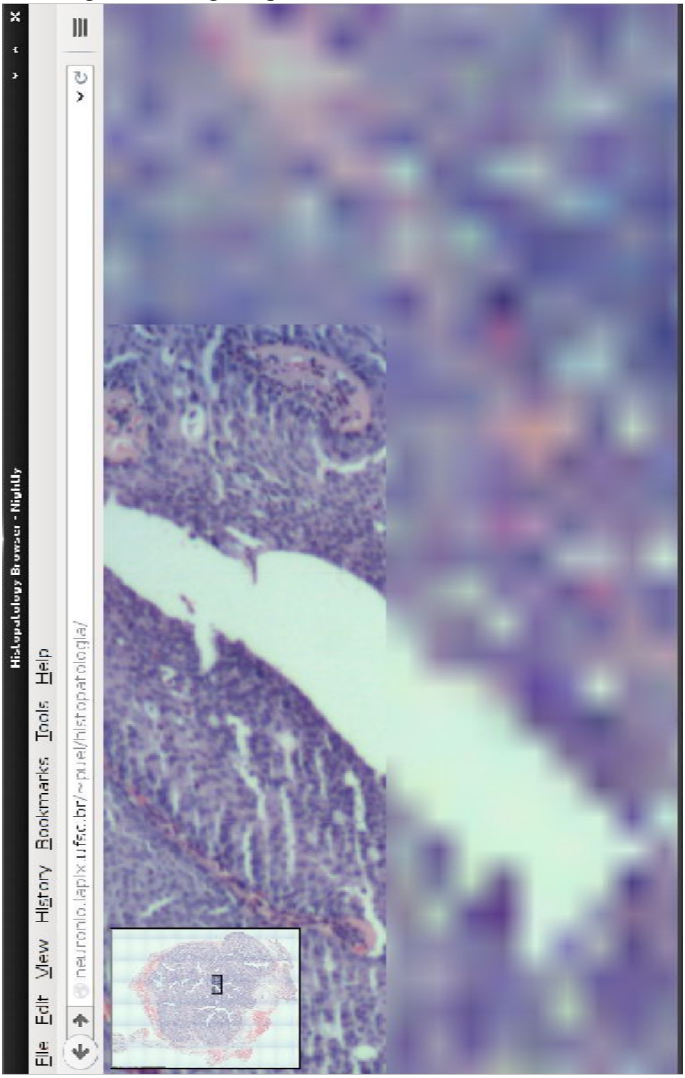


a este, sendo que apenas *tiles* na área visível são obtidos. Os *tiles* são obtidos em uma ordem que prefere níveis menores primeiro para tentar rapidamente fornecer a imagem em qualidade boa, para só então baixar a imagem em qualidade máxima. Qualquer *tile* que já foi obtido durante a navegação ficará salvo, como não há custos para visualizar este *tile*, ele sempre é desenhado quando esta na região visível.

5.2 COLABORAÇÃO ENTRE USUÁRIOS

Além de visualização, foi focado em diagnóstico remoto e colaborativo utilizando recursos multimídia. Se dois ou mais usuários estão vendo o mesmo conjunto de imagens, eles iniciam uma sessão colaborativa e uma caixa de bate papo é exibida para comunicação textual. Os usuários veem a mesma imagem porque toda área de visão é sincronizada, no entanto, a sincronização é feita por propagação de ação e eventos ao invés de *streaming* de vídeo, o que diminuiu o uso de banda e atrasos.

Figura 15 – Captura de tela do módulo de visualização no instante em que carrega as imagens de alta qualidade. No canto superior esquerdo está o mini-mapa com a visualização da imagem inteira, a área destacada é a região da imagem que está sendo visualizada no momento.



5.2.1 Sincronização de Ações

A técnica AJAX é utilizado para estabelecer comunicação com o servidor *web* responsável pela sincronizações de ações e troca de mensagens no bate papo. A princípio, a comunicação via AJAX segue o modelo tradicional de requisição-resposta, o cliente faz uma requisição ao servidor, e é dada uma resposta. Não é possível o servidor tomar a iniciativa de comunicação com o cliente. Este modelo não serve para a aplicação pois a qualquer momento o usuário pode receber uma mensagem do bate papo ou uma sincronização do estado da aplicação.

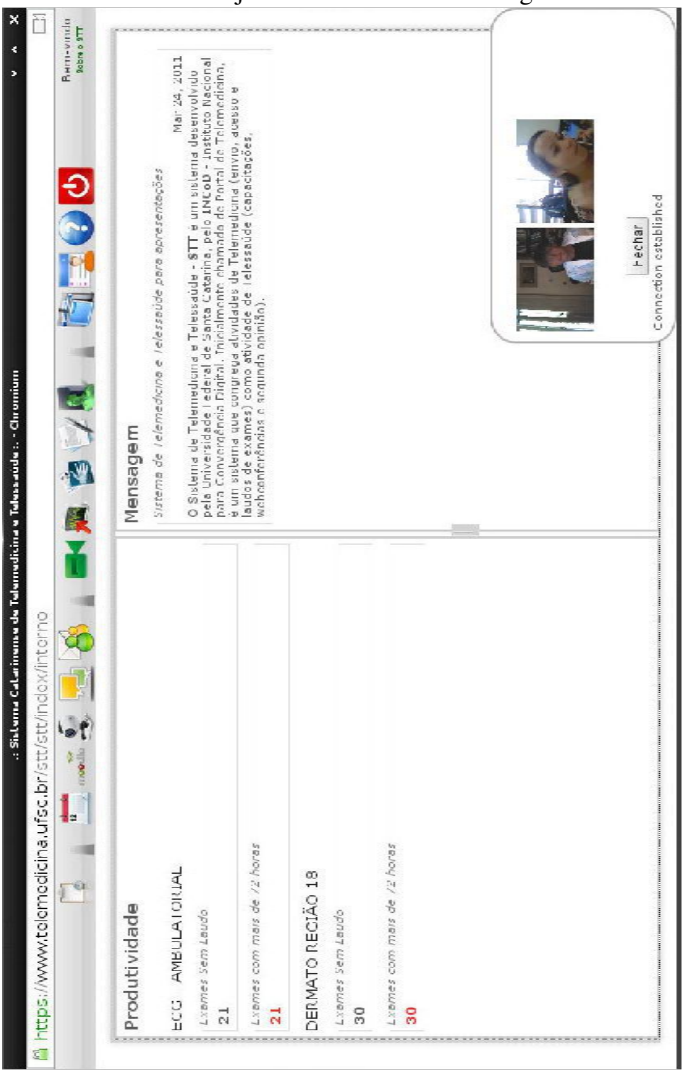
A primeira solução para este problema é a utilização de *polling,* o cliente fica o tempo inteiro fazendo requisições para o servidor. O servidor dá uma resposta nula caso não tenha nenhuma mensagem nova para o cliente e quando há alguma mensagem a espera, o servidor a retorna. Desta forma, o servidor poderá enviar mensagens para o cliente a qualquer momento sem ferir o princípio de requisição-resposta. A grande desvantagem é a grande quantidade de troca de mensagens nulas entre o cliente o servidor. Uma otimização que pode ser feita ao *polling* é ao invés do servidor mandar uma resposta nula para o cliente quando não tiver nada, o servidor manter a conexão aberta até a necessidade de uma resposta. Para o navegador, o servidor apenas está demorando para responder, mas na prática, o cliente está com um canal aberto esperando uma mensagem do servidor.

A biblioteca *Ajax Push Engine* (APE, 2014) foi usada para abstrair a comunicação bidirecional entre o cliente e o servidor.

5.2.2 Vídeo Chamadas

Também há suporte para vídeo chamadas em tempo real, aumentando o nível de comunicação e melhorando a colaboratividade em comparação a bate papo tradicional baseado em texto. É utilizado HTML5 para ter acesso a *webcam* e vídeo de um usuário, bem como para criar a reprodução do outro lado. A utilização de WebRTC (*Web Real-Time Communication*) faz com que isso seja possível sem a utilização de qualquer *plugin* auxiliar.

Figura 16 – Captura de tela da página inicial do Sistema Catarinense de Telessaúde com mini-janela de vídeo chamada ligada



Quando estabelecendo uma vídeo chamada, os clientes trocam informações sobre a conexão através do servidor de sinalização, no nosso caso, o próprio servidor *web*. Após esta troca, os clientes formar uma conexão direta de forma a garantir performance máximo e não sobrecarregar o servidor. Os clientes mantém também uma comunicação ativa com o servidor de sinalização para que seja possível monitor o estado da conexão de ambas as partes o tempo todo. Durante a comunicação, pode haver instabilidades na conexão entre os usuários. No entanto, a conexão com o servidor de sinalização utiliza AJAX e é confiável, isso significa que se a conexão de monitoramento for interrompida, houve algum problema real na comunicação, como por exemplo perda de acesso a Internet.

Quando a aplicação detecta uma perda de conexão, o processo é reiniciado e os clientes ficam requisitando conexão até que ela seja restabelecida. Portanto, o processo fica transparente para o usuário, quando o fluxo de vídeo e áudio são interrompidos, basta esperar que o *software* restabelecerá a conexão automaticamente. Como a quantidade de informação trocada na monitoração é de apenas alguns bytes por segundo, este processo não cria interferências na vídeo chamada.

6 VALIDAÇÃO

6.1 DESEMPENHO DO VISUALIZADOR DE IMAGENS GRANDES

A validação do módulo de visualização de imagens grandes foi feita através da análise do espaço necessário para armazenar a imagem no lado servidor, a quantidade de dados que é transferida do servidor para o cliente e o tempo de resposta após cada ação de navegação feita pelo usuário. Sendo que o tempo de resposta é o aspecto mais importante pois está relacionado com a usabilidade do sistema. Foram gravadas as movimentações feitas pelo usuário em quatro casos de navegação com em média 1400 amostras de região visível em cada caso. As Figuras 18 e 19 contém as imagens utilizadas nos casos de teste.

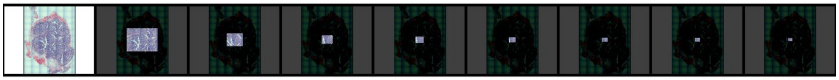
A imagem de lâmina histopatológica ocupava em seu formato original 130MB, na versão em formato piramidal a soma dos *tiles* ocupa 165MB, um aumento de 19% no espaço ocupado. Já a imagem formado por fotografias da porção visível da Lua ocupava originalmente 420MB, e precisa de 49% a mais de espaço em sua versão piramidal, 625MB. Os casos de testes estão demonstrados na figura 17.

O caso de teste (a) tem o objetivo de analisar o comportamento do sistema no momento em que o usuário está aumentando o *zoom* para focar um ponto específico. No caso de teste (d) a imagem é navegada integralmente em *zoom* máximo, ou seja no final da navegação a imagem inteira foi baixada. Já os casos (b) e (c) demonstram casos típicos da ferramenta, em que o usuário navega em *zoom* baixo pela imagem inteira e amplia para inspecionar pontos característicos.

Três dados foram extraídos em função do tempo para cada caso de teste. Primeiro, os gráficos da Figura 20 demonstram a quantidade de banda necessária para baixar os *tiles* relacionado à região visível em cada instante desconsiderando todo *tile* que já foi baixado antes, além de o tempo necessário para *download* considerando velocidade de conexão de 128KB/s. Estão separados os *tiles* relacionados a qualidade rápida, que são os *tiles* do nível anterior com o objetivo de fornecer

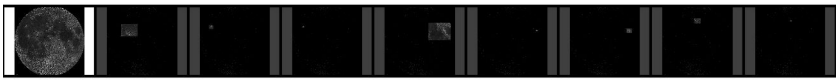
Figura 17 – Trechos da navegação percorrida em cada caso de teste do visualizador de imagens grandes, a região iluminada no minimapa indica a porção visível da imagem a cada instante

(a) Nenhuma navegação lateral, apenas *zoom* sobre o cento da imagem até atingir 100%



(b) Navegação lateral na metade do *zoom* e foco de 100% em alguns pontos característicos da imagem

(c) Mesmo comportamento de inspeção de *zoom* máximo em apenas alguns pontos utilizando a imagem maior



(d) Navegação em *zoom* máximo por toda a imagem, nenhum ponto é visualizado com menos de 100% de *zoom* além da posição inicial

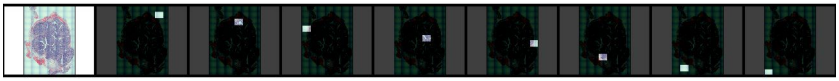


Figura 18 – Lâmina histopatológica digitalizada em microscópio óptico

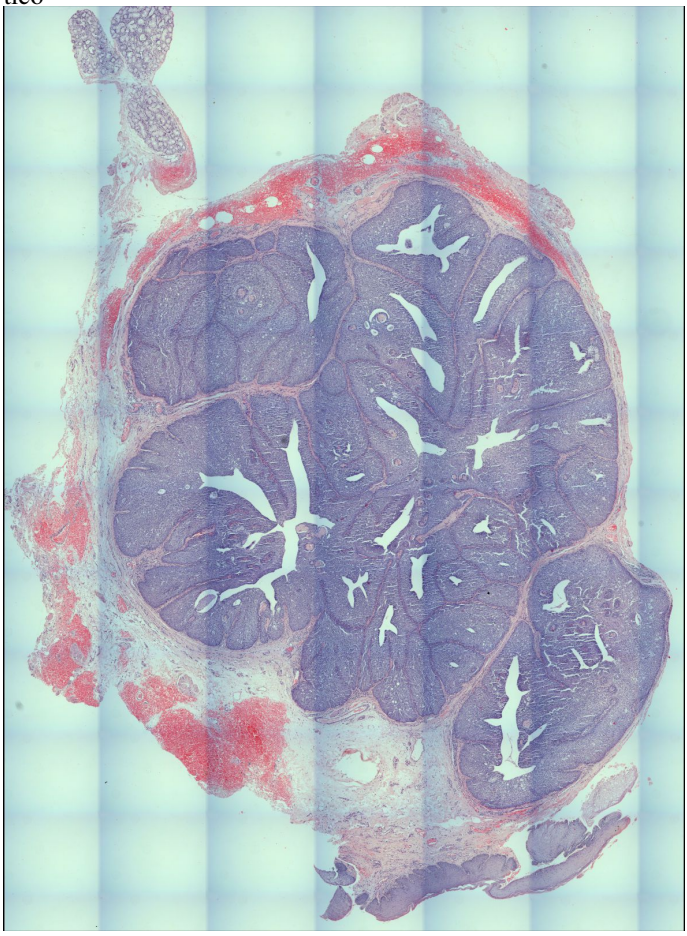
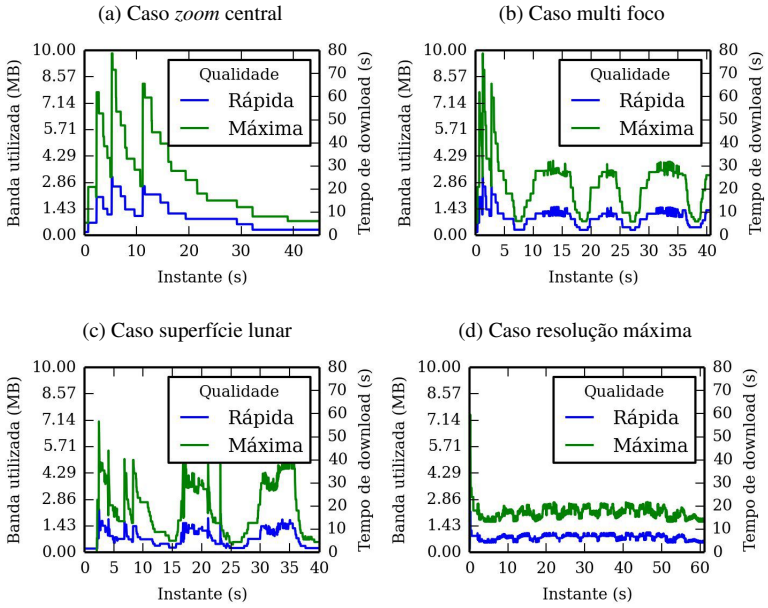


Figura 19 – Reconstrução em mosaico de várias fotografias do lado visível da Lua

Fonte: Lunar Reconnaissance Orbiter Camera (LROC) (2014)

Figura 20 – Quantidade de banda necessária em cada instante individualmente

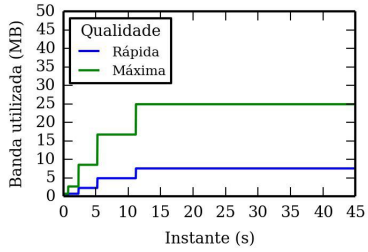


um resultado visual agradável o mais rápido possível, dos *tiles* de qualidade máxima, que tem o tamanho adequado para o tamanho de *zoom* selecionado. Na Figura 21 é demonstrado quantos MB foram necessários até então em cada instante da navegação, também separados em qualidade rápida e qualidade máxima. Finalmente, a quantidade total de *pixels* visíveis e quantos *pixels* foram de fato baixados (devido ao *tiling*) é demonstrado na Figura 22.

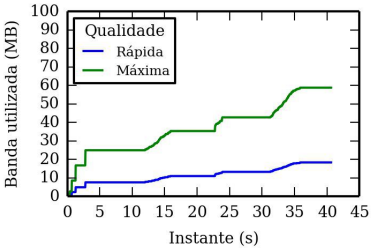
Através dos dados extraídos da Figura 20 conclui-se que em poucos instantes observados individualmente é necessário esperar mais do que 10 segundos para visualizar a imagem com qualidade razoável. Como a movimentação do usuário é contínua e suave, isto é, a cada instante apenas alguns novos *tiles* são requisitados ao servidor, a navegação demanda pouco tempo de espera. A quantidade de banda total utilizada na Figura 21 demonstra que em uma navegação com-

Figura 21 – Quantidade de banda total utilizada durante a navegação

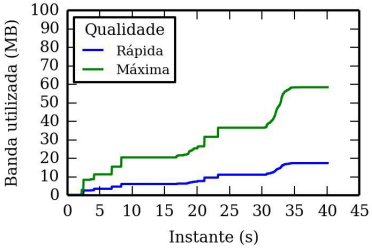
(a) Caso zoom central



(b) Caso multi foco



(c) Caso superfície lunar



(d) Caso resolução máxima

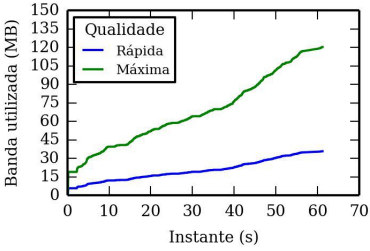
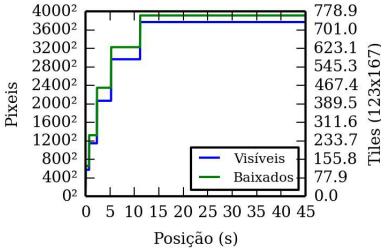
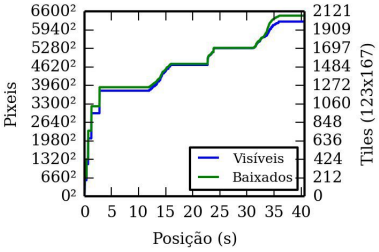


Figura 22 – Quantidade de *pixels* visualizados e de *pixels* baixados

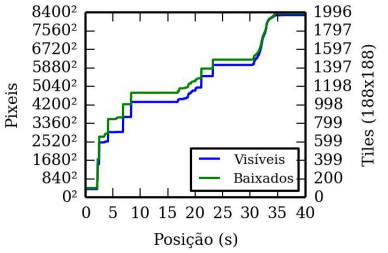
(a) Caso *zoom* central



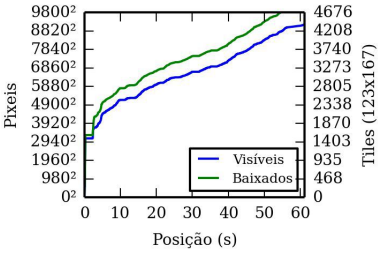
(b) Caso multi foco



(c) Caso superfície lunar



(d) Caso resolução máxima



pleta são baixados menos MBs do que o tamanho da imagem inteira. Finalmente, percebe-se que a abordagem baseada em *tiles* é eficiente na relação de quantos *pixels* foram baixados e quantos são realmente necessários para a região de visão atual, demonstrado na Figura 22.

6.2 USABILIDADE DE USUÁRIOS

Para validar o impacto da ferramenta sob a prática do trabalho diagnóstico de especialista orais, a experiência do usuário foi avaliada para como forma de medir as ferramentas propostas.

Nós decidimos usar a Escala de Usabilidade de Sistema (SUS) (BROOKE, 1996) como estratégia de validação por ser amplamente utilizada como um instrumento que captura as impressões particulares que os indivíduos tiveram, baseado em suas experimentações com o *software*, mas nos dá uma pontuação numérica então comparações com o estado da arte é possível. A SUS é composta por um formulário simples com dez possíveis afirmações em escala likert relacionadas a visão global da avaliação de usabilidade.

1. “Eu acho que gostaria de usar esse sistema frequentemente.”
2. “Eu achei o sistema desnecessariamente complexo.”
3. “Eu achei o sistema fácil de usar.”
4. “Eu acho que eu precisaria de ajuda de um técnico para ser capaz de usar o sistema.”
5. “Eu achei que as funções do sistema estavam bem integradas.”
6. “Eu achei que havia muitas inconsistências no sistema.”
7. “Eu acho que a maioria das pessoas aprenderia a usar esse sistema rapidamente.”
8. “Eu achei o sistema muito incômodo de usar.”
9. “Eu me senti confiante usando o sistema.”
10. “Eu precisei aprender muitas coisas antes de conseguir usar o sistema.”

Para cada item, há cinco opções de respostas que variam de “discordo totalmente” (posição 1) e “concordo totalmente” (posição 5). O formulário é aplicado depois que o indivíduo teve a oportunidade de manusear a aplicação, mas antes de mais discussões sobre ela.

A pontuação final é calculada dando um valor de escala para cada item. Itens pares contém aspectos positivos, então o valor é a posição da resposta menos 1, e itens impares contém aspectos positivos, então o valor é 5 menos a posição da resposta. Isso normaliza as respostas para um intervalo de 0 a 4, com 4 sendo o melhor valor. Os valores são somados e multiplicados por 2,5, que dá uma pontuação variando de 0 a 100. Adicionalmente, a moda de cada questão pode ser analisada para avaliar o impacto de cada aspecto.

Os indivíduos da validação foram divididos em dois grupos. O primeiro sendo composto por onze pesquisadores que estavam direta ou indiretamente envolvidos no projeto, distribuídos em cinco especialidades de diagnóstico/tratamento oral, o segundo grupo envolve nove usuários da rede de Telemedicina, mas não estavam envolvidos na pesquisa. A Figura 23a mostra o resultado para o grupo 1, e a Figura 23b mostra o resultado para o grupo 2.

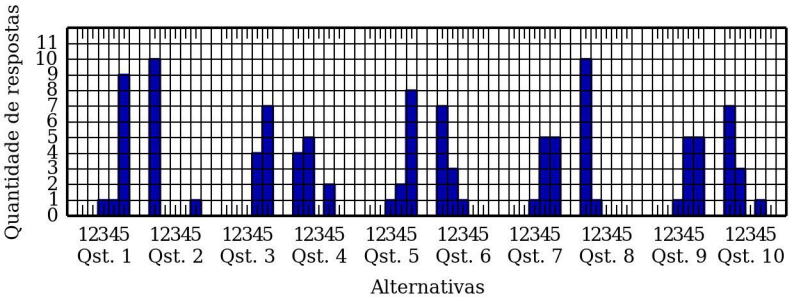
A pontuação media para o grupo 1 é 88,2, a pontuação média para o grupo 2 é 82,5 e a média geral é 85,63. Como o grupo 1 teve algum contato com a aplicação antes do teste de usabilidade, acredita-se que essa experiência extra afetou suas performances no teste, que afeta a pontuação individual.

Apesar do SUS não ser uma escala absoluta, sabe-se que pontuações por volta dos 70s indicam um valor mínimo aceitável, enquanto que resultados melhores tendem a variar entre 70 e 80, e resultadores superiores passam da marca do 90 (BANGOR; KORTUM; MILLER, 2008). Bangor, Kortum e Miller (2009) até definiram uma tradução de valores numéricos para adjetivos como mostra a Tabela 6.

Os aspectos 4, o usuário considera que precisaria de ajuda de um técnico, e 10, o usuário considera que precisou aprender muita coisa antes de usar o sistema, tiveram pontuação levemente a baixo da média. Esses aspectos estão relacionados com a curva de aprendizado para utilização de uma ferramenta, isto significa que a ferramenta tem uma usabilidade boa mas que é necessário algum treinamento prévio. Para melhorar essa deficiência, é necessário investigar quais aspectos da interface não são intuitivos para o usuário. Por exemplo, o ícone

Figura 23 – Frequência de cada resposta (1 a 5) para cada uma das 10 questões do *SUS*

(a) Grupo 1



(b) Grupo 2

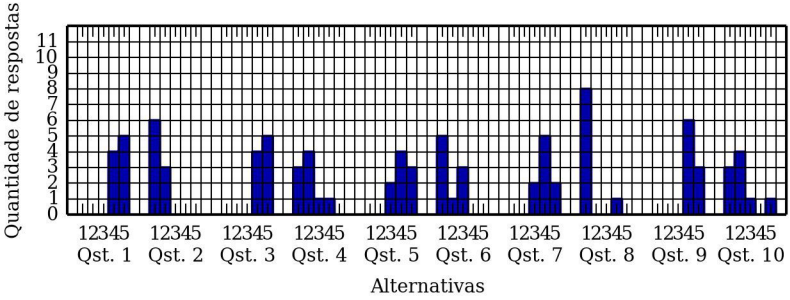


Tabela 6 – Pontuação *SUS* como Adjetivos

Pontos	Desvio Padrão	Adjetivo
90.9	13.4	"melhor imaginável"
85.5	10.4	"excelente"
71.4	11.6	“bom”
50.9	13.8	“OK”
35.7	12.6	“pobre”
20.3	11.3	“terrível”
12.5	13.1	“pior imaginável”

em um botão pode ser óbvio ao ponto de que o usuário saiba qual papel desempenha o botão sem nenhuma informação adicional. É o que acontece com o botão de "Salvar documentos", ele é o mesmo na maioria dos *softwares*, o que faz com que as pessoas identifiquem-no rapidamente.

No entanto, os resultados gerais mostram que a plataforma satisfaz a expectativa de nosso público alvo. A moda das resposta é o valor 4 ("concordo totalmente" para questões positivas e "discordo totalmente" para questões negativas). Mesmo as questões que possuem valores abaixo da média possuem pontuação que fica entre "bom" e "excelente".

7 CONCLUSÃO E TRABALHOS FUTUROS

Foi apresentada os componentes que formam um sistema de comunicação multimídia e como que eles são montados para criar uma ferramenta completamente baseada em *web* integrando as funcionalidades de visualização de imagens médicas com a comunicação na plataforma colaborativa. Graças a abordagem *web*, a ferramenta é acessível de qualquer dispositivo que tenha um navegador *web* moderno instalado com conexão a Internet, atingindo o objetivo de garantia de acesso. As técnicas empregadas ajudam a manter os requisitos de sistema e de rede baixos. A ausência de qualquer tipo de *plugin* auxiliar melhora mais ainda o aspecto da portabilidade.

A validação de usabilidade demonstra que a aplicação *web-based* adapta-se as necessidades de experiência de usuário do público alvo. A pontuação geral do SUS indica que há correspondência de qualidade entre a solução proposta e uma solução tradicional baseada em *desktop*.

O servidor de sinalização (*signaling server*) toma um papel menos passivo do que acontece tradicionalmente, ele monitora as conexões dos usuários para mitigar qualquer instabilidade da conexão P2P por parte do WebRTC e da camada de rede subjacente. A medida que o protocolo WebRTC e suas implementações evoluem, será possível adaptar mais ainda e evoluir o processo de monitoramento com as funcionalidades novas.

A conexão direta entre os usuário deve ser utilizada na vídeo chamada para mantê-la viável do ponto de vista de rede. Por outro lado, o processo de colaboração a nível de ferramentas da plataforma não utiliza P2P pois é composto de transferência de dados relativamente menor. Como trabalho futuro, o AJAX poderia ser abandonado em favor de *WebSockets* ou WebRTC, que se adaptam melhor para troca constante de mensagens. Cada um tem vantagens e desvantagens, portanto experimentações se mostram necessárias para decidir qual abordagem é mais viável para o contexto de plataforma colaborativa.

A ferramenta de colaboração por vídeo ainda tem a limitação de não suportar de forma eficiente conferência com mais de duas pes-

soas, o usuário precisa estabelecer uma conexão para cada participante novo. Nesse ponto há grande potencial para melhorias, principalmente a adição de suporte aos protocolos do WebRTC no lado do servidor. Quando o servidor consegue se comunicar com os usuários através de WebRTC, além do conceito de sala de reunião virtual, seria possível fornecer uma série de recursos secundário, como por exemplo a possibilidade de realizar a gravação e arquivação de de vídeo chamadas para serem assistidas posteriormente.

A portabilidade da aplicação é um dos objetivos mais importantes deste trabalho que foi alcançado através do uso da *web* como base. Porém, nenhum experimento foi feito com usuários de *smartphones* e *tablets*, a interface com usuário é baseada em gestos de mouse (principalmente arrastar), o que dificulta muito a usabilidade em dispositivos de *touchscreen.* O próximo passo para garantir portabilidade da aplicação é fazer versões especializadas para esse tipo de dispositivos e repetir a avaliação de usabilidade.

O trabalho proposto produziu o seguinte artigo, onde foi avaliado e aceito para a publicação:

André Puel, Aldo von Wangenheim, Maria Inês Meurer, and Douglas D. J. de Macedo. Bucomax: Collaborative multimedia platform for real time manipulation and visualization of bucomaxillofacial diagnostic images. In *Proceedings of the 2014 IEEE 27th International Symposium on Computer-Based Medical Systems*, CBMS '14, pages 392–395, Washington, DC, USA, 2014. IEEE Computer Society.

REFERÊNCIAS

ABRARDO, A.; CASINI, A. L. Embedded java in a web-based teleradiology system. **Internet Computing, IEEE**, v. 2, n. 3, p. 60–68, May 1998. ISSN 1089-7801.

AHMED, N.; NATARAJAN, T.; RAO, K. R. Discrete cosine transform. **Computers, IEEE Transactions on**, IEEE, v. 100, n. 1, p. 90–93, 1974.

APE, A. P. E. **Online at http://www.ape-project.org/**. [S.l.]: June, 2014.

AUDET, F.; JENNINGS, C. **RFC 4787-Network Address Translatoin NAT Behavioral Requirements for Unicast UDP**. 2007.

BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. **Journal of usability studies**, University of applied science of Häme, v. 4, n. 3, p. 114–123, 2009.

BANGOR, A.; KORTUM, P. T.; MILLER, J. T. An empirical evaluation of the system usability scale. **International Journal of Human-Computer Interaction**, v. 24, n. 6, p. 574–594, 2008.

BANKOSKI, J. et al. **VP8 Data Format and Decoding Guide**. 2011.

BAUGHER, M. et al. The secure real-time transport protocol (srtp). **Internet Engineering Task Force**, 2004.

BRADEN, R. **Requirements for Internet Hosts–Application and Support**. 1989.

BRADEN, R. et al. **Requirements for Internet Hosts—Communication Layers**. 1989.

BROOKE, J. Sus-a quick and dirty usability scale. **Usability evaluation in industry**, London: Taylor & Francis, v. 189, p. 194, 1996.

BURNETT, D. et al. **WebRTC 1.0: Real-time Communication Between Browsers**. [S.l.], set. 2013. Http://www.w3.org/TR/2013/WD-webrtc-20130910/.

CALIFORNIA, R. of the University of. **"The BSD 2-Clause License"**. 1998. Http://opensource.org/licenses/BSD-2-Clause. Acessado: 06/03/2014.

CHAN, Y. L.; LIM, Y. S.; FENG, D. D. Web-based multimedia collaboration system for medical images analysis and diagnosis. In: **Selected Papers from the 2002 Pan-Sydney Workshop on Visualisation - Volume 22**. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2002. (VIP '02), p. 13–15. ISBN 1-920682-01-5.

CONSTANTINESCU, L.; KIM, J.; FENG, D. Sparkmed: A framework for dynamic integration of multimedia medical data into distributed m-health systems. **Information Technology in Biomedicine, IEEE Transactions on**, v. 16, n. 1, p. 40–52, Jan 2012. ISSN 1089-7771.

DIERKS, T.; RESCORLA, E. **The Transport Layer Security (TLS) protocol version 1.2**. [S.l.], 2008.

EGEVANG, K.; FRANCIS, P. Rfc1631: The ip network address translator. **Internet Engineering Task Force**, 1994.

FAIRCHILD, M. D. **Color appearance models**. [S.l.]: John Wiley & Sons, 2013.

FETTE, I.; MELNIKOV, A. Rfc 6455: The websocket protocol. **Internet Engineering Task Force (IETF), December**, 2011.

HAMZA-LUP, F. G.; DAVIS, L.; ZEIDAN, O. A. Web-based 3d planning tool for radiation therapy treatment. In: **Proceedings of the Eleventh International Conference on 3D Web Technology**. New York, NY, USA: ACM, 2006. (Web3D '06), p. 159–162. ISBN 1-59593-336-0.

HICKSON, I. **The WebSocket API**. [S.l.], set. 2012. Http://www.w3.org/TR/2012/CR-websockets-20120920/.

HSIAO, C.-H. et al. Use of a rich internet application solution to present medical images. **Journal of Digital Imaging**, Springer-Verlag, v. 24, n. 6, p. 967–978, 2011. ISSN 0897-1889.

HUFFMAN, D. A. A method for the construction of minimum redundancy codes. **proc. IRE**, v. 40, n. 9, p. 1098–1101, 1952.

HUSTON, G. Anatomy: A look inside network address translators. **The Internet Protocol Journal**, v. 7, n. 3, p. 2–32, 2004.

IEC, I. Information technology-digital compression and coding of continuous-tone still images: Requirements and guidelines. **Standard, ISO IEC**, p. 10918–1, 1994.

ISERHARDT-BAUER, S. et al. Case study: Medical web service for the automatic 3d documentation for neuroradiological diagnosis. In: **Proceedings of the Conference on Visualization '01**. Washington, DC, USA: IEEE Computer Society, 2001. (VIS '01), p. 425–428. ISBN 0-7803-7200-X.

ISO/IEC. 200 (1994)| iso/iec 7498-1. **Information technology–Open Systems Interconnection–Basic Reference Model: The basic model**, 1994.

ITU, I. T. U. H. 262| iso/iec 13818-2. **Information technology—Generic coding of moving pictures and associated audio information—Video**, 2000.

ITU, I. T. U. Recommendation itu-r bt.709-5 – parameter values for the hdtv standards for production and international programme exchange. **Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange, now ITU-R BT**, v. 709-5, 2002.

JENNINGS, C.; MAHY, R.; AUDET, F. **RFC5626: Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)**. 2009.

JVT, J. V. T. recommendation and final draft international standard of joint video specification (itu-t rec. h. 264| iso/iec 14496-10 avc). **Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVTG050**, v. 33, 2003.

KASPAR, M.; PARSAD, N. M.; SILVERSTEIN, J. C. Cowebviz: Interactive collaborative sharing of 3d stereoscopic visualization among browsers with no added software. In: **Proceedings of the 1st ACM International Health Informatics Symposium**. New York, NY, USA: ACM, 2010. (IHI '10), p. 809–816. ISBN 978-1-4503-0030-8.

KIM, J.; FENG, D. D.; CAI, T. W. A web based medical image data processing and management system. In: **Selected Papers from the Pan-Sydney Workshop on Visualisation - Volume 2**. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2001. (VIP '00), p. 89–91. ISBN 0-909-92580-1.

KIM, J. et al. Integrated multimedia medical data agent in e-health. In: **Proceedings of the Pan-Sydney Area Workshop on Visual Information Processing - Volume 11**. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2001. (VIP '01), p. 11–15. ISBN 0-909-92589-5.

KITCHENHAM, B. A.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. 2007.

LIM, Y. S.; FENG, D. D.; CAI, T. W. A web-based collaborative system for medical image analysis and diagnosis. In: **Selected Papers from the Pan-Sydney Workshop on Visualisation - Volume 2**. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2001. (VIP '00), p. 93–95. ISBN 0-909-92580-1.

MAHMOUDI, S. E. et al. Web-based interactive 2d/3d medical image processing and visualization software. **Computer Methods and Programs in Biomedicine**, v. 98, n. 2, p. 172 – 182, 2010. ISSN 0169-2607.

MATA, C. et al. Mammoapplet: An interactive java applet tool for manual annotation in medical imaging. In: **Bioinformatics Bioengineering (BIBE), 2012 IEEE 12th International Conference on**. [S.l.: s.n.], 2012. p. 34–39.

MATSOPOULOS, G. K. et al. Mitis: a www-based medical system for managing and processing gynecological–obstetrical–radiological

data. **Computer Methods and Programs in Biomedicine**, v. 76, n. 1, p. 53 – 71, 2004. ISSN 0169-2607.

MATTHEWS, P.; ROSENBURG, J.; MAHY, R. **The internet engineering task force RFC5766: TURN-traversal using relays around NAT**. 2010.

MOZILLA. **"Firefox Notes - v22"**. 2013. Https://www.mozilla.org/en-US/firefox/22.0/releasenotes/. Accessado: 06/03/2014.

OPERA. **"Opera Desktop 18 released"**. 2013. My.opera.com/ODIN/blog/opera-desktop-18-released. Acessado: 05/03/2014.

POSTEL, J. User datagram protocol. **Internet Engineering Task Force**, 1980.

POSTEL, J. Transmission control protocol. **Internet Engineering Task Force**, 1981.

POYNTON, C. Gamma faq–frequently asked questions about gamma. v. 25, 2002.

POYNTON, C. Color faq–frequently asked questions about color. 2006.

POYNTON, C. **Digital video and HD: Algorithms and Interfaces**. [S.l.]: Elsevier, 2012.

REKHTER, Y. et al. Rfc 1918: Address allocation for private internets. 1996. **Internet Engineering Task Force**, p. 8, 1996.

ROSENBERG, J. **RFC5245: a protocol for network address translator (NAT) traversal for ofer/answer protocols**. 2010.

ROSENBERG, J. et al. Rfc 5389: Session traversal utilities for nat (stun). **Internet Engineering Task Force**, 2008.

SALOMON, D. **Data compression: the complete reference**. [S.l.]: Springer, 2004.

SáNCHEZ, C.; TRIANA, E.; ROMERO, E. A flexible web oriented telehealth platform using a rim-hl7 based model. In: **Proceedings of the 2008 Euro American Conference on Telematics and Information Systems**. New York, NY, USA: ACM, 2008. (EATIS '08), p. 7:1–7:8. ISBN 978-1-59593-988-3.

SCHULZRINNE, H. et al. Rtp: A transport protocol for real-time applications. **Internet Engineering Task Force**, 2003.

SHEN, H. et al. Managing and collaboratively processing medical image via the web. In: BAO, Z. et al. (Ed.). **Web-Age Information Management**. [S.l.]: Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science, v. 7419). p. 252–263. ISBN 978-3-642-33049-0.

SHEN, H. et al. Miaps: A web-based system for remotely accessing and presenting medical images. **Computer Methods and Programs in Biomedicine**, v. 113, n. 1, p. 266 – 283, 2014. ISSN 0169-2607.

SMITH, T.; GUILD, J. The c.i.e. colorimetric standards and their use. **Transactions of the Optical Society**, v. 33, n. 3, p. 73, 1931.

STEPHAN, E. Ip performance metrics (ippm) metrics registry. **Internet Engineering Task Force**, 2005.

SUNG, M. Y. et al. Comed: a real-time collaborative medicine system. **International Journal of Medical Informatics**, v. 57, n. 2–3, p. 117 – 126, 2000. ISSN 1386-5056.

WANG, F.; RABSCH, C.; LIU, P. Native web browser enabled svg-based collaborative multimedia annotation for medical images. In: **Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on**. [S.l.: s.n.], 2008. p. 1219–1228.

WEBRTC. 2013. Http://www.webrtc.org/. Accessado: 05/03/2014.

ZHANG, J. et al. Web-based electronic patient records for collaborative medical applications. **Computerized Medical Imaging and Graphics**, v. 29, n. 2–3, p. 115 – 124, 2005. ISSN 0895-6111. Imaging Informatics.

APÊNDICE A – Figuras Utilizadas nos Experimentos

Figura 24 – Arara-canindé

As Figuras 24 e 25 são as fotos utilizadas nas demonstração com processamento de imagens. Fotos reais foram utilizadas nas demonstrações pois a suavidade das cores (pontos próximos tendem a ter valor de cor semelhantes) desempenha um papel importante nos algoritmos de compactação de imagens.

Figura 25 – Luminância de Arara-canindé

APÊNDICE B – Exemplo de Codificação de Huffman

O algoritmo de Huffman utiliza uma tabela de frequência para construir uma codificação de tamanho variado onde símbolos mais frequentes ocupam menos bits para ser representados. Esta etapa pode ser vista como a construção de uma árvore binária onde dois símbolos de baixa frequência se juntam num nodo pai de frequência maior. Na árvore resultante, o caminho até um símbolo, indica sua codificação, para cada nodo há dois caminhos, para baixo (*bit* 0) ou para cima (*bit* 1).

Para exemplificar o processo, foi utilizado o próprio texto desta dissertação para calcular a tabela de frequência, Tabela 7. Com a tabela, foi construído a codificação, a árvore representante desta codificação pode ser vista na Figura 26.

O arquivo de texto original, que usa 8 bits para representar cada símbolo, ocupa 927K. O arquivo codificado em Huffman (incluindo a tabela de símbolos) ocupa 526K. O tamanho foi reduzido quase pela metade.

Figura 26 – Exemplo de Árvore de Codificação de Huffman

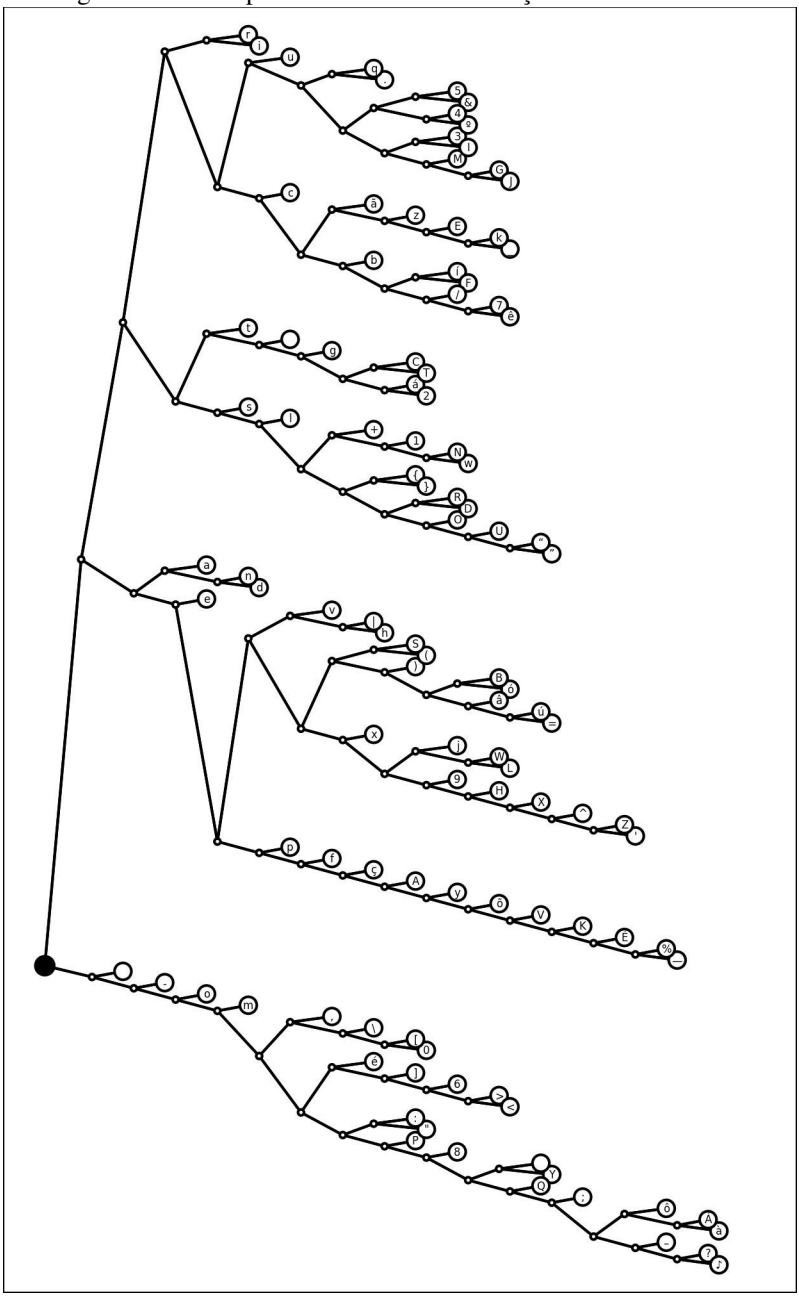


Tabela 7 – Frequência de cada símbolo no texto

Sím.	Freq.	Sím.	Freq.	Sím.	Freq.	Sím.	Freq.	Sím.	Freq.
espaço	29934	ã	824	0	264	y	159	“	46
-	17154	.	754	[	262	j	141	V	39
e	9902	q	750	]	261	9	139	=	37
a	9623	+	703	P	253	6	131	ú	35
o	8371	ç	635	"	243	8	126	X	32
i	5986	h	600	:	235	ê	116	Y	32
r	5710	|	599	F	231	7	107	tabulação	30
s	5645	x	539	í	231	_	105	Q	28
t	5003	\	529	/	217	k	101	K	21
d	4542	é	499	E	215	J	100	^	15
n	4506	z	436	I	206	G	93	;	15
m	3595	}	353	3	196	U	90	É	9
c	3360	{	353	M	194	õ	89	’	9
u	3152	1	350	º	193	ó	78	Z	9
l	2668	2	344	4	192	B	77	—	5
nova linha	2548	á	338	&	192	â	72	%	5
p	2446	T	338	5	185	L	70	ô	4
g	1301	C	335	O	184	W	69	–	3
f	1237	A	331	D	180	H	67	à	2
v	1221	)	284	R	174	<	64	Á	2
,	1044	(	284	w	173	>	64	?	2
b	912	S	282	N	171	”	46	♪	1

APÊNDICE C – Transformada Discreta de Cosseno

A Equação (C.2) demonstra a transformação de um vetor X com M elementos no vetor de coeficientes G_x com a mesma quantidade de elementos. A transformação de G_X de volta para X é demonstrado na Equação (C.3) (AHMED; NATARAJAN; RAO, 1974).

$$
\begin{aligned}
C(k) &= \begin{cases} \frac{\sqrt{2}}{M}; & k = 0 \\ \frac{2}{M}; & k \neq 0 \end{cases} \\
D(k) &= \begin{cases} \frac{1}{\sqrt{2}}; & k = 0 \\ 1; & k \neq 0 \end{cases} \qquad \text{(C.1)} \\
G_x(k) &= C(k) \sum_{m=0}^{M-1} X_m \cos\left(\frac{(2m+1)k\pi}{2M}\right) \qquad \text{(C.2)} \\
X_m &= \sum_{k=0}^{M-1} D(k)\, G_x(k) \cos\left(\frac{(2m+1)k\pi}{2M}\right) \qquad \text{(C.3)}
\end{aligned}
$$

Para dados bidimensionais, o DCT é aplicado em cada uma das linhas, com a matriz resultante desta operação, o DCT é aplicado em cada uma das colunas. A Equação (C.4) demostra a primeira etapa do DCT bidimensional que é aplicado em cada linha, a Equação (C.5) demonstra a segunda etapa do DCT bidimensional que é aplicado em cada coluna, enquanto que a Equação (C.7) é a deduzida para representar o DCT bidimensional diretamente sobre a matriz original. O IDCT bidimensional é representado pela Equação (C.8), seguida pela Equação (C.9) e a Equação (C.11) representa o IDCT bidimensional sem passos intermediários.

$$
\begin{aligned}
C_M(k) &= \begin{cases} \frac{\sqrt{2}}{M}; & k = 0 \\ \frac{2}{M}; & k \neq 0 \end{cases} \\
C_N(k) &= \begin{cases} \frac{\sqrt{2}}{N}; & k = 0 \\ \frac{2}{N}; & k \neq 0 \end{cases} \\
D(k) &= \begin{cases} \frac{1}{\sqrt{2}}; & k = 0 \\ 1; & k \neq 0 \end{cases}
\end{aligned}
$$

$$H_x(u,v) = C_M(v)\sum_{j=0}^{M-1} X_{u,j}\cos\left(\frac{(2j+1)v\pi}{2M}\right) \quad \text{(C.4)}$$

$$G_x(u,v) = C_N(u)\sum_{i=0}^{N-1} H_x(i,v)\cos\left(\frac{(2i+1)u\pi}{2N}\right) \quad \text{(C.5)}$$

$$G_x(u,v) = C_N(u)\sum_{i=0}^{N-1}\left(C_M(v)\sum_{j=0}^{M-1} X_{i,j}\cos\left(\frac{(2j+1)v\pi}{2M}\right)\right)\cos\left(\frac{(2i+1)u\pi}{2N}\right) \quad \text{(C.6)}$$

$$G_x(u,v) = C_N(u)C_M(v)\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} X_{i,j}\cos\left(\frac{(2j+1)v\pi}{2M}\right)\cos\left(\frac{(2i+1)u\pi}{2N}\right) \quad \text{(C.7)}$$

$$H_x(i,j) = \sum_{u=0}^{N-1} D(u)G_x(u,j)\cos\left(\frac{(2i+1)u\pi}{2N}\right) \quad \text{(C.8)}$$

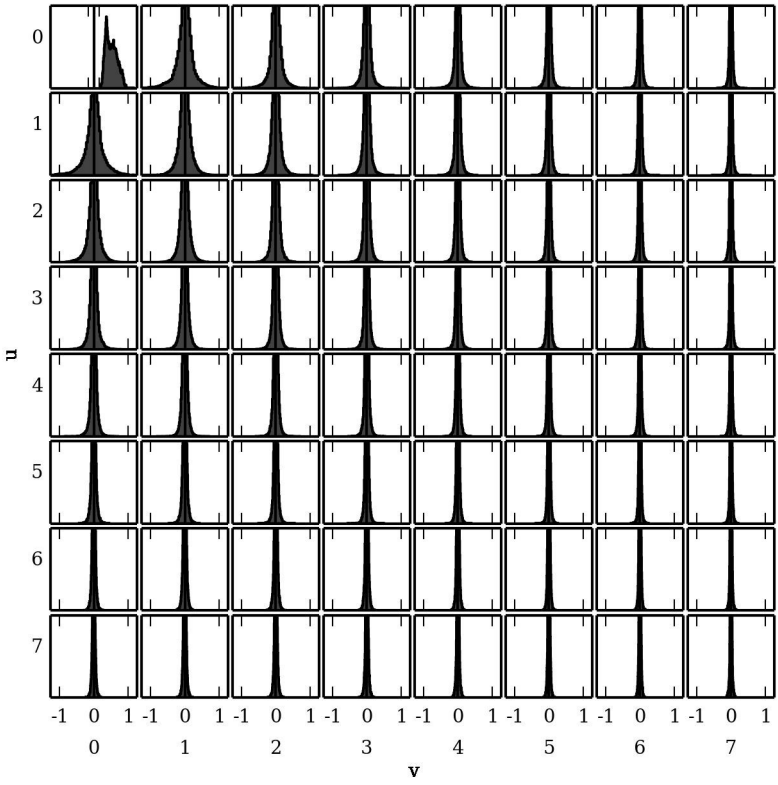
$$X_{i,j} = \sum_{v=0}^{M-1} D(v)H_x(i,v)\cos\left(\frac{(2j+1)v\pi}{2M}\right) \quad \text{(C.9)}$$

$$X_{i,j} = \sum_{v=0}^{M-1} D(v)\left(\sum_{u=0}^{N-1} D(u)G_x(u,v)\cos\left(\frac{(2i+1)u\pi}{2N}\right)\right)\cos\left(\frac{(2j+1)v\pi}{2M}\right) \quad \text{(C.10)}$$

$$X_{i,j} = \sum_{v=0}^{M-1}\sum_{u=0}^{N-1} D(v)D(u)G_x(u,v)\cos\left(\frac{(2i+1)u\pi}{2N}\right)\cos\left(\frac{(2j+1)v\pi}{2M}\right) \quad \text{(C.11)}$$

A Figura 25 foi utilizada para analisar a distribuição dos coeficientes do DCT. Aplicando a transformada nela é possível perceber que a medida que os coeficientes do DCT se afastam da origem – $G_x(0,0)$ – os valores tendem a se concentrar em torno do zero. A Figura 27 demonstra a frequência dos valores do DCT 8x8 por todas posições possíveis da imagem, o eixo X de cada quadro representa o valor do coeficiente, sendo que o centro do gráfico representa o valor 0. Quanto maior o valor do gráfico, maior a frequência de ocorrência do valor naquela posição. Cada quadro representa um elemento da matriz resultando do DCT $G_x(u,v)$. A medida que as coordenadas u e v aumentam, em direção do canto inferior direito, é mais frequente a ocorrência de valores próximos de zero. O primeiro elemento, que representa a frequência nula, é um caso excepcional que tende a ter valores muito maiores do que o restante dos coeficientes.

Figura 27 – Distribuição de frequência nos valores de cada posição (u, v) do DCT 8x8 aplicado em todas posições de uma foto, a primeira figura tem uma escala horizontal diferente das outros, como indicado pelas marcações verticais.



C.1 ANÁLISE DA EFICIÊNCIA DO DCT COMO CODIFICADOR COM PERDAS

Nesta seção, segue um conjunto de funções codificadas com *Discrete Cosine Transform*, quantificadas, e recuperadas. Algumas funções são codificadas com mais precisão, outras com menos. São utilizados exemplos que demonstram tanto casos bem comportados quanto casos em que a abordagem da quantificação não é satisfatória. Todas imagens são dividas em dois, a parte superior indica a função recodificada em preto, e a função exata em pontilhados, a parte inferior indica os coeficientes do DCT exato em linha pontilhada e os coeficientes quantizados em cor preta.

A função cosseno com frequência baixa, na Figura 28, representa exatamente o que o DCT consegue representar, basta um dos coeficientes para representar esta função, a reconstrução só causa erros notáveis se a quantização tiver passo de precisão muito grande.

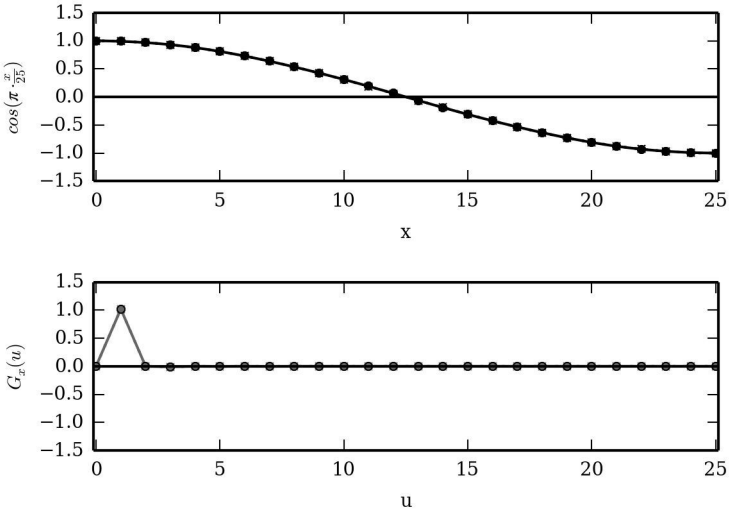
Como mencionado na Subseção 3.2.2, a matriz de quantização assume valores maiores para os coeficientes de alta frequência, portanto a função da Figura 29 não é satisfatoriamente representada pois a precisão da quantificação não é alta o suficiente.

A função seno, Figura 30, é um exemplo de uma função suave que não é facilmente representável com a DCT. Várias cossenos em frequências diferentes tem que ser somado para poder representar a curva do seno. O mesmo acontece para equações do tipo $\cos(x+c)$, onde há um deslocamento lateral na curva do cosseno.

Na Figura 31 está a função mais fácil de ser representada pelo DCT, a função constante. Uma função constante sempre tem todos coeficientes do DCT nulos exceto o primeiro. Na prática, isso significa que se houver uma região na imagem a ser compactada que tenha uma cor sólida, cada bloco vai ser representado utilizando apenas um valor. Equações de primeiro grau também são facilmente representadas pelo DCT, utilizando poucos coeficientes, como mostra a Figura 32.

Finalmente, na Figura 33, é representada uma função que tem oscilação e ao mesmo tempo mudança brusca de valor. Na função original, são assumidos apenas os valores 1 ou 2, essa forma discreta da função faz com que o DCT precise usar muitos coeficientes de alta frequência para conseguir representar o dado original.

Figura 28 – Função Cosseno codificada em DCT, quantizada e reconstruída



A transformada discreta do cosseno tende a usar coeficientes de baixa frequência para representar funções que tem mudanças suaves em relação ao domínio. Como fotografias tendem a ter correlação espacial, é possível utilizar DCT para compactar a informação através da representação de coeficientes de alta frequência com precisão baixa.

Figura 29 – Função Cosseno com Alta Frequência codificada em DCT, quantizada e reconstruída

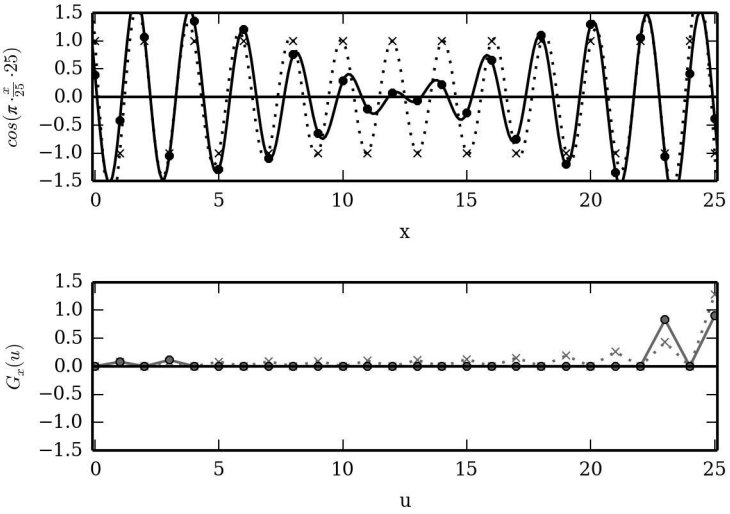


Figura 30 – Função seno codificada em DCT, quantizada e reconstruída

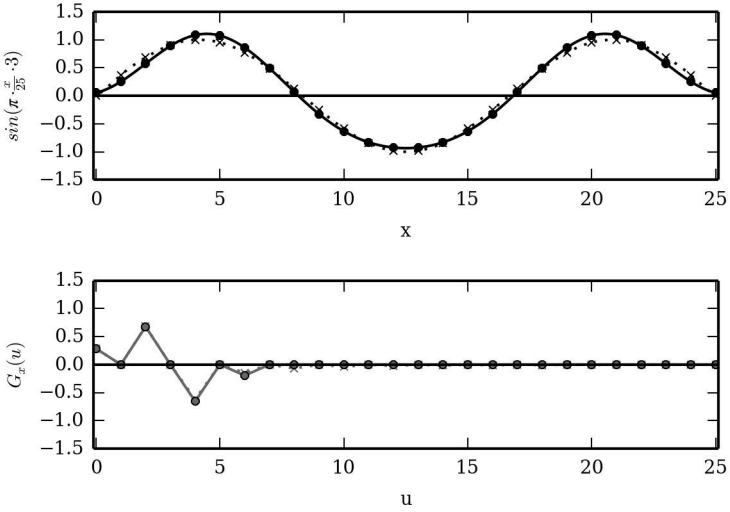


Figura 31 – Função constante codificada em DCT, quantizada e reconstruída

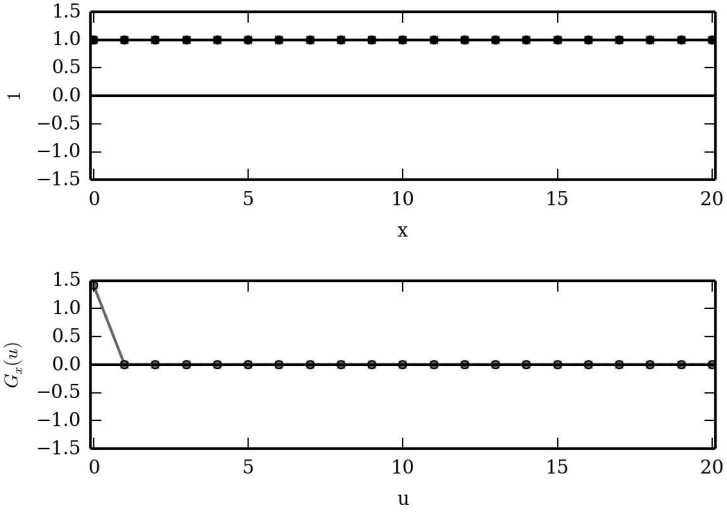


Figura 32 – Função linear codificada em DCT, quantizada e reconstruída

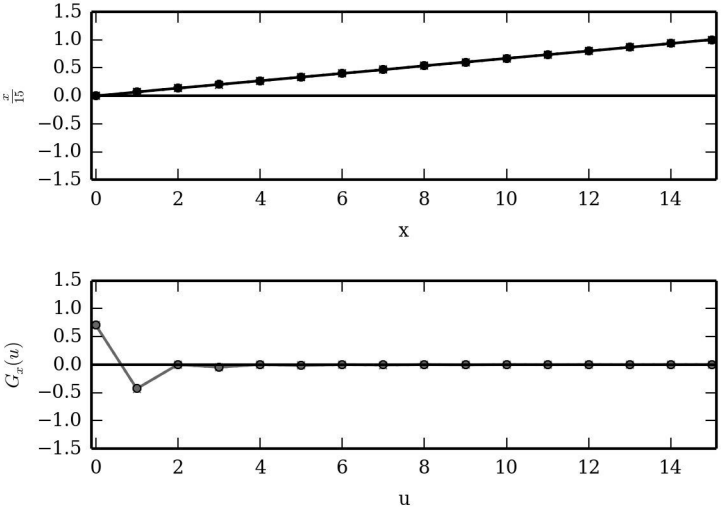


Figura 33 – Função cosseno com arredondamento para ter mudanças bruscas, codificada em DCT, quantizada e reconstruída