UNIVERSIDADE SALVADOR – UNIFACS
PROGRAMA DE PÓS-GRADUAÇÃO EM REDES DE COMPUTADORES
MESTRADO PROFISSIONALIZANTE EM REDES DE COMPUTADORES

Camilo Telles Pereira Santos

5CAM: Sistema de Captura Panorâmica de Vídeos para Anotação de Reuniões

Salvador – Bahia

Junho 2006

Camilo Telles Pereira Santos

5CAM: Sistema de Captura Panorâmica de Vídeos para Anotação de Reuniões

Dissertação apresentada à Universidade Salvador, como parte das exigências do Curso de Mestrado Profissional em Redes de Computadores, área de concentração em Tecnologias Web e Aplicações Distribuídas, para obtenção do título de “Mestre”.

Orientador: Prof. Dr. Celso Alberto Saibel Santos

Salvador – Bahia

Junho 2006

Para Bruna e Ró

AGRADECIMENTOS

O trabalho envolvido na elaboração de uma dissertação é realizado predominantemente sozinho, mas está longe de ser solitário. Uma série de pessoas ajudam, se preocupam, acompanham e sentem a tua falta.

Em primeiro lugar queria agradecer à minha família (Pai, Mãe, Manas) que desde pequeno me apoiaram em cada pequena decisão que eu tomei. E nos erros, me ajudavam a sacudir a poeira e dar a volta por cima.

Gostaria de agradecer a minha esposa, Roxana. Ela que com certeza sentiu muito a minha falta, enfurnado na frente do computador numerosas madrugadas e fins de semanas. Beijo grande.

Agradecer a Bruna, por neste quase primeiro ano e meio de vida estar sendo adorável e dormindo quase que as noites inteiras deixando o pai trabalhar. ☺

Gostaria de agradecer também ao meu orientador Dr. Celso Saibel que apesar da minha teimosia me deu apoio em todas as mudanças de percurso que eu tive no caminho do meu trabalho. Agradecer a Dr. Joberto que me apoiou na minha entrada na UNIFACS e a Dr. Manoel Mendonça pelos debates, por me deixar sentar ao lado dele e pelo constante apoio.

Ao pessoal da Preview Computadores, André Brasil e Ricardo Freire sem eles o hardware nunca teria saído.

A Likiso Hattori que sempre foi um mestre.

Bruno Kraychete, sempre sabia responder a pergunta mais cabeluda em C++.

Ao pessoal que trabalha comigo, fiquem tranqüilos, vai rolar o almoço terminal.

Aos meus amigos da Mandinga (Cláudio, Guilherme, Ronaldo, Fredie, Pedro Flávio, Luciano, Carlos André, Matheus) e da Dvorak (Pinho, Boullosa , Joniel, Neca, Luciana, Pontual) pelas boas risadas, idéias e discussões.

RESUMO

Em qualquer ambiente corporativo, as decisões do dia a dia devem ser consensuadas e debatidas entre os membros da organização. Um dos mecanismos mais freqüentes nesse processo é o das reuniões. Este trabalho tem como objetivo a construção de uma câmera panorâmica para ser utilizada na captura de reuniões. Com esta câmera colocada no centro de uma mesa, os participantes da reunião são capturados em vídeo em tempo real. O vídeo da captura resultante pode então ser armazenado ou transmitido via IP para outros locais.

ABSTRACT

In all corporative environments the everyday decisions have to be consensed and discussed among the organizations members. One of the most frequently applied mechanisms in this situation are the meetings. This work focuses on the building of a panoramic camera to be used in meeting capture. When this camera is placed at the center of a meeting table, all the participants will be captured in video. The resulting video can be stored or transmited by IP network for other places.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1 INTRODUÇÃO

As câmeras de vídeo tradicionais têm uma limitação em relação ao campo de visão. Mesmo com o uso de lentes grandes angulares, não se consegue obter uma visão de 360º. A possibilidade de enxergar todos os lados ao mesmo tempo tem aplicações nas áreas de robótica {DeSouza, 2002 #16}, (Oh and Hall 1988), segurança (Mittal and Huttenloche 2000), captura e anotação de reuniões (Cutler, Rui et al. 2002), entre outras.

Este trabalho tem como objetivo construir um sistema para a captura de vídeos panorâmicos de reuniões. O requisito principal é obter um equipamento que seja leve, barato e de fácil reprodução, sem o uso de um sistema ótico complexo. Este equipamento deve ter um campo de visão horizontal de 360º ou próximo a isso. Com este equipamento colocado no centro de uma mesa de reunião torna-se possível capturar ao mesmo tempo todos os seus participantes. O vídeo capturado pode, se necessário, ser anotado utilizando softwares de anotação, como o desenvolvido em (Neto 2005).

Com a captura e anotação da reunião, o processo de tomada de decisões e suas sutilezas podem ser armazenados. São facilitados a transmissão do conhecimento e o entendimento

das decisões tomadas em uma organização. Participantes que não puderam estar presentes, seja por restrições de tempo ou de presença física, podem assistir ao vídeo e entender o processo decisório, que não está registrado em ata. Dessa forma, organizações que necessitem de maior controle nos seus processos decisórios[1] poderiam registrar reuniões para fins de auditoria, por exemplo. Este tipo de equipamento já tem alguma atenção da comunidade como podemos ver nos trabalhos de (Cutler, Rui et al. 2002) e de (Foote and Kimber 2000).

1.1 Objetivos do trabalho

O objetivo principal deste trabalho é o desenvolvimento de um sistema de captura omnidirecional, denominado 5Cam, para captura de reuniões com custo reduzido e sem a utilização de um conjunto de espelhos na sua construção.

Como objetivo secundário, a parte de otimização do código, para que o vídeo gerado pela câmera tenha uma taxa a partir de 15 quadros por segundo, como forma de assegurar uma qualidade mínima para o fluxo de vídeo gerado.

1.2 Justificativa

Uma boa parte do dia a dia das pessoas dentro das organizações é empregada na participação em reuniões. Muitas vezes, por problemas de tempo ou distância, nem todos os participantes podem estar presentes às reuniões. Em outros casos, é do interesse da pessoa participar da reunião para saber dos detalhes das decisões tomadas, pois estas decisões o afetam, mas não necessariamente deseje interferir no processo.

[1] Normas como a Sarbanes-Oxley tendem pedir este tipo de registro.
(http://www.legalarchiver.org/soa.htm)

A 5Cam tem como objetivo atender a estes cenários. O vídeo da reunião pode ser armazenado em disco para ser utilizado posteriormente. A reunião pode ser anotada e indexada para que os usuários do vídeo possam somente assistir os trechos específicos da reunião que sejam do seu interesse.

O vídeo também pode ser transmitido via Internet para um participante que não esteja presente fisicamente e que pode utilizar a própria rede para interferir na reunião utilizando tecnologias como Voz sobre IP, mensagens instantâneas ou canais de bate papo.

1.3 Organização

Esta dissertação está dividida em quatro capítulos, além dessa introdução. No Capítulo 2 serão analisados o problema da captura panorâmica de vídeo, os tipos de câmera existentes e suas características. No Capítulo 3 vamos demonstrar a arquitetura da 5Cam e do seu sistema acessório de calibração, o 5CamCap. No capítulo, os problemas específicos para o desenvolvimento da solução com várias câmeras serão apresentados assim como as soluções adotadas.

O Capítulo 4 trata da implementação da 5Cam. Neste capítulo serão apresentadas as tecnologias que estão envolvidas na implementação do 5Cam descrevendo o software e hardware envolvido. No Capítulo 5 são apresentados conclusões, limitações e possíveis desdobramentos do trabalho.

Capítulo 2 CAPTURA PANORÂMICA DE VÍDEO

2.1 Introdução

As câmeras de vídeo tradicionais têm uma limitação do campo de visão de tipicamente 45º. Existem uma série de aplicações que dependem ou se beneficiam de uma câmera que tenha um campo de visão maior ou, até mesmo, que tenha uma visão em todos os ângulos possíveis. Uma câmera que tenha um campo de visão com 360º é chamada de omnidirecional.

Existe uma literatura abrangente em relação às aplicações possíveis das câmeras com largo campo de visão. Na robótica, estas câmeras são utilizadas principalmente para navegação (DeSouza and Kak 2002), (Oh and Hall 1988), (Murphy 1995), (Gaspar, Winter et al. 2000). Elas são também utilizadas na engenharia reversa de instalações industriais (Chapman, Deacon et al. 2002), na área de videoconferência (Maita, Hashimoto et al. 2005), (Majumder, Seales et al. 1999), segurança (Mittal and Huttenloche 2000), (Onoe, Yokoya et al. 1998) e captura/anotação de reuniões (Cutler, Rui et al. 2002). Neste capí-

tulo serão apresentadas e analisadas algumas implementações de câmeras omnidirecionais.

2.2 Captura Omnidirecional de imagem

Uma sensor omnidirecional de imagem perfeito deve permitir a captura de um círculo completo atingindo 360º na horizontal e 360º na vertical e do ponto de vista teórico deve conter um único ponto de projeção, como se todos os raios de luz capturados passassem por um único ponto. Somente com estas características é possível se gerar uma imagem panorâmica ou perspectiva perfeita a partir da imagem omnidirecional conforme (Baker and Nayar 2001). Conforme a Figura 1 o círculo representa a imagem ominidirecional.

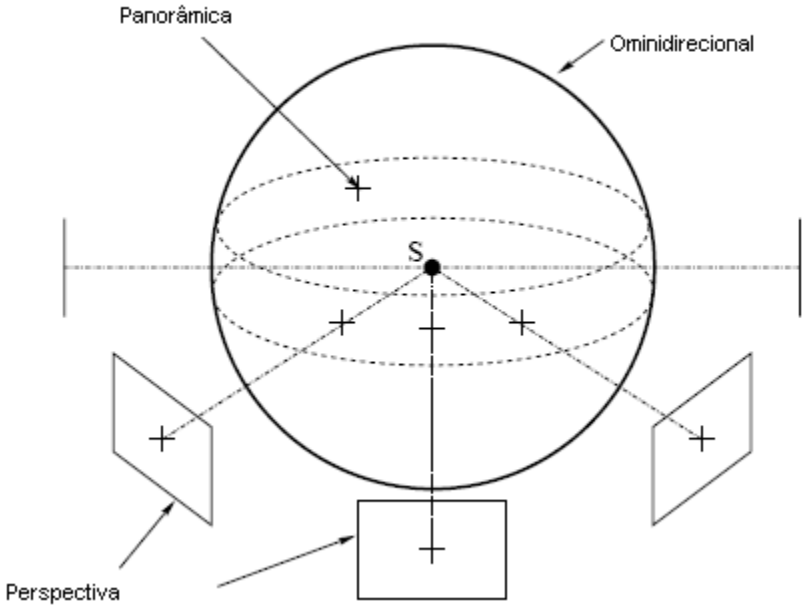


Figura 1 - Um sensor real omnidirecional. Permite a reconstrução das imagens panorâmicas e perspectivas. O ponto S é o centro de projeção (Winters 2001).

A imagem perspectiva pode ser gerada a partir de uma imagem omnidirecional aplicando uma projeção planar, também conforme a Figura 1. Imagens geradas desta forma mantêm a geometria perspectiva Estas imagens são importantes, pois são consistentes com a nossa

forma de enxergar o mundo. Além disso grande parte dos algoritmos de visão computacional assumem projeção perspectiva (Nayar 1997). Para gerar uma imagem panorâmica como representada na Figura 1, é utilizada uma projeção cilíndrica da imagem omnidirecional.

2.3 Tipos de câmeras omnidirecionais

Apesar do nome omnidirecional, não existe na prática nenhuma câmera que capture realmente 360º na horizontal e na vertical. Neste tópico iremos descrever os tipos de câmeras existentes, algumas implementações e características. A taxonomia utilizada neste tópico foi baseada em (Nayar 1997) e (Winters 2001).

2.3.1 Uma Câmera - Sistemas mecânicos rotatórios

A implementação utilizando uma câmera com um sistema mecânico rotatório é feita da seguinte forma:

1. Coloca-se uma câmera em cima de uma base rotatória com o eixo de rotação da base coincidindo com o centro do plano de projeção da câmera. Normalmente o local da câmera que é utilizado para conectar um tripé coincide com o eixo do plano de projeção da mesma.
2. As imagens obtidas, que tem projeção perspectiva, são fundidas em uma única imagem panorâmica. O campo de visão vertical é determinado pela lente utilizada, o campo horizontal é de 360º. A resolução vertical é igual à resolução do sensor utilizado. A resolução horizontal depende do mecanismo mecânico de rotação e do sensor utilizado.

Esta câmera não consegue capturar todos os lados simultaneamente, impedindo a utilização para aplicações de tempo real como robótica ou no caso de anotação de reuniões. Ela tem um ponto central de projeção e dependendo da qualidade da mecânica empregada, é possível se reconstruir qualquer imagem perspectiva a partir da imagem capturada. Este tipo de câmera é muito utilizado para panoramas estáticas que são utilizadas em sites ou em apresentações. Um exemplo deste tipo de construção pode ser encontrado em (Chen 1995). A Figura 2 apresenta um exemplo comercial deste tipo de construção da empresa IPIX Corporation. (http://www.ipix.com)

Figura 2 – Câmera panorâmica com base rotatória da IPIX Corporation. Nesta implementação a câmera está posicionada de forma a ampliar o campo de visão vertical (Fonte: http://www.ipix.com/).

2.3.2 Múltiplas Câmeras – Múltiplos Espelhos

Na implementação com múltiplas câmeras, múltiplos espelhos um conjunto de câmeras é apontado para um conjunto de espelhos. As câmeras deste tipo são chamadas de catadióptrica (Houaiss and Vilar 2001). *"Dioptria"* é a ciência das lentes enquanto *"cata"* é a ciência dos espelhos. O conjunto de espelhos atua de forma que as câmeras tenham o mesmo centro de projeção, tornando a câmera realmente omnidirecional. Um problema comum quando se utiliza abordagens com câmeras múltiplas é a necessidade de se combinar as imagens compensando os efeitos de luz e cor que podem aparecer.

Um primeiro exemplo desta técnica é de (Majumder, Seales et al. 1999) e que é demonstrado na Figura 3. Nesta abordagem foi utilizado um conjunto de 6 câmeras filmando 6 espelhos cuidadosamente dispostos. O campo de visão horizontal foi de 360º e vertical de 90º. A resolução alcançada foi de 2880 horizontal x 432 vertical.

Figura 3 – Múltiplas Câmeras – Múltiplos Espelhos, implementação realizada por uma equipe da University of North Carolina (Majumder, Seales et al. 1999).

Um exemplo comercial desta abordagem é da Full View[2] de Vic Nalwa[3], ex-pesquisador do Bell Labs. Na abordagem da Full View se utiliza quatro espelhos triangulares formando uma pirâmide. Cada espelho é filmado por uma câmera posicionada em cada lado da pirâmide a Figura 4.

Figura 4 – Múltiplas Câmeras – Múltiplos Espelhos da Full View (Fonte: http://www.fullview.com/).

[2] http://www.fullview.com/

[3] É importante ressaltar que o primeiro trabalho a demonstrar a possibilidade de se utilizar um conjunto de espelhos para obter o mesmo centro de projeção foi o do Sr. Nalwa na época em que era pesquisador do Bell Labs.

Esta abordagem tem como maior dificuldade a construção mecânica utilizada para garantir o alinhamento dos centros de projeção do espelhos, o resultado é um dispositivo volumoso composto de câmeras e espelhos.

2.3.3 Uma Câmera – Um Espelho

O método básico desta abordagem é apontar uma câmera orientada verticalmente para um espelho colocado acima dela. São utilizados espelhos elípticos, parabólicos, hiperbólicos ou esféricos. Todas estas soluções permitem um campo de visão de 360º na horizontal e dependendo do tipo de espelho utilizado de 70º a 120º vertical.

A discussão sobre os tipos de espelhos e lentes em configurações destes tipos de câmeras se encontra em (Baker and Nayar 2001) e (Winters 2001). As duas configurações mais importantes e práticas são utilizando espelhos parabólicos com lentes ortográficas e espelhos hiperbólicos com lentes convencionais. Nos dois casos a restrição de um único centro de projeção é atendida.

Espelhos Parabólicos: Quando são utilizados espelhos parabólicos, os raios de luz são refletidos na superfície do espelho e seguem paralelos entre si e perpendiculares em relação ao plano de imagem da câmera, ou seja, uma projeção ortográfica conforme a Figura 5. Para satisfazer a condição de um único centro de projeção, um espelho parabólico deve ser utilizado junto com uma lente ortográfica.

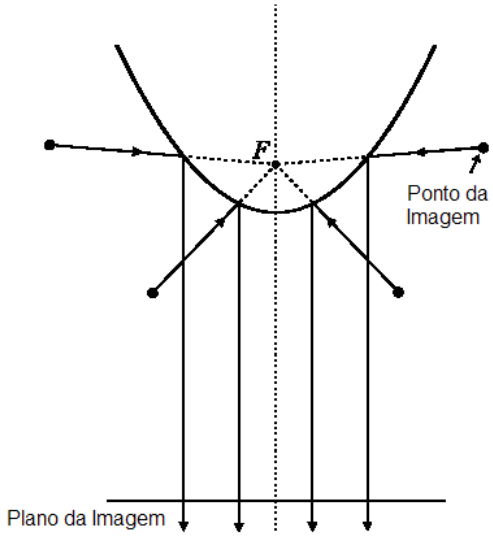


Figura 5 – Exemplo conjunto de espelho parabólico com lente ortográfica (Winters 2001).

A montagem de uma câmera deste tipo é simplificada. O eixo da câmera não precisa estar perfeitamente alinhado com o centro do espelho, e translações são toleradas. A distância da lente até a câmera também não é fator impeditivo nesta configuração.

Espelhos Hiperbólicos: Um conjunto com espelho hiperbólico somente consegue atender o requisito de um único centro de projeção se o ponto focal da lente convencional da câmera estiver posicionado no mesmo ponto que um dos focos da hipérbole, conforme a Figura 6. Esta restrição e a dificuldade de construção limita a utilização deste tipo de configuração.

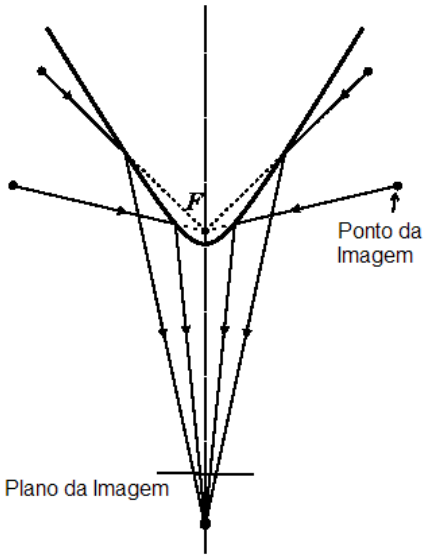


Figura 6 – Conjunto de espelho hiperbólico com lente normal (perspectiva) (Winters 2001).

Os outros tipos de espelhos e suas restrições são:

Elípticos: Apesar dos espelhos elípticos satisfazerem a condição de um único centro de projeção, caso o ponto focal da lente da câmera esteja posicionado em um dos centros da elipse, existe uma restrição do campo de visão, pois o espelho elíptico funciona no modelo apresentado na Figura 7. O ponto focal da câmera deve estar no ponto *p*. O plano de visão obtido ao final do processo é restrito não sendo interessante para aplicações práticas.

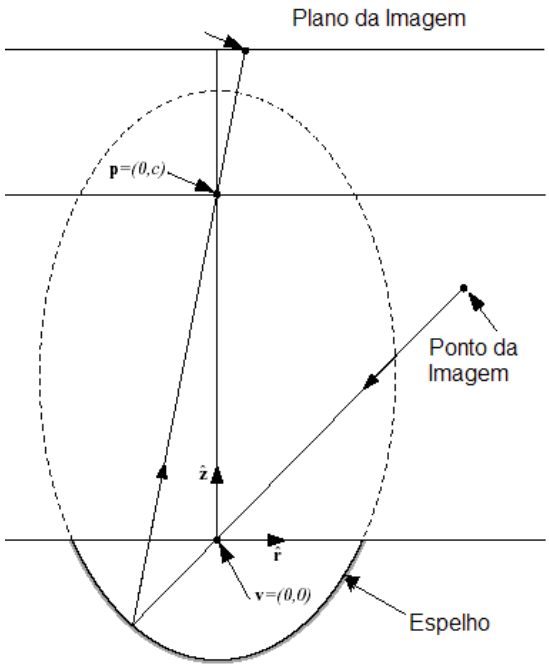


Figura 7 – Espelho elíptico (Baker and Nayar 2001).

Esféricos: Para funcionar com um único centro de projeção no espelho esférico, a lente da câmera deveria estar justamente no centro do espelho, impossibilitando completamente a sua aplicação.

Existe mais duas considerações em relação a utilização destes tipos de sistemas. A primeira é a resolução do sistema, que é limitado a um único sensor, limitando a quantidade de pixels obtidos na imagem. No caso de se utilizar um CCD convencional tem-se uma imagem panorâmica com 640x480 pixels. Além disso ,o espelho, por ser curvo, insere distorções no foco da imagem que comprometem ainda mais a resolução (Baker and Nayar 2001).

2.3.4 Uma Câmera – Múltiplos Espelhos

As câmeras desta categoria têm grande semelhança com as câmeras **uma câmera – um espelho** analisadas no tópico anterior. O principal motivador desta categoria é o seu ta-

manho reduzido em relação as implementações **uma câmera – um espelho**. Um estudo deste tipo de câmera se encontra em (Nayar and Peri 1999). A Figura 8 mostra um exemplo de uma configuração deste tipo de câmera.

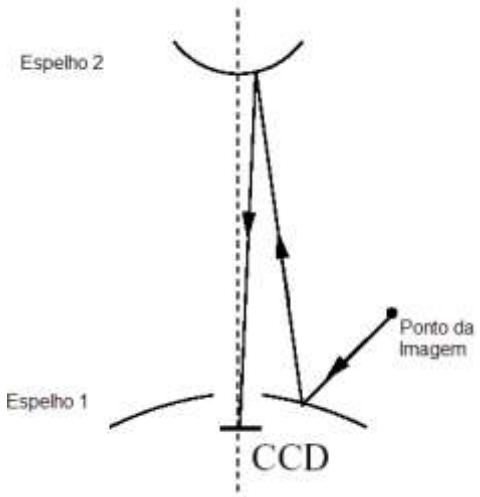


Figura 8 – Exemplo de única câmera – múltiplos espelhos (Winters 2001).

2.3.5 Múltiplas Câmeras

Esta é a solução adotada neste trabalho e será amplamente discutida no Capítulo 3. Nesta solução temos múltiplas câmeras apontando em várias direções diferentes de forma que o conjunto do campo de visão alcançado seja de 360º. As imagens adquiridas são combinadas para formar uma única imagem panorâmica. Esta câmera não tem um centro de projeção único o que na nossa aplicação não é relevante. Esta câmera tem uma resolução horizontal alta que é a combinação da resolução horizontal do conjunto de câmeras utilizado, a resolução vertical é igual a resolução do CCD, descontando as perdas de distorção e fusão de imagens.

Exemplos de contrução desta câmera podemos encontrar a Flycam construída por um laboratório da Xerox, descrita em (Foote and Kimber 2000) conforme a Figura 9 e a

RingCam construída pela Microsoft Research, descrita em (Cutler, Rui et al. 2002) conforme a Figura 10.

Figura 9 – Flycam (Foote and Kimber 2000)

Figura 10 – RingCam - (Nanda and Cutler 2001)

2.4 Conclusão do capítulo

Neste capítulo foram apresentados os principais tipos de câmeras omnidirecionais e suas características. As premissas no desenvolvimento do trabalho aqui apresentado são o de-

senvolvimento de uma câmera panorâmica de baixo custo, com vídeo em tempo real ou quase em tempo real, de fácil construção com uma boa resolução horizontal.

A Tabela 1 resume as características das câmeras apresentadas e as relações com o objetivo do trabalho.

Câmera	Custo	Fácil Construção	Resolução Horizontal	Centro Único de Projeção	Tempo Real
Única Câmera Sistema Mecânico Rotatório	Baixo	Sim	Alta	Sim	Não
Múltiplas Câmera / Múltiplos Espelhos	Alto	Não (dificuldade de alinhamento do conjunto câmera / espelho)	Alta	Sim	Sim
Única Câmera / Único Espelho	Médio	Não (dificuldade de obtenção do espelho e construção do equipamento)	Baixa	Sim	Sim
Única Câmera / Múltiplos Espelhos	Médio	Mesmas dificuldades da "Única Câmera / Único Espelho"	Baixa	Sim	Sim
Múltiplas Câmeras	Baixo	Sim	Alta	Não	Sim

Tabela 1 - Comparação das câmeras omnidirecionais.

Realizando a análise da tabela acima, é possível perceber que o único porém dentro dos critérios aqui apresentados para a opção Múltiplas Câmeras é a falta de um centro único de projeção. Como a aplicação de captura de reunião não necessita deste requisito, foi decidida a construção de uma câmera do tipo Múltiplas Câmeras. A descrição do modelo e sua implementação serão os assuntos dos próximos capítulos.

Capítulo 3 O Sistema 5CAM

Este capítulo vai apresentar os principais problemas para a implementação do sistema 5Cam e as soluções utilizadas. O Sistema 5Cam utiliza a abordagem de um conjunto de cinco câmeras para a formação da imagem panorâmica. O número de câmeras foi determinado pela característica do hardware utilizado na construção. Conforme será visto posteriormente, a lente utilizada tem um ângulo de visão horizontal de 80,95º. Para se cobrir um campo de visão de 360º com esta lente são necessárias, no mínimo, cinco câmeras. A abordagem de múltiplas câmeras está baseada nos trabalhos de (Cutler, Rui et al. 2002) e (Foote and Kimber 2000).

Para descrever o problema de forma ampla, será utilizada a Figura 11. Na parte superior dessa figura, temos a imagem capturada de duas câmeras contíguas. O problema em questão é como realizar as correções e fundir as imagens para que seja gerada uma imagem como aquela mostrada na parte inferior da mesma figura.

Figura 11 – Descrição do problema.

A solução do problema consiste da seguinte seqüência de etapas que resulta na produção de uma imagem panorâmica a partir de um conjunto de câmeras:

- **Captura de imagem de referência para calibração:** Captura de uma matriz de quadrados que será "distorcido" pelo conjunto câmera e lente. A distorção será medida para calcular o procedimento inverso.
- **Calibração da câmera:** A partir da imagem capturada pela câmera, achar a transformação inversa que gera novamente a matriz de quadrados original e seus pontos de registro entre as câmeras.
- **Geração da imagem panorâmica:** Aplicação da transformada inversa da etapa anterior nos fluxos de vídeo em tempo real, gerando o vídeo panorâmico.

3.1 Captura de imagem de referência para calibração

O objetivo do Sistema 5Cam é a geração de uma imagem panorâmica plana de 360º a partir de 5 imagens distintas cobrindo todo o campo de visão desejado. Caso as imagens de cada câmera estivessem no mesmo plano, o trabalho de gerar a imagem panorâmica seria simplesmente o de realizar translações nas imagens até que tivessem um acoplamento perfeito entre elas. Entretanto, como elas estão em planos diferentes, a fusão simples das imagens geraria uma imagem panorâmica distorcida. Isto se deve a uma série de fatores como o efeito barril, o efeito paralaxe e o serrilhamento que serão abordados nas próximas seções do texto.

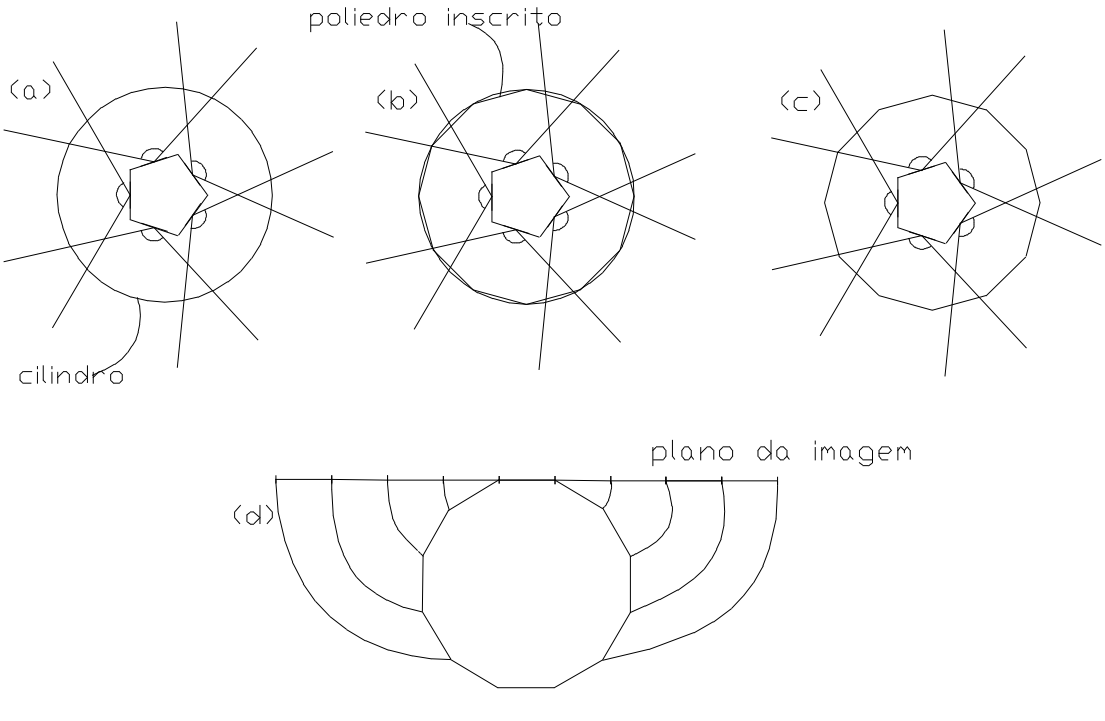


Figura 12 – (a) Imagem Panorâmica a ser registrada, representado pelo cilindro, e campo de visão das câmeras. (b) Poliedro inscrito ao cilindro correspondente da imagem panorâmica. (c) Poliedro sem o cilindro. (d) Planificação das arestas do poliedro no plano de imagem. Este plano de imagem corresponde a imagem panorâmica.

Para compensar os efeitos barril e paralaxe, gerando a imagem panorâmica, foi adotado a abordagem da Figura 12. Capturar a imagem panorâmica formada pelo cilindro da Figura 12a. Para realizar a calibração, foi construído um dispositivo que será descrito no protótipo que simula a inscrição de um poliedro no cilindro conforme a Figura 12b. Cada face deste poliedro é uma coluna de quadrados. O procedimento final é representado pela Figura 12d com a planificação das faces no plano de imagem panorâmica.

3.1.1 Distorção na captura da imagem pela câmera. (Efeito Barril)

O modelo teórico de uma câmera é chamada de *pinhole câmera* (Yu and McMillan 2004). Esta câmera teria um configuração semelhante à Figura 13 sendo constituída de uma caixa ou cilindro, com um furo de um lado e o plano que recebe a imagem invertida do outro.

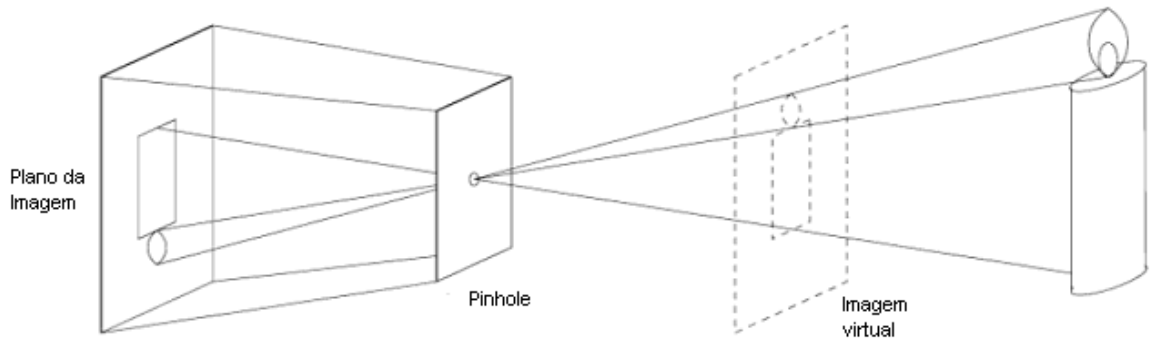


Figura 13 - Pinhole Câmera (Lowe 2004)

A luz caminha em linha reta e o furo, teoricamente, permite que somente um raio de luz oriundo de cada ponto do objeto passe pelo furo e incida no outro lado da câmera formando a imagem. As câmeras baseadas neste princípio são livres de distorções lineares e tem um foco infinito. Existem dois problemas com este tipo de câmera. A resolução da imagem dessa câmera está relacionada com o tamanho do furo da câmera. Se o furo for acima de um determinado tamanho, regem as leis da ótica geométrica e a imagem é des-

focada, pois o raio de luz resultante no plano da imagem é proporcional a distância focal e ao tamanho do furo, como na Figura 14.

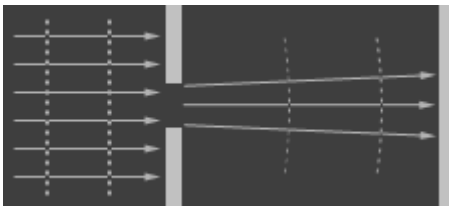


Figura 14 - Furo grande o suficiente para atuar a ótica geométrica (Lowe 2004)

Caso o furo seja muito pequeno o raio de luz sofre difração ótica por causa da característica onda do raio de luz conforme a Figura 15.

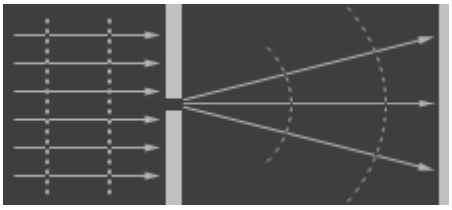


Figura 15 - Furo pequeno suficiente para atuar a difração ótica (Lowe 2004)

O tamanho ideal do furo é superior ao limite em que ocorre a difração, ficando no limite entre a ótica geométrica e os efeitos da difração ótica. Caso se resolva o problema do tamanho do furo, a quantidade de luz necessária para obter uma boa imagem fica prejudicada pela pouca quantidade de luz que entra pelo furo.

Por causa disso, as câmeras normalmente inserem lentes no conjunto. A lente tem a função de focar a imagem em um plano, com a vantagem de aproveitar todos os raios de luz oriundos do objeto, aumentando assim a quantidade de luz capturada conforme a Figura 16.

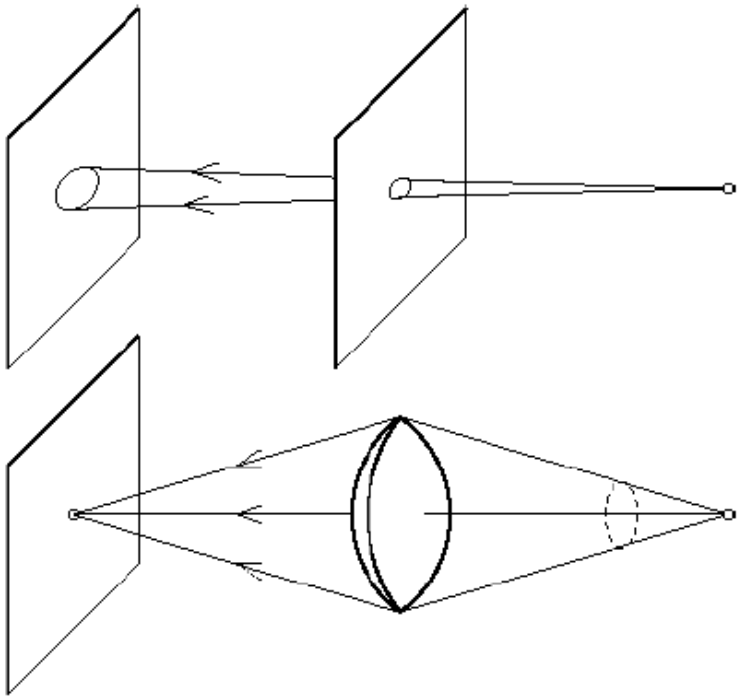


Figura 16 - Uso de lentes na câmera. (Lowe 2004)

Para ocorrer um foco perfeito o formato da lente deveria ser o de um polinômio de grau 4 o que formaria uma lente chamada de aesférica (Feynman, Leighton et al. 1989). Porém, por questões de facilidade de fabricação e custo, normalmente as lentes disponíveis são esféricas. As lentes esféricas introduzem erros na captura da imagem como o efeito barril[4] aqui demonstrado na Figura 17.

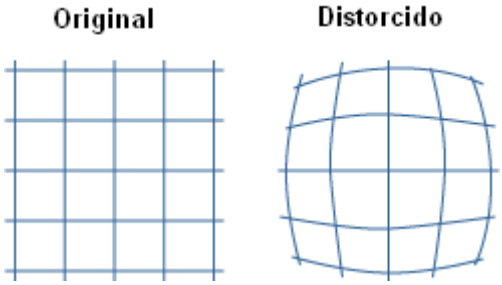


Figura 17 - Distorção da lente esférica grande angular[5]

[4] Barrel effect em inglês.
[5] Fonte: http://www.mellesgriot.com/glossary/wordlist/glossarydetails.asp?wID=102

O efeito barril ocorre pois, quando a imagem se afasta do centro ótico da lente, diminui o índice de aumento causando a forma abaulada da Figura 17. Caso estivesse sendo capturada uma imagem de um círculo, a imagem capturada não teria nenhuma alteração perceptível, pois a distorção causada pela lente esférica é radial. Na Figura 17 mostramos o resultado da captura de uma grade de quadrados. Conhecendo as características da imagem original, uma série de linhas verticais e horizontais paralelas, podemos utilizar a imagem resultado para calcular as distorções inseridas pela lente esférica e compensar estas distorções no sistema.

A Figura 18 demonstra o resultado do procedimento anterior. Na parte superior da figura temos a captura de duas câmeras com a imagem da matriz de quadrados utilizado para calibração superposta. Podemos observar que a última coluna da primeira imagem é correspondente à primeira coluna da imagem posterior. Esta intersecção é utilizada para a fusão das imagens e calibração entre as câmeras. O passo seguinte é o mapeamento de cada imagem inscrita nos quadriláteros oriundos das câmeras, nos quadrados da matriz de destino que forma a imagem panorâmica, representada pela parte inferior da Figura 18.

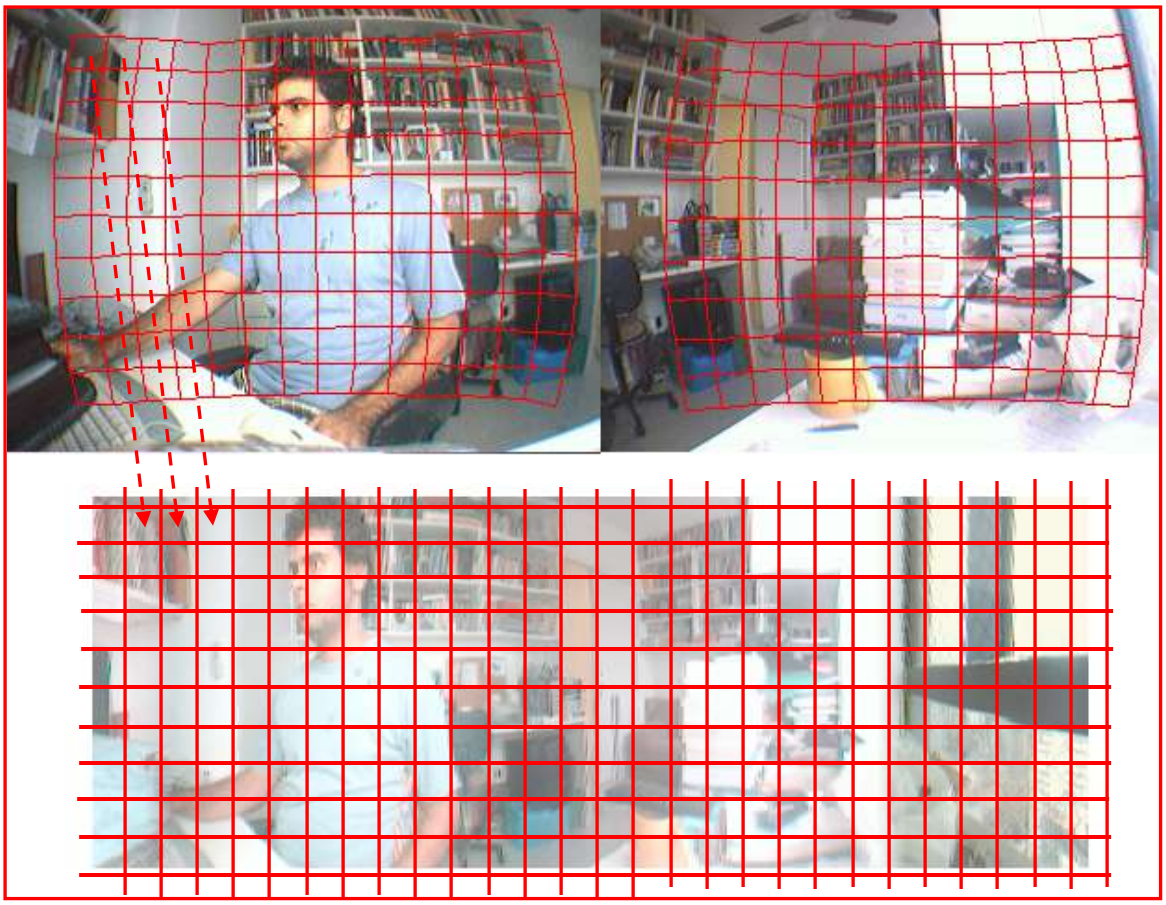


Figura 18- Mapeando as imagens capturadas na imagem final.

3.1.2 Efeito Paralaxe

O efeito paralaxe[6] (Woods, Docherty et al. 1993) deve ser mencionado, apesar de não ter grande impacto na aplicação. Ele ocorre porque as câmeras não compartilham do mesmo centro de projeção. Conforme podemos observar na Figura 19, as imagens da mesma esfera capturadas pelas duas câmeras são diferentes. Quanto mais próximo da câmera o objeto estiver, mais acentuado será este efeito. Isso pode ser observado pela representação da imagem capturada abaixo de cada câmera. No nosso, caso este efeito ocorre nas colunas que realizam a intersecção entre as duas imagens. Este efeito foi desprezado por (Foote and Kimber 2000) de forma similar a abordagem adotada.

[6] Parallax Effect em inglês

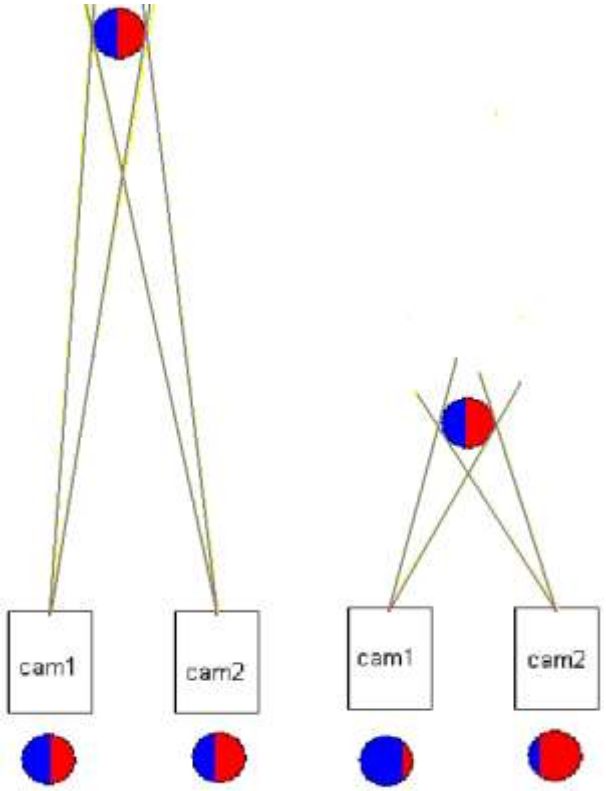


Figura 19 - Efeito estéreo

3.2 Calibração da câmera

A etapa Captura da Imagem de Referência para Calibração fornece uma lista de quadriláteros que devem ser planificados para a matriz de quadrados que formam a imagem panorâmica. Em outras palavras, as imagens contidas em cada destes quadriláteros devem ser mapeadas para a matriz de quadrados que forma a imagem. Este processo é chamado de distorção de imagens ou *image warping* (Heckbert 1989).

O objetivo da distorção de imagens é mapear uma imagem origem, também chamada de textura, para uma imagem destino. No caso estudado, estamos considerando que a imagem origem está em um quadrilátero, similar a qualquer um daqueles mostrados na parte superior da Figura 18. Esses quadriláteros devem ser mapeados em quadrados correspon-

dentes, conforme indicam as setas na imagem resultante da fusão das duas câmeras. Conforme é possível observar na Figura 20, o problema da distorção de imagens consiste em mapear cada ponto da textura, normalmente caracterizado pelas coordenadas (*u*,*v*) na, imagem destino, que normalmente utiliza a notação (*x*,*y*)[7]. Depois de encontrada a relação de mapeamento entre os pontos da textura e a imagem objetivo, é copiado a cor do ponto (*u*,*v*) no ponto (*x*,*y*).

Existem algumas técnicas de distorção de imagem.As mais comuns são as transformações lineares (cizalhamento, escalamento, rotação e reflexão), afins (composta das transformações lineares com translação), bilineares e perspectivas (Heckbert 1989).

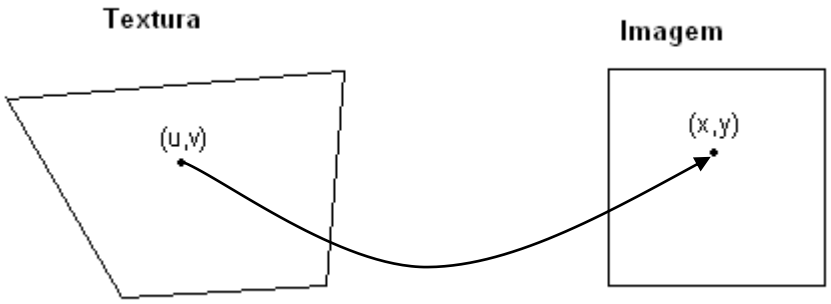


Figura 20 - Distorção de imagens (*image warping*)

[7] A descrição do processo foi simplificada, pois os efeitos relacionados a quantização não foram levados em conta até este ponto.

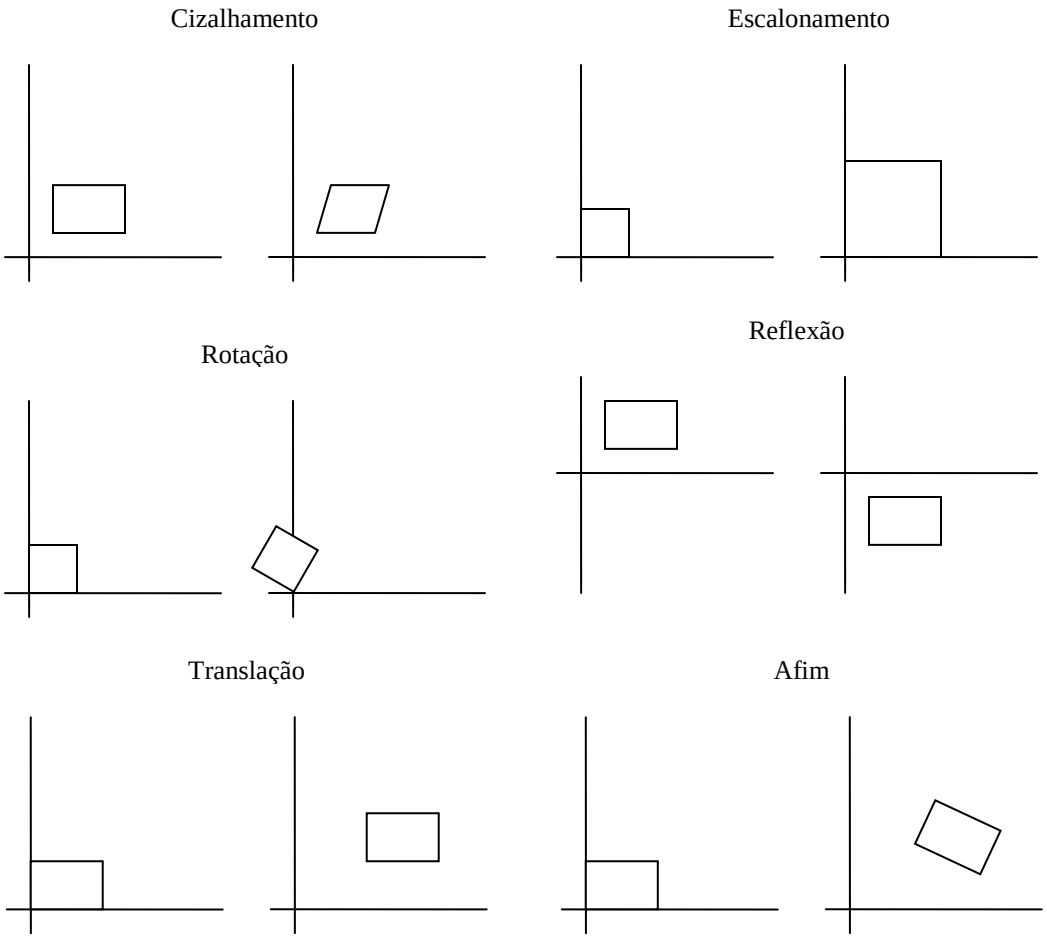


Figura 21 – Transformações lineares (cizalhamento, escalamento, rotação e reflexão) e afins (composta das transformações lineares com translação),

Conforme pode ser observado na Figura 21, as transformações lineares e afins tem a limitação de manter as linhas de um quadrilátero sempre paralelas. Elas são definidas por três pontos na imagem de origem e três na imagem de destino, perfazendo um sistema com seis graus de liberdade. Como no nosso caso temos o mapeamento de qualquer quadrilátero em um quadrado, não é possível utilizar as transformações lineares ou afins. Assim, é necessário utilizar a transformação bilinear ou perspectiva que permite o mapeamento de um quadrilátero qualquer em outro. A transformação bilinear não preserva as diago-

nais da imagem prejudicando a sua utilização em algumas situações, deixando como única escolha possível a transformação perspectiva (Heckbert 1989).

A transformação perspectiva é uma projeção de um plano em outro passando por um ponto central de projeção semelhante a uma câmera pinhole. A forma geral da transformação perspectiva é a seguinte: (Heckbert 1989):

$$x = \frac{au + bv + c}{gu + hv + i}, y = \frac{du + ev + f}{gu + hv + i} \qquad \text{(eq. 1)}$$

A manipulação é simplificada utilizando a notação de matrizes com coordenadas homogêneas:

$$(x' y' w) = (u' v' q) \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix} \qquad \text{(eq. 2)}$$

onde $(x,y) = (x'/w, y'/w)$ para $w \neq 0$

$(u,v) = (u'/q, v'/q)$ para $q \neq 0$.

Apesar da presença de 9 graus de liberdade, o mapeamento é homogêneo. Por causa disso, qualquer múltiplo escalar desta matriz retorna um mapeamento equivalente. Como somente existem 8 graus de liberdade no mapeamento perspectivo, quatro pontos na origem e quatro no destino, podemos assumir sem perda de generalidade que $i=1$ exceto no caso especial que $(u,v) = (0,0)$.

Nota-se ainda que a transformação perspectiva é inversível. Utilizando a Figura 22 podemos inferir isso. Conforme comentado anteriormente, a projeção perspectiva tem o mesmo modelo que uma *câmera pinhole*. Uma matriz *I* representaria a transformação perspectiva que realiza o mapeamento da imagem virtual da vela para o plano da imagem dentro da câmera.

Mas se considerarmos que por trás do plano da imagem fosse colocado uma luz, e a imagem antes capturada estivesse fixada em um meio transparente, o sistema ótico iria projetar o conteúdo do plano da imagem de volta para a imagem virtual. Esse processo realiza exatamente a operação inversa da matriz *I*, a qual é representada na figura pela matriz *I'*.

A matriz *I'* é obtida a partir do cálculo da inversa ou a matriz adjunta da matriz *I*. No caso de coordenadas homogêneas a matriz adjunta pode ser utilizada no lugar da matriz inversa, pois uma é múltipla escalar da outra (Heckbert 1989), (Anton and Rorres 2001).

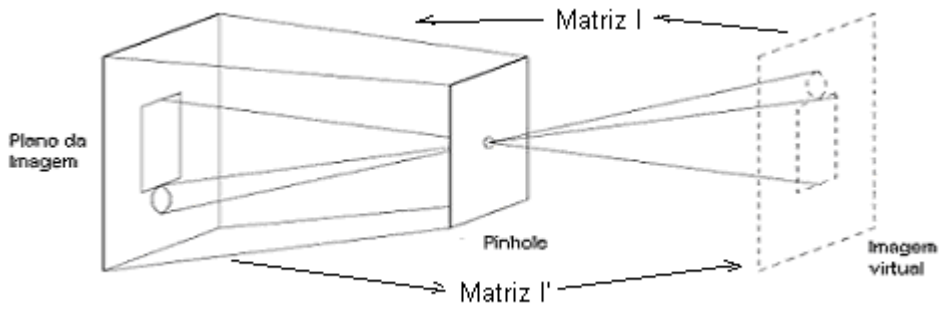


Figura 22 - Mapeamento perspectiva e sua inversa

Existe um ponto adicional que merece ser analisado sobre a projeção perspectiva. As câmeras utilizadas neste trabalho têm semelhança com o modelo *pinhole*. Ao capturar a imagem da matriz de quadrados, desconsiderando os efeitos da lente esférica, as câmeras provocam uma distorção perspectiva. Ao se aplicar à distorção perspectiva na imagem capturada, mapeando cada quadrilátero em um quadrado, está justamente se realizando o processo de mapeamento inverso ao que foi aplicado pelo conjunto de câmeras. Este fato vem exatamente ao encontro da decisão de se utilizar à distorção perspectiva no projeto. Ou seja, seguindo a Figura 22 o processo de captura da imagem pela câmera é representada pela matriz *I*. Neste instante o objetivo passa a descobrir a matriz I^{1} que irá retornar

a imagem correspondente à matriz de quadrados original, eliminando os efeitos inseridos pelo conjunto câmera e lente. O modelo ideal deveria considerar a remoção do efeito de distorção radial antes de aplicar a transformação perspectiva conforme demonstrado por (Devernay and Faugeras 2001). Entretanto, para este trabalho, a utilização da distorção perspectiva se mostrou adequada, pois não se utiliza a imagem capturada, por exemplo, para fins de fotogrametria. Na fotogrametria todas as relações existentes na imagem devem manter as proporções para medições.

Na entrada desta etapa, partiu-se de uma lista de quadriláteros a serem mapeados dentro de uma matriz de quadrados, que servem de base para a formação da imagem panorâmica. A saída desta etapa é o conjunto de matrizes que representam a transformação de cada quadrilátero contendo a imagem oriunda da câmera para o quadrado correspondente na imagem panorâmica de destino.

3.3 Geração da imagem panorâmica

Com as matrizes de transformações resultantes da etapa anterior e os vídeos originais sendo gerado pelas câmeras, o próximo passo é processar os vídeos individualmente para obter o vídeo panorâmico. O processo é realizado a partir de cada quadrilátero associado a um *frame* do vídeo original, gerando o quadrado correspondente no *frame* do vídeo panorâmico final.

Existem duas abordagens para se realizar esta transformação: varredura da imagem origem ou varredura da imagem destino. No caso da varredura da textura, para cada ponto da textura (no caso, o quadrilátero), calcula-se o ponto correspondente na imagem destino e copia-se o valor da cor do ponto da imagem textura no ponto correspondente na imagem destino. No caso da varredura da imagem destino, realiza-se a mesma operação, po-

rém com a mudança de que se calcula o ponto na textura a partir do ponto da imagem destino, utilizando para isso a função inversa da varredura da textura.

A abordagem da varredura da imagem destino é a mais adequada em nossa implementação pelas seguintes razões (Heckbert 1989):

- **Facilidade da varredura:** Como é varrido um quadrado perfeito, são necessários somente dois laços de controle varrendo cada um dos eixos e realizando as operações de transformação e cópia no laço mais interno. Caso fosse utilizada a varredura da textura, deveria ter sido utilizada alguma técnica como, por exemplo, varrer um quadrado que englobasse a textura toda e utilizar uma função para detectar se o ponto estava ou não dentro da textura, aumentando a complexidade do código e o custo computacional.
- **Aproveitamento completo da varredura**: Quando se realiza a varredura a partir da imagem destino, todo ponto calculado é efetivamente utilizado. No caso de se utilizar uma varredura da textura podem ocorrer, por questões de perspectiva, que alguns pontos não sejam utilizados. Imagine em um caso extremo no qual se fosse aplicar uma textura qualquer em um plano com um ponto de fuga no infinito. Alguns dos pontos da textura que estivessem mapeados no ponto de fuga no infinito seriam desprezados desperdiçando o esforço computacional gasto na transformação conforme a Figura 23.

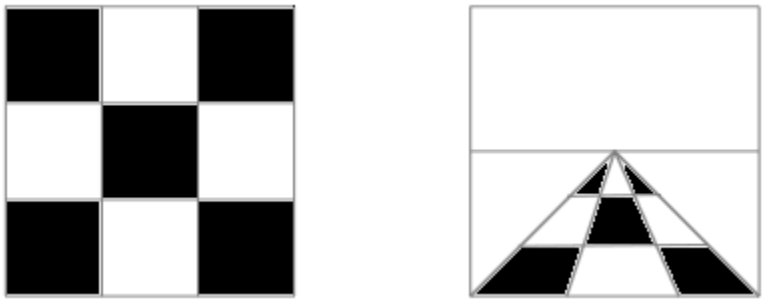


Figura 23 – Perda de pontos da textura no processo de mapeamento perspectivo.

Depois de encontrados os pontos correspondentes entre a imagem textura e destino, a forma mais simples de mapeamento é simplesmente copiar o ponto de cor da textura para o destino. Entretanto, esta forma não é a mais recomendável por causa dos problemas de serrillhamento na imagem resultante.

3.3.1 Efeito Serrilhamento

Para explicar o efeito serrilhamento, que é uma causa do processo de quantização e amostragem do sinal será utilizada a Figura 24. Na esquerda da figura se apresenta um sinal contínuo que está sofrendo um processo de digitalização. A escala do eixo x indica os instantes de amostragem usando a escala do eixo y como referência.

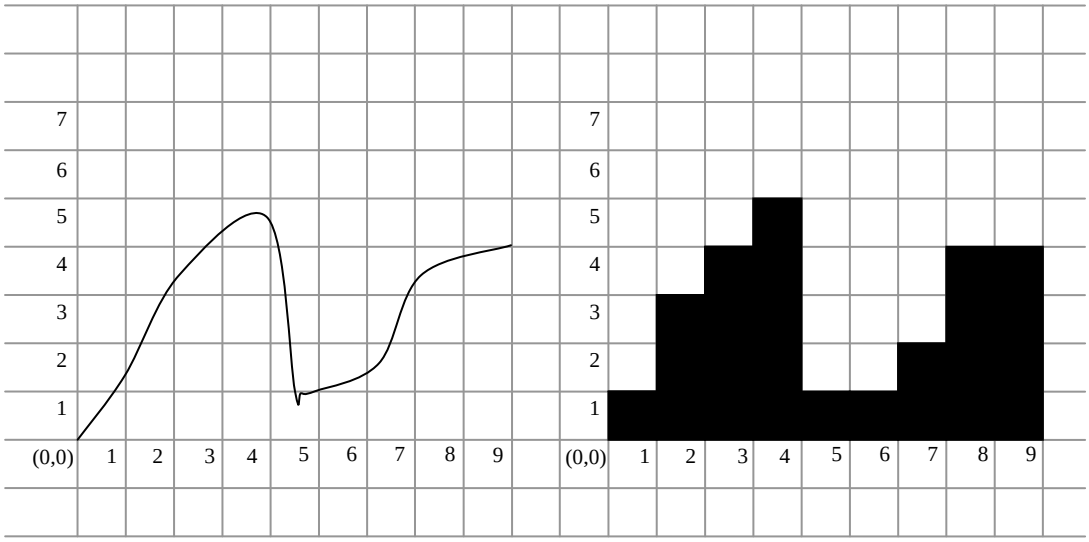


Figura 24 – Amostragem e Quantitatização

Os pontos da figura da esquerda são os pontos registrados na digitalização e a Tabela 2 representa os valores capturados.

X	1	2	3	4	5	6	7	8	9
Y	1	3	4	5	1	1	3	4	4

Tabela 2 - Valores do sinal contínuo

As diferenças entre os valores reais e os valores capturados são conhecidas como erros de quantitização.

Uma imagem digital tem as mesmas propriedades. A imagem é digitalizada na câmera usando uma matriz de elementos sensíveis à luz chamados de CCD[8]. Cada um destes elementos vai capturar somente um valor utilizando uma escala semelhante à nossa escala da amplitude na Figura 24. Nas imagens normalmente, se utiliza uma escala de 8 bits para cada componente da cor (vermelho, verde, azul) o que permite se distinguir 2^{24} valores de cor diferentes. A escala x do exemplo, é no caso da imagem, a resolução de cada

[8] CCD – Charged Coupled Device

dimensão. A câmera utilizada no protótipo tem uma resolução horizontal de 320 pontos e vertical de 240. Caso no momento da transformação, por exemplo, seja necessário obter o valor do ponto (25,5, 14,7) isso não será possível. No caso da implementação será aplicada aproximação simples que utiliza o valor do ponto mais próximo. Caso otimizemos a câmera para trabalhar em tempo real, pode-se utilizar uma interpolação bilinear que dá maior qualidade a imagem. A técnica bilinear também é utilizada em (Foote and Kimber 2000), (Szeliski 1996) e (Glasbey and Mardia 1998). Uma discussão detalhada dos problemas da quantização e amostragem na deformação de imagem é apresentada em (Heckbert 1989).

3.4 Conclusão

Neste capítulo foi apresentada a arquitetura da 5Cam. Foi apresentado o problema envolvido na geração da imagem panorâmica, a solução adotada e os efeitos que devem ser compensados (barril, paralaxe e serrilhamento).

Capítulo 4 5CAM: Implementação do Protótipo

Neste capítulo será apresentada a implementação do protótipo da 5Cam e do seu sistema de calibração chamado 5CamCap. A Tabela 3 relaciona os tópicos apresentados no capítulo anterior com as soluções adotadas na implementação do protótipo.

Problema a ser resolvido	**Solução**	**Descrição**	**Observação**
Captura de imagem de referência para calibração	Software **5CamCap** e base para captura	Software para plataforma Windows 5CamCap	Somente executado uma vez para captura dos quadriláteros de referência.
Calibração da câmera	Software **5Cam** e câmera panorâmica	Filtro do *DirectShow*, implementado como uma DLL, que constroi o vídeo panorâmico a partir dos vídeos originais e dos quadriláteros de referência	Executado durante o início do processo de geração somente uma vez.
Geração da imagem panorâmica			Executado em tempo real.

Tabela 3 - Relacionamento modelo x implementação

Como o protótipo foi desenvolvido na plataforma Windows utilizando o *DirectShow*, será iniciada a descrição da implementação do protótipo explicando este componente, seguindo o seguinte roteiro:

1. *DirectShow* –Descrição do *DirectShow* da Microsoft
2. 5CamCap – Descrição da implementação e funcionamento do software e da base de calibração de captura da imagem de calibração
3. 5Cam – Descrição da implementação do filtro *DirectShow* e do hardware que implementa o vídeo panorâmico.
4. Otimização – A implementação que é descrita no item anterior é de referência sem se preocupar com questões de desempenho. Nesta etapa será descrito o processo de otimização do código.

4.1 Arquitetura DirectShow

O *DirectShow* é um subsistema do Windows para reproduzir, capturar, gravar e manipular fluxos de mídia. O material deste tópico foi baseado em (Microsoft 2005), (Pesce 2003) e (Davies 2000).

Na Figura 25 podemos observar os seus principais pontos. O conceito básico no *DirectShow* é o filtro. Uma série de filtros interconectados formam grafo que manipula um ou mais fluxos de mídia. Existem três tipos de filtros:

- Filtro Fonte: Sempre inicia um grafo e funciona como uma fonte de um fluxo de mídia, que pode ter como origem um arquivo no disco, Internet (*streaming*) , uma webcam, uma entrada de aúdio, uma câmera digital etc.
- Filtro de Transformação: Transforma, combina, divide um fluxo de mídia. Exemplos de filtros de transformação: Conversores de um fluxo de vídeo RGB→YUV,

conversores que manipulam a resolução ou profundidade de um vídeo, filtros que combinam um fluxo de áudio com um fluxo de vídeo, sincronizando os dois e entregando um fluxo de áudio com vídeo único, o filtro 5Cam que captura cinco vídeos tradicionais e entrega um único vídeo panorâmico etc.

- Filtro de Apresentação: Funciona como o final de um grafo, apresenta o fluxo de mídia, seja na tela do micro, em uma saída de vídeo externa podendo também gerar um arquivo no disco, encaminhar o fluxo pela rede, etc.

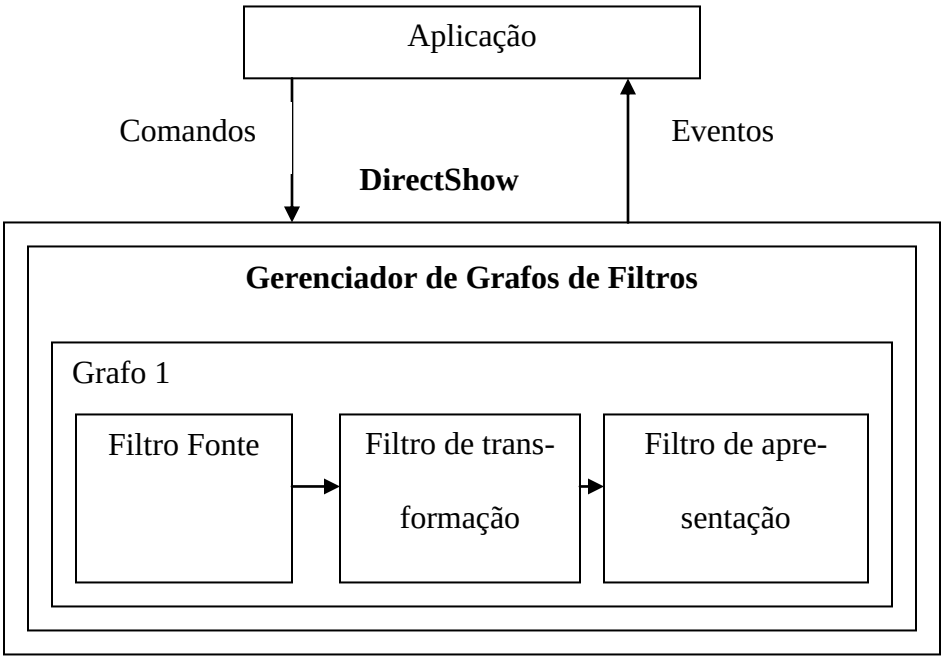


Figura 25 - Arquitetura *DirectShow*

Na Figura 25 o fluxo de mídia segue dentro do Grafo 1 que é composto por um filtro fonte até um filtro de apresentação passando por um filtro de transformação. Depois de criado o grafo de filtros se torna uma unidade funcional encapsulando os filtros que nele existem. Podem existir grafos que tenham qualquer combinação destes filtros. Podem

existir vários fluxos de mídia dentro de um mesmo grafo. Por exemplo, um grafo pode conter um fluxo de captura de áudio e um de vídeo separados, os quais são combinados para gerar um vídeo com áudio. Também é possível capturar um fluxo de vídeo e depois dividir este fluxo em dois outros os encaminhando para filtros de apresentação diferentes. Por exemplo, um dos filtros de apresentação poderia ter como destino a tela do micro e o outro a criação de um arquivo de vídeo em disco.

A construção e controle dos grafos ficam a cargo de um Gerenciador de Grafos de Filtros (GGG). A aplicação se comunica com o gerenciador por uma interface para a construção dos grafos. Quando o grafo está pronto, a aplicação envia comandos ao gerenciador como tocar, adiantar, pausar, voltar etc. O gerenciador se comunica com a aplicação por meio de eventos que indicam que a mídia chegou ao fim, ou que um erro aconteceu etc.

Os filtros são objetos COM (Component Object Model) (Microsoft 1998), e são tratados como elementos atômicos expondo uma interface pré-determinada. Cada filtro pode estabelecer conexões com outros filtros e negociar os tipos de fluxos que eles podem transmitir e aceitar. Os filtros devem aceitar os comandos de tocar, pausar, parar, adiantar, voltar desde que isso seja adequado para o tipo de mídia que está sendo tratado. Por exemplo, não é possível voltar um vídeo proveniente de uma *webcam*.

Os filtros sempre têm no mínimo um pino[9]. Os pinos funcionam como os pontos de conexão entre os filtros. Existem dois tipos de pino: os de entrada que recebem um fluxo de dados e os de saída que produzem um fluxo de dados para outros filtros.

Quando dois filtros estão sendo conectados, devem chegar a um acordo do tipo de fluxo que será transportado. Os pinos dos filtros gerenciam a negociação entre eles. Cada pino deve publicar a lista de fluxos de dados que ele pode tratar. Quando o Gerenciador de

[9] Do original em inglês *pin*

Grafos de Filtros tenta conectar um pino de saída, no pino de entrada se inicia o processo de negociação. O Gerenciador de Grafos de Filtros examina a lista de tipos de mídia que o pino de saída pode transmitir com a lista de tipos que o pino de entrada pode receber. Caso não exista um acordo, ocorrerá um erro na formação do grafo.

Depois de acordado o tipo de mídia que vai ser transmitida, deve-se chegar a um acordo sobre mecanismo de transporte. Se não se chegar a um acordo, novamente o processo falha. Chegando a um acordo, um dos pinos tem a responsabilidade de criar um alocador O alocador tem como finalidade gerenciar os *buffers* de dados que são utilizados na passagem do fluxo entre o pino de saída de um filtro para o de entrada do outro.

Ocorrendo um acordo entre o tipo de mídia e o mecanismo de transporte os dois filtros estão conectados pelos seus pinos. Esta operação deve ser repetida para todos os filtros do grafo de forma que exista um caminho entre o filtro fonte, passando por um ou mais filtros de transformação, e um filtro de apresentação. Quando isso ocorrer, será possível enviar comandos para o Gerenciador de Grafos de Filtro, que vai comandar os filtros individualmente para executar todas ações possíveis no fluxo de dados que estará sendo tratado.

4.1.1 Grafo de Filtros

O grafo de filtros organiza os filtros em uma unidade funcional. Depois de montado o grafo, o desenvolvedor pode desconsiderar os filtros individualmente e tratar o grafo como uma entidade única. Os comandos de controle de mídia são enviados para o grafo que gerencia os filtros. Além disso, os filtros devem estar sincronizados, pois eles estão lidando com amostras de mídia que devem estar sincronizadas entre si. Caso isso não ocor-

ra, por exemplo, poderá ocorrer uma desincronização entre o vídeo e áudio durante a execução de um filme.

Para isso, o grafo de filtros gera um relógio que é utilizado como referência por todos os filtros no grafo. Este relógio é utilizado para manter a sincronização e permite que os filtros mantenham o fluxo de mídia sincronizado enquanto as amostras passam de um filtro para outro.

Quando o programador envia um dos três comandos básicos – tocar, pausar e parar – o GGF retransmite este comando para os filtros. Todo filtro do *DirectShow* deve processar no mínimo estes três comandos. Por exemplo, enviar um comando de tocar para uma *webcam* vai iniciar um fluxo de mídia no grafo, enquanto o comando de parar vai interromper o fluxo e limpar todos os *buffer*s intermediários. O comando pausar funciona de forma semelhante, com a diferença que os *buffer*s intermediários não são esvaziados, permitindo o reinício imediato do fluxo de mídia.

4.1.2 O ciclo de vida de uma amostra de mídia

A Figura 26 buscar esclarecer o ciclo de vida de uma amostra de mídia no *DirectShow*

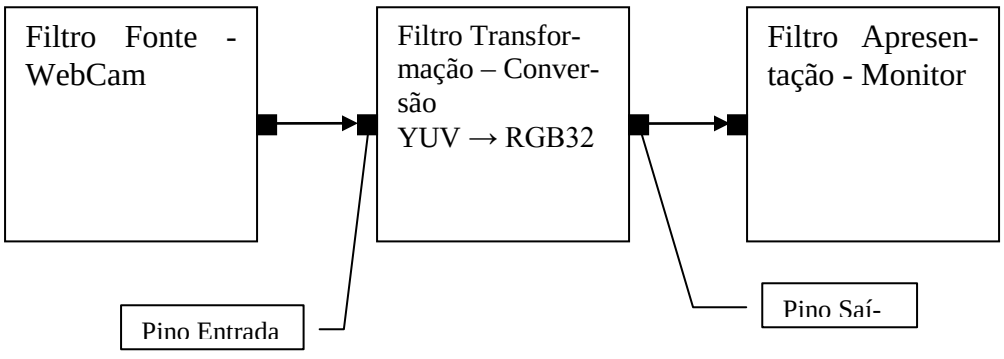


Figura 26 - Grafo de Filtros

O grafo anterior possui 3 filtros. O filtro fonte, que neste caso, é um filtro de captura e se comunica diretamente com o hardware da WebCam produzindo uma saída YUV. O filtro de apresentação recebe um vídeo RGB32, resultante da transformação do filtro intermediário, e tem como saída uma janela do sistema operacional Windows. Como existe uma incompatibilidade entre a saída do filtro fonte e a entrada do filtro de apresentação foi colocado um filtro de transformação de YUV para RGB.

Normalmente cada quadro do vídeo é armazenado em uma amostra do fluxo de mídia. Neste caso a função do filtro fonte é transformar os pacotes de dados oriundos da câmera em amostras do fluxo de mídia *DirectShow*. Cada uma das amostras tem um marca de tempo indicando o momento de sua geração e um valor de tempo mínimo e máximo para o seu processamento e apresentação.

A amostra segue para o filtro de transformação. Este filtro modifica o formato da codificação de cor de YUV para RGB que é necessário para o próximo filtro. Se for necessário ele também trata de questões como entreleçamento. A amostra segue então para o próximo filtro, o de apresentação.

Caso a amostra chegue no filtro de apresentação antes do tempo mínimo, o filtro aguarda até o momento adequado para apresentação. Caso contrário, se a amostra chegar depois do tempo máximo permitido ela é descartada. Caso a amostra chegue entre o tempo mínimo e máximo de apresentação é decodificada e apresentada em uma janela do Windows. O filtro de apresentação é o ponto final do grafo e o *buffer* de memória que armazenava a amostra é liberado para receber uma nova amostra.

Existe um caso em que os relógios são desconsiderados, como por exemplo, na conversão de um tipo de arquivo em outro. Neste caso, o filtro fonte seria um leitor de um arquivo

MPEG1, cujo fluxo resultante passaria por um filtro de conversão e depois enviaria para um filtro de apresentação que escreveria em um arquivo MPEG4. O grafo vai desconsiderar os relógios e vai processar o fluxo de mídia na máxima velocidade possível.

4.2 5CamCap

O 5CamCap é o conjunto de hardware e software que realiza a captura da imagem de referência simulando a inscrição do poliedro no cilindro conforme foi descrito no tópico 3.2 Calibração da câmera.

4.2.1 Hardware

O hardware do 5CamCap foi criado para simular a inscrição do poliedro no cilindro e pode ser visto na Figura 27.

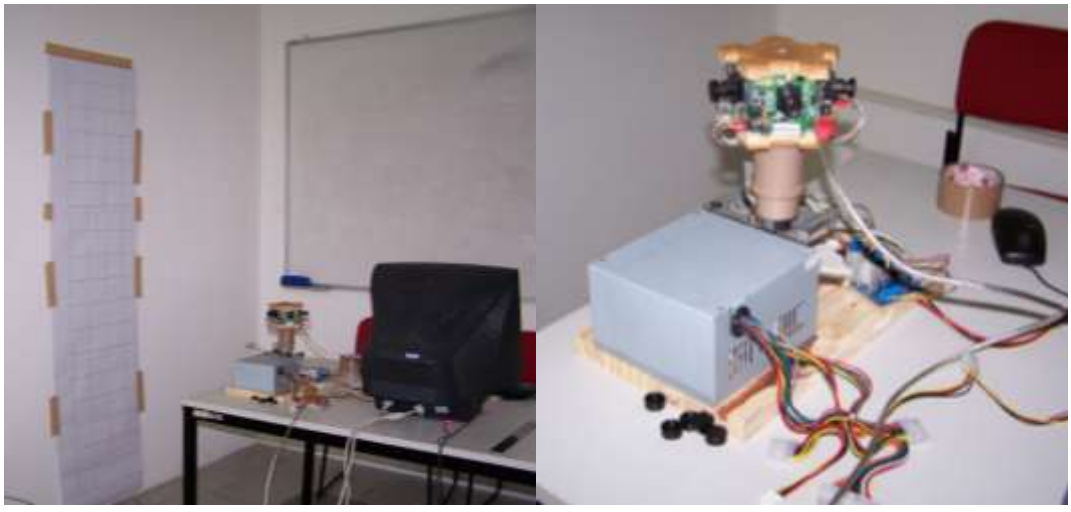


Figura 27 - Hardware do 5CamCap

O hardware é composto da câmera 5Cam montada sobre um, motor de passo com uma caixa de redução ligado a um PC pela porta paralela para comando. A câmera filma uma matriz de quadrados colocados a uma distância de cerca de dois metros. A distância de dois metros foi obtida de forma experimental, sendo suficiente para minimizar a distorção

por paralaxe. A matriz de quadrados tem a sua coluna central marcada para que sirva como a coluna de referência. Com este dispositivo simula-se a inscrição de um poliedro dentro de um cilindro.

Na Figura 30(a) pode-se observar o campo de visão da câmera com a coluna de referência marcada em vermelho. Na Figura 30 (c), pode-se perceber o momento em que as duas câmeras observam a mesma coluna e são marcados os quadriláteros. É com a informação desta coluna que será usada para combinar as duas câmeras conforme será demonstrado no item 4.3.2 – Software.

4.2.2 Software

A Figura 28 apresenta a tela resultante do software 5CamCap.

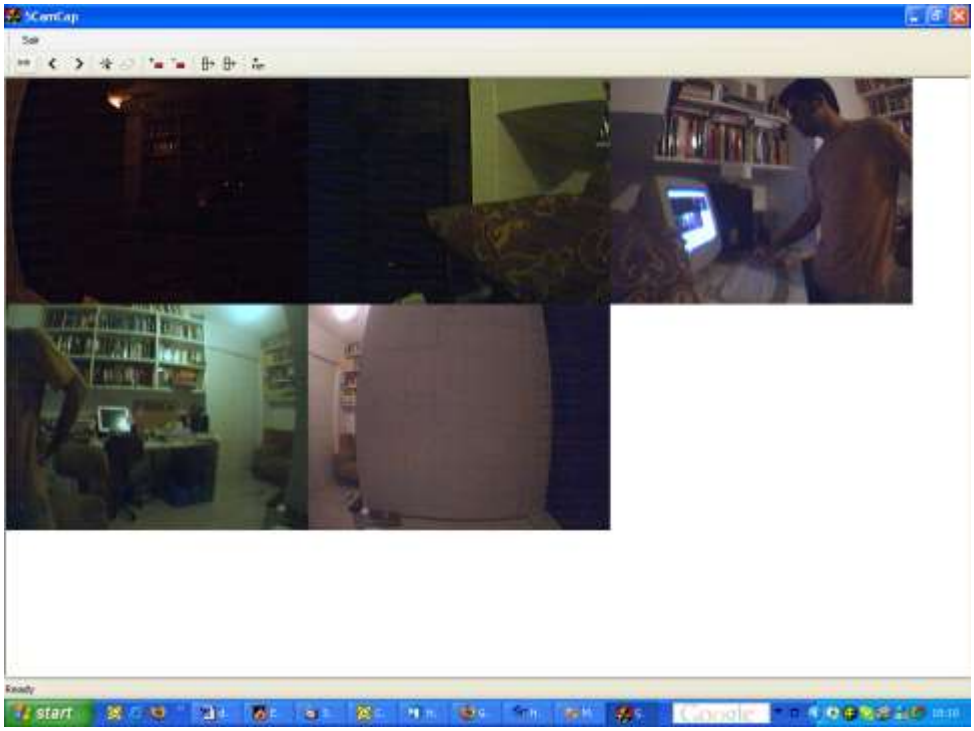


Figura 28 - 5CamCap

Internamente o 5CamCap mantém uma lista de pontos e de polígonos. No momento da inicialização das câmeras são gerados 5 grafos para captura dos vídeos. Os pontos e polí-

gonos são desenhados em 5 imagens em memória e combinados em tempo real com o vídeo.

4.2.3 Processo de Calibração

Com o software e hardware funcionando, pode se dar início ao processo de calibração, a Figura 29 e Figura 30 irão auxiliar na descrição do processo.

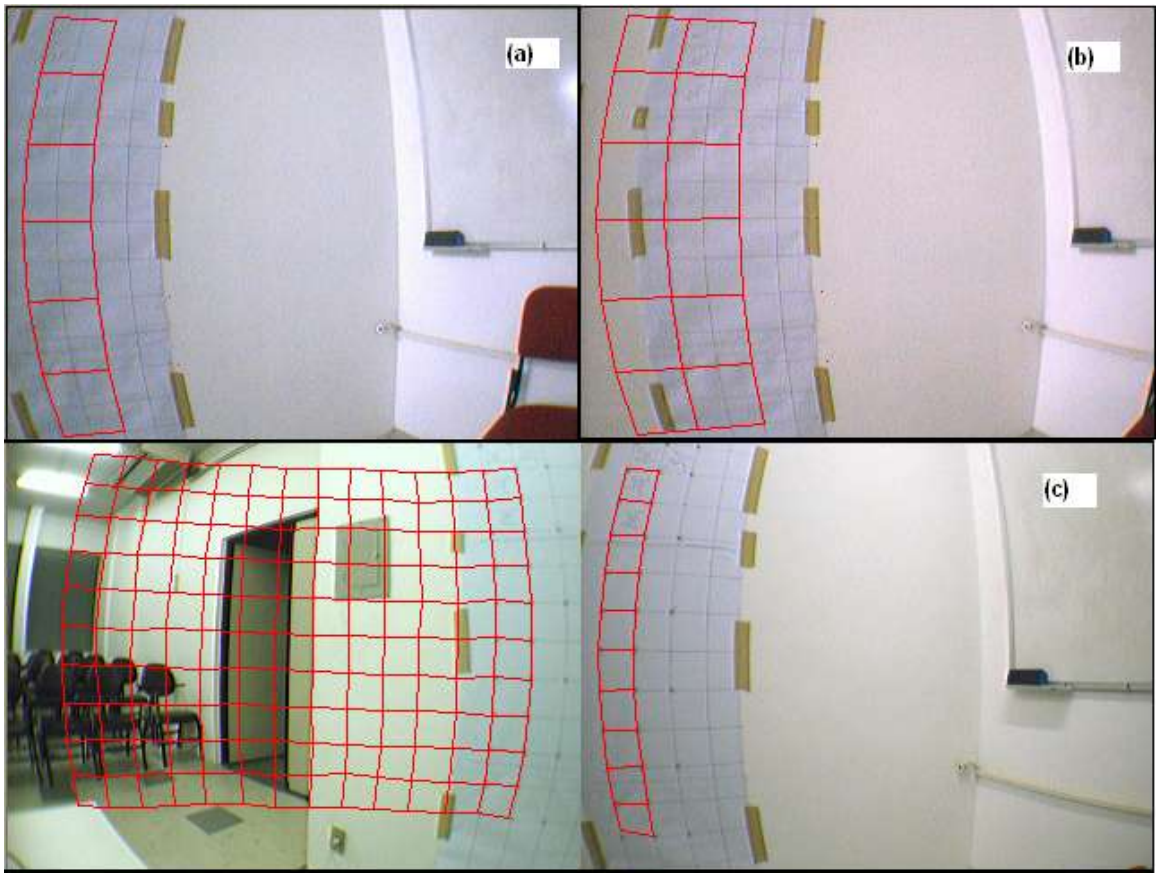


Figura 29 – Imagens do processo de calibração

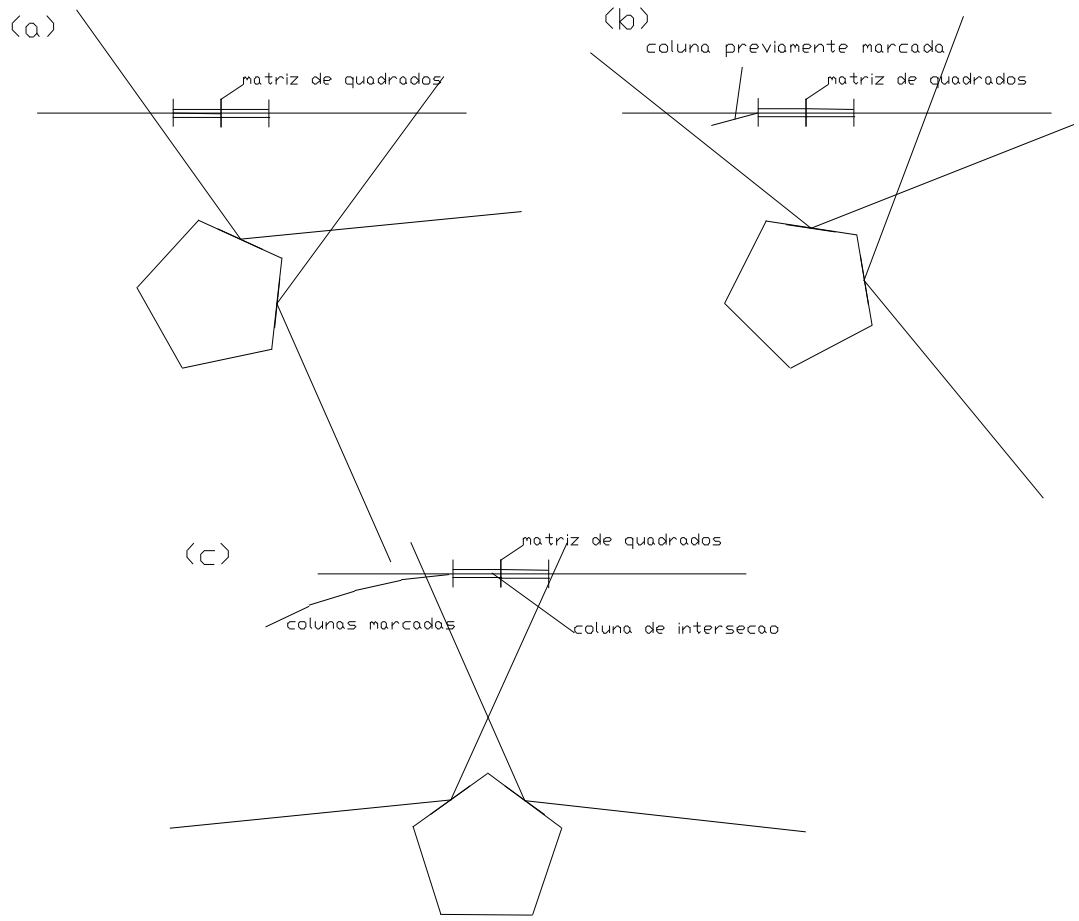


Figura 30 - Simulação da inscrição de um poliedro na circuferência, correspondente com as imagens da Figura 28

O processo se inicia com a coluna de referência da matriz de quadrados na posição mais à esquerda da primeira câmera. A partir desse posicionamento, marcam-se os pontos e os quadriláteros correspondentes utilizando o software 5CamCap, conforme podemos ver o resultado na Figura 29 (a).

Em seguida, gira-se a câmera até a coluna de referência estar com o seu lado esquerdo alinhado com a coluna de quadriláteros recém marcados no passo anterior. Na seqüência, marcamos estes novos pontos e quadriláteros conforme a figura Figura 29 (b). Essa seqüência de passos é repetida até completar o campo de visão da câmera com os quadriláteros conforme a figura Figura 29(c) a esquerda.

No momento que a coluna de referência estiver sendo visualizada por duas câmeras adjacentes, como na figura Figura 29 (c), é marcado as colunas que intersectam as duas câmeras e que serão utilizadas para o registro entre câmeras. A Figura 31 mostra o resultado de uma calibração completa.

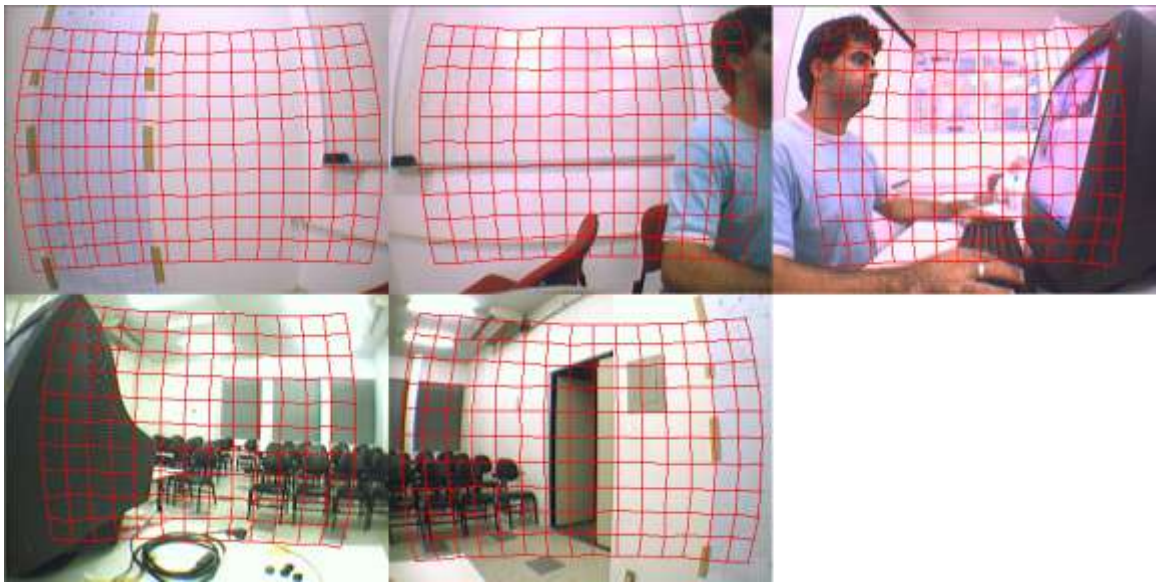


Figura 31 - Calibração Completa

Ao final deste processo, exportamos os quadriláteros capturados para serem utilizados na calibração da 5Cam. A forma da utilização destes quadriláteros será explicada no próximo tópico.

4.3 Implementação da 5Cam

5Cam é a implementação da câmera panorâmica e utiliza a lista de quadriláteros gerados pelo 5CamCap para a sua calibração. A implementação da 5Cam é dividido no hardware e software. O hardware é a composto de uma base de madeira com as cinco câmeras. O software é o filtro fonte do *DirectShow* que, a partir das 5 imagens geradas pelo hardware, gera a imagem panorâmica.

4.3.1 Hardware

O hardware da 5Cam é composto de 5 câmeras Unibrain Fire-i Board Digital Câmera (Figura 32) com lentes de distância focal de 2,1 mm. Estas lentes dão a câmera uma campo de visão horizontal de 80,95º.

Figura 32 - Unibrain Fire-i Board Digital Câmera

Esta câmera é adequada para aplicações embarcadas, com suporte a duas conexões IEEE 1394 (Firewire) e suporte IIDC DCAM (Association. 2001). Ela tem saída de até 640x480 e o fabricante já fornece estudos e informações para conexões de múltiplas câmeras e o impacto no consumo de banda do link IEEE 1394 no caso de múltiplas câmeras variando a resolução e número de cores (Unibrain 2005).

A câmera foi montada em uma base de madeira balsa e as conexões firewire foram ligadas em uma configuração de *daisy chain*. O resultado final pode ser observado na Figura 33.

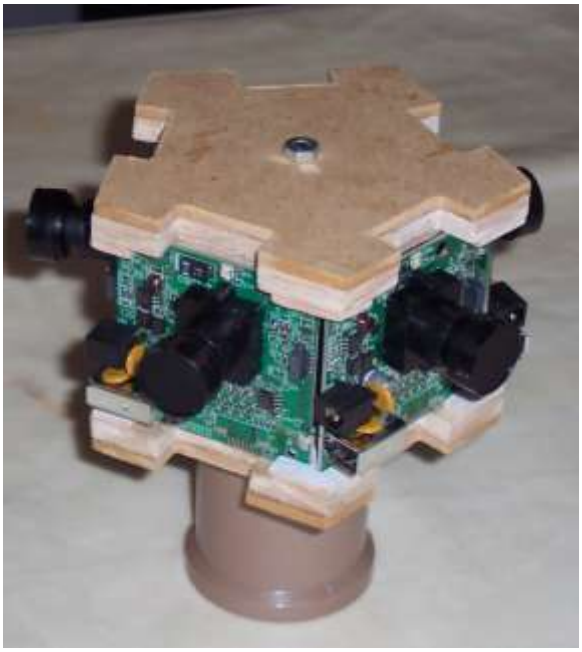


Figura 33 - Câmera 5Cam

4.3.2 Software

O software da 5Cam é um filtro fonte do *DirectShow*. É importante salientar que a implementação descrita neste capítulo não é a final. Ela foi a implementação inicial que depois será otimizada, conforme apresentado no tópico seguinte. Na Figura 34 se apresenta a arquitetura que vai auxiliar na descrição do filtro.

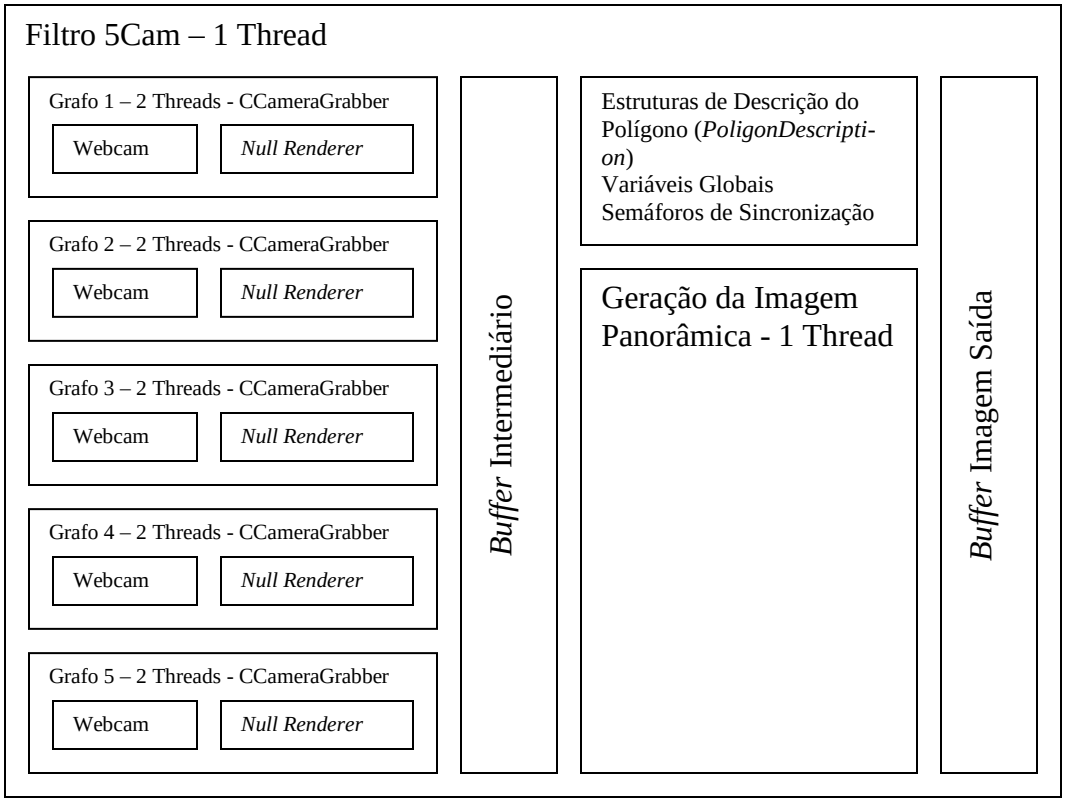


Figura 34 - Arquitetura Filtro 5Cam

O funcionamento básico do filtro é o seguinte: Cada um dos grafos tem como fonte um filtro ligado a uma das câmeras firewire com o filtro *Null Renderer* como filtro de apresentação. O filtro *Null Renderer* é responsável por disponibilizar os quadros de vídeo no *buffer* interno. De forma sincronizada a *thread* de geração da imagem panorâmica coleta as imagens do *buffer* intermediário e, utilizando as matrizes 3x3 existentes na estrutura de dados *PoligonDescription*, gera a imagem panorâmica final no *buffer* de imagem de saída. Os dados deste *buffer* serão enviados para o próximo filtro do *DirectShow* conectado depois do 5Cam.

Na inicialização do filtro do 5Cam são realizadas as seguintes operações:

- Inicialização das câmeras
- Reserva dos *buffers*

- Inicialização das estruturas de dados de mapeamento (matrizes)

O grafo *DirectShow* que contém cada câmera é encapsulado em uma classe C++ (*CCameraGrabber*). Com a utilização desta classe o processo de inicialização do grafo de filtros fica simplificado. A classe também é responsável por disponibilizar a imagem no *buffer* intermediário. O processo de sincronização entre o *thread* que fornece a imagem e o que consome será descrito posteriormente na geração da imagem panorâmica.

Cada câmera é identificada internamente por um número que no caso da 5Cam vai de 0 a 4. A classe deve ser inicializada indicando qual vai ser o identificador da câmera e associar a instância da classe com uma das câmeras físicas da 5Cam. O identificador da câmera sempre deve apontar para a mesma câmera física no 5Cam, pois no processo de calibração as câmeras não são permutáveis.

O DevicePath (Microsoft 2005) da câmera identifica a câmera física e é utilizado para associá-la com o identificador interno. O DevicePath é uma string única para cada dispositivo ligado ao *DirectShow*. No caso das câmeras utilizadas nesta aplicação o número serial delas é utilizado pelo Windows[10] no *DevicePath* como é demonstrado no exemplo abaixo na sequência em negrito.

```
\\?\1394#unibrain&fire-i_1.2#9003010061431408#{65e8773d-8f56-11d0-a3b9-
00a0c9223196}\global
```

É utilizado parte do número serial da câmera para associar o indentificador com câmera física. De forma simplificada o pseudocódigo para inicializar uma câmera é:

```
//Identifica a câmera como número 0
CCameraGrabber.Init(0); //
//Utiliza um ponteiro (pMoniker) e vincula a câmera física pelo serial
//"9003" com o ponteiro. m_deviceList é um objeto que contém uma lista
//de todos os dispositivos
```

[10] Este é um ponto para melhora da implementação. Ler diretamente o número da serial na câmera eliminando a possibilidade de um possível erro causado por uma mudança na implementação do DevicePath.

```
m_deviceList.GetMoniker(L"9003", &pMoniker);
// Associa a câmera com a instância do CCameraGrbber
CCameraGrabber.RenderCamera(pMoniker);
```

Ao final deste código o grafo, representado pelo *CCameraGraber*, já está pronto para preencher o *buffer* intermediário com amostras de vídeo. O processo de preenchimento do *buffer*, depende do estado do filtro (parado, tocando pausa) e do processo de sincronismo entre os grafos de captura de imagem e o de geração da imagem panorâmica.

O próximo passo é inicializar as estruturas de dados que contêm as matrizes de transformação.

```
typedef struct {        // a poligon point
      double u, v;            // texture space coords
      double sx, sy;          // screen space coords
      double x, y, z;         // object space coords
} POLY_VERT;

typedef struct {        // um polígono
      int n;                        // número de lados
      POLY_VERT vert[POLY_VERT_MAX];
} POLY;

struct PoligonDescription
{
      // image number where I can find the source of this poligon
      int nImg;
      // struct that holds the screen space and texture space coords
      POLY poligon;
      // tranformation matrix
      double ST[3][3];
      if this poligon has to be merged with another, this variable
      will host the / poligon number of the another poligon
      the polMerg vector;
      int polMerg;
} polDesc[POLY_MAX], polMerg[POLY_MAX];
```

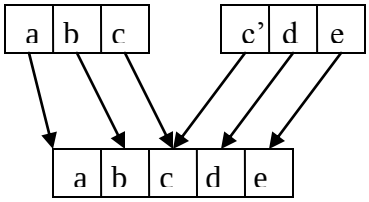
Antes de descrever a estrutura *PoligonDescription* é necessário descrever a estrutura POLY. O objetivo desta estrutura é armazenar os dados de dois polígonos que no nosso caso é o quadrilátero na imagem origem e o quadrado no destino. Os vértices destes polígonos são armazenados na estrutura POLY_VERT. No caso, armazenamos o quadrilátero da imagem original oriundo da câmera e o quadrado na imagem panorâmica correspon-

dente. O quadrilátero da imagem original tem os seus vértices armazenados nos valores u,v com o vértice correspondente da imagem panorâmica armazenado nos valores sx,sy. Cada elemento *PoligonDescription* representa a transformação de um quadrilátero no destino e seu correspondente na origem como pode se observar na Tabela 4.

PoligonDescription	
Campo	Descrição
int nImg	Número da câmera com a imagem de origem
POLY poligon	Armazena o quadrilátero de origem e o destino
Double ST[3][3]	Armazena a matriz de transformação
int polMerg	Se diferente de -1 indica que neste poligono destino existe a combinação de dois polígonos na origem como demonstrado na Figura 18.

Tabela 4 - Campos da Estrutura *PoligonDescription*

Existem dois vetores de *PoligonDescription*. O vetor polDesc armazena um elemento por quadrilátero na imagem panorâmica final. Caso este quadrilátero tenha como origem mais de um polígono, o campo *polMerg* do elemento *PoligonDescription* do quadrilátero destino estará diferente de -1. O valor do polDesc.*PoligonDescription.polMerg* será um índice para buscar no vetor *polMerg* o elemento *PoligonDescription* que representa a segunda origem que será combinada com a primeira.



polDesc = {a,b,c,d,e}
polMerg = {c'}

Figura 35 - Relação imagens, polDesc e polMerg

Para exemplificar é utilizada a Figura 35. Nesta figura as imagens de origem são representadas por Img1 e Img2, cada uma com somente uma linha e três quadriláteros. As duas imagens são combinadas na imagem panorâmica. O vetor *polDesc* tem um elemento por cada quadrilátero da imagem panorâmica final representado pelos elementos (a,b,c,d,e). O quadrilátero representado por c tem como origem dois quadriláteros nas imagens origem. Neste caso o valor do *polMerg* do quadrilátero representado pelo elemento c, será o índice que corresponde ao quadrilátero c´ do vetor *polMerg*.

Para preencher estes dois vetores usamos o arquivo de origem do 5CamCap. A Figura 36 demonstra a relação entre o arquivo e o conjunto de câmeras. Cada linha do arquivo representa um quadrilátero de origem de uma câmera. Os quadriláteros são varridos primeiramente da esquerda para a direita, de cima para baixo conforme a numeração da figura. O primeiro campo representa o número do polígono. O segundo o número da câmera de origem. Pode-se perceber neste exemplo que existem polígonos oriundos das cinco câmeras. Os valores subsequentes de cada linha representa os vértices de cada quadrilátero de origem.

```
00 - 00 - x1=0087 y1=0014 x2=0025 y2=0017 x3=0030 y3=0064 x4=0088 y4=0064
01 - 00 - x1=0152 y1=0012 x2=0087 y2=0014 x3=0088 y3=0064 x4=0150 y4=0064
02 - 00 - x1=0219 y1=0010 x2=0152 y2=0012 x3=0150 y3=0064 x4=0208 y4=0065
03 - 00 - x1=0219 y1=0010 x2=0208 y2=0065 x3=0272 y3=0066 x4=0273 y4=0014
04 - 01 - x1=0034 y1=0011 x2=0033 y2=0063 x3=0102 y3=0060 x4=0106 y4=0016
05 - 01 - x1=0172 y1=0016 x2=0106 y2=0016 x3=0102 y3=0060 x4=0176 y4=0065
06 - 01 - x1=0172 y1=0016 x2=0176 y2=0065 x3=0237 y3=0063 x4=0237 y4=0023
07 - 02 - x1=0119 y1=0018 x2=0044 y2=0018 x3=0042 y3=0063 x4=0118 y4=0063
08 - 02 - x1=0189 y1=0016 x2=0119 y2=0018 x3=0118 y3=0063 x4=0184 y4=0064
09 - 02 - x1=0231 y1=0015 x2=0189 y2=0016 x3=0184 y3=0064 x4=0236 y4=0066
10 - 02 - x1=0280 y1=0010 x2=0231 y2=0015 x3=0236 y3=0066 x4=0279 y4=0056
11 - 03 - x1=0098 y1=0019 x2=0028 y2=0025 x3=0027 y3=0071 x4=0098 y4=0072
12 - 03 - x1=0098 y1=0019 x2=0098 y2=0072 x3=0170 y3=0062 x4=0169 y4=0025
13 - 03 - x1=0234 y1=0021 x2=0169 y2=0025 x3=0170 y3=0062 x4=0236 y4=0067
14 - 03 - x1=0288 y1=0017 x2=0234 y2=0021 x3=0236 y3=0067 x4=0291 y4=0075
15 - 04 - x1=0027 y1=0022 x2=0027 y2=0064 x3=0081 y3=0066 x4=0077 y4=0024
16 - 04 - x1=0077 y1=0024 x2=0081 y2=0066 x3=0139 y3=0063 x4=0142 y4=0026
17 - 04 - x1=0202 y1=0025 x2=0142 y2=0026 x3=0139 y3=0063 x4=0202 y4=0063
18 - 04 - x1=0202 y1=0025 x2=0202 y2=0063 x3=0267 y3=0060 x4=0266 y4=0027
19 - 00 - x1=0088 y1=0064 x2=0030 y2=0064 x3=0026 y3=0112 x4=0085 y4=0111
20 - 00 - x1=0150 y1=0064 x2=0088 y2=0064 x3=0085 y3=0111 x4=0148 y4=0114
```

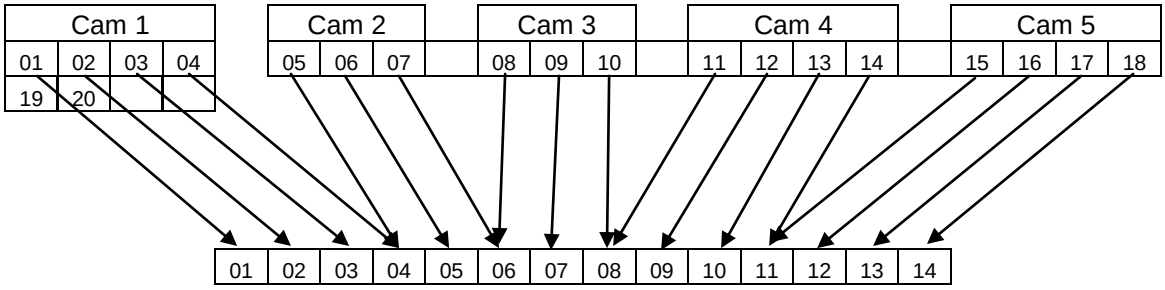


Figura 36 - Relação entre o arquivo de descrição e as imagens

Existem dois casos especiais que devem ser observados na leitura do arquivo. O primeiro é na transição entre duas câmeras. Neste caso que é representado pelos polígonos (4,5), (7,8) (10,11) e (14,15) o algoritmo percebe que o número da câmera muda e o valor da câmera subsequente é maior que o atual, indicando que os polígonos de origem devem ser combinados no mesmo quadrilátero de destino. O outro caso é na mudança de linha representado neste caso pelo conjunto (18,19). Neste caso o valor da câmera subsequente é menor que o valor atual indicando uma mudança de linha.

Ao final deste processo além de termos os vetores *polMer* e *polDesc* preenchidps, as variáveis *nColPolMax* e *nLinPolMax* também estão preenchidas com o número de colunas e linhas de quadrados que a imagem panorâmica possui.

Uma etapa adicional é o cálculo da matriz de transformação 3x3 que é armazenada em *PoligonDescripton.ST* a partir dos valores do *PoligonDescripton*.POLY. O processo de cálculo original desta matriz foi retirado de (Heckbert 1989).

Os valores dos vértices estão na estrutura de dados POLY. Para facilitar a explicar a explicação, neste trecho, a estrutura de dados será mapeada em uma notação matemática.. Os vértices da imagem original que consideramos textura estão armazenados nas variáveis vert[k].u e vert[k].v, sendo k = {0,1,2,3}. E os vértices da imagem panorâmica destino estão nas variáveis vert[k].sx e vert[k].sy, sendo k = {0,1,2,3}. O objetivo é achar o mapeamento entre (vert[k].u, vert[k].v) e (vert[k].sx, vert[k].sy). Para o desenvolvimento vamos considerar vert[k].u = u_k, vert[k].v = v_k, vert[k].sx = sx_k e vert[k].sy = sy_k.

Conforme já foi discutido anteriormente (3.2 Calibração da Câmera), considerando i=0 na equação a seguir temos:

$$(sx_k, sy_k, w) = (u_k, v_k, q)\begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix} \qquad \text{(Eq 3)}$$

Podemos montar a matriz M resolvendo o sistema de equações com 8 equações e 8 variáveis composto pelos componentes:

$$sx_k = \frac{au_k + bv_k + c}{gu_k + hv_k + 1} \Rightarrow u_k a + v_k b + c - u_k sx_k g - v_k sx_k h = sx_k \qquad \text{(Eq 4)}$$

$$sy_k = \frac{du_k + ev_k + f}{gu_k + hv_k + 1} \Rightarrow u_k d + v_k e + f - u_k sy_k g - v_k sy_k h = sy_k \qquad \text{(eq 5)}$$

para k = 0,1,2,3.

Isso pode ser escrito como um sistema 8x8:

$$\begin{pmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0 sx_o & -v_0 sx_o \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1 sx_1 & -v_1 sx_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2 sx_2 & -v_2 sx_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3 sx_3 & -v_3 sx_3 \\ 0 & 0 & 0 & u_0 & v_0 & 1 & -u_0 sy_o & -v_0 sy_o \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1 sy_1 & -v_1 sy_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2 sy_2 & -v_2 sy_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3 sy_3 & -v_3 sy_3 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{pmatrix} = \begin{pmatrix} sx_0 \\ sx_1 \\ sx_2 \\ sx_3 \\ sy_0 \\ sy_1 \\ sy_2 \\ sy_3 \end{pmatrix} \qquad \text{(eq 6)}$$

Este sistema linear pode ser resolvido usando eliminação de Gauss. A equação acima resolve um caso genérico de mapeamento de quadriláteros para quadriláteros. O resultado deste sistema é a matriz 3x3, utilizada para realizar o mapeamento do quadrilátero da imagem original para o quadrado da imagem panorâmica. Existem formas mais eficientes de se realizar este cálculo em casos especiais. Serão analisados os casos de quadrados para quadriláteros, quadriláteros para quadrados e quadriláteros para quadriláteros usando a combinação dos dois casos anteriores. Na Figura 37 estes casos são demonstrados graficamente.

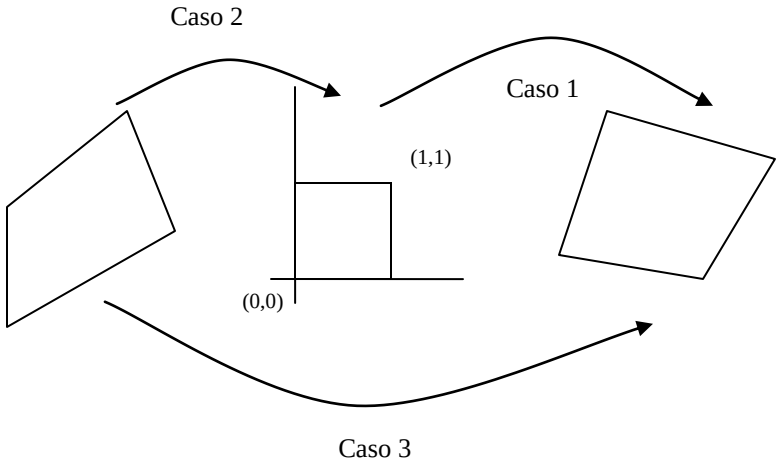


Figura 37 - Mapeamento quadrilátero - quadrilátero a partir de casos simples de mapeamento (Heckbert 1989)

Caso 1 - quadrado para quadrilátero: O sistema é resolvido simbolicamente para o caso que o quadrado é unitário. Se os vértices correspondentes forem:

sx	**sy**	**u**	**v**
sx_0	sy_0	0	0
sx_1	sy_1	1	0
sx_2	sy_2	1	1
sx_3	sy_3	0	1

As equaçoes ficam reduzidas a

$sx_0 = c$
$sx_1 = a + c - gsx_1$
$sx_2 = a + b + c – gsx_2 + hsx_2$
$sx_3 = b + c – hsx_3$
$sy_0 = f$
$sy_1 = d + f – gsy_1$
$sy_2 = d + e + f – gsy_2 + hsy_2$
$sy_3 = e + f – hsy_3$

Se definirmos:

$$\Delta sx_1 = sx_1 - sx_2 \qquad \Delta sx_2 = sx_3 - sx_2 \qquad \sum x = sx_0 - sx_1 + sx_2 - sx_3$$

$$\Delta sy_1 = sy_1 - sy_2 \qquad \Delta sy_2 = sy_3 - sy_2 \qquad \sum y = sy_0 - sy_1 + sy_2 - sy_3$$

a solução fica definida em dois subcasos:

(a) $\sum x = 0$ e $\sum y = 0$. Isso implica que o polígono sx sy é um palelograma, então o mapeamento é afim, e a=sx_1-sx_0, b=sx_2-sx_1, c=sx_0, d=y_1-y_0, e=y_2-y_1, f=y_0, g=0, h=0.

(b) $\sum x \neq 0$ ou $\sum y \neq 0$. Neste caso temos uma perspectiva e utilizamos o seguinte conjunto de equações:

$$g = \begin{pmatrix} \sum x & \Delta sx_2 \\ \sum y & \Delta sy_2 \end{pmatrix} / \begin{pmatrix} \Delta sx_1 & \Delta sx_2 \\ \Delta sy_1 & \Delta sy_2 \end{pmatrix}$$

$$h = \begin{pmatrix} \Delta sx_1 & \sum x \\ \Delta sy_1 & \sum y \end{pmatrix} / \begin{pmatrix} \Delta sx_1 & \Delta sx_2 \\ \Delta sy_1 & \Delta sy_2 \end{pmatrix}$$

$$a = sx_1 - sx_0 + gsx_1$$

$$b = sx_3 - sx_0 + hsx_3$$

$$c = sx_0$$

$$d = sy_1 - sy_0 + gsy_1$$

$$e = sy_3 - sy_0 + hsy_3$$

$$f = sy_0$$

Este método é muito mais rápido que a solução do sistema de equações 8x8 ((Heckbert 1989).

Caso 2 - quadrilátero para quadrado: A forma mais eficiente de realizar este mapeamento não é simbólica, como o caso anterior, mas numérica. Utilizando as fórmulas do caso anterior e se obtém a matriz adjunta da matriz resultado. A matriz adjunta é a inversa no caso de coordenadas homogêneas.

Caso 3 - quadrilátero para quadrilátero: Como conseguimos mapear de um quadrilátero para um quadrado e de um quadrado para um quadrilátero podemos combinar as duas operações para mapear de um quadrilátero para outro quadrilátero conforme demonstrado na Figura 37

Ao final deste processo já temos as câmeras inicializadas, as estruturas de mapeamento preenchidas e os *buffer*s reservados estando tudo pronto para gerar o vídeo panorâmico. A geração ocorre chamando a função Fill*Buffer*. Cada vez que o Windows precisa de uma nova amostra de vídeo esta função é chamada. Na primeira vez que a função é chamada, os *buffer*s intermediários já estão preenchidos e prontos para gerar a imagem panorâmica. Cada quadrilátero é individualmente processado e a imagem contida transformada, usando a matriz 3x3 adequada, em um quadrado, que irá compor a panorâmica. A imagem panorâmica é gerada em um *buffer* disponibilizado pelo *DirectShow* e encaminhada ao próximo filtro subsequente ao 5Cam. Ao final deste procedimento o Fill*Buffer* libera os *threads* de captura de imagem para preencher novamente os *buffer*s intermediários. **Finalmente uma imagem semelhante à Figura 38 aparece no monitor do usuário.**

Figura 38 - Imagem Panorâmica

4.3.3 Otimização

A implementação inicial do projeto não teve como objetivo o melhor desempenho. Depois que o Sistema 5Cam tornou-se funcional, iniciou-se o processo de otimização, de

seu código, visando a obtenção de uma taxa ‘mínima de 15 quadros por segundo para o vídeo. Este tópico está baseado em (Geber 2002) e (Intel 2001). O processo de otimização de código inicia-se com a escolha de um *benchmark*, que é uma medida objetiva sobre o desempenho do código para avaliar o resultado do processo de otimização.

Depois de definido o *benchmark*, se inicia o processo de otimização, com a análise dos pontos mais críticos da aplicação, os que estão consumindo mais tempo de execução, memória ou qualquer outro critério que esteja sendo otimizado. A partir da identificação destes pontos, é realizada a análise que identifica a razão do ponto crítico, que podem ser os mais variados como excesso de loops, acessos de memória, execução de ponto flutuante, etc.

Com o trecho identificado e a razão diagnosticada, modifica-se a aplicação e executa-se novamente o *benchmark*. Nem sempre as modificações irão otimizar o código e neste caso devem ser desfeitas ou re-analisadas. A Figura 39 demonstra o processo.

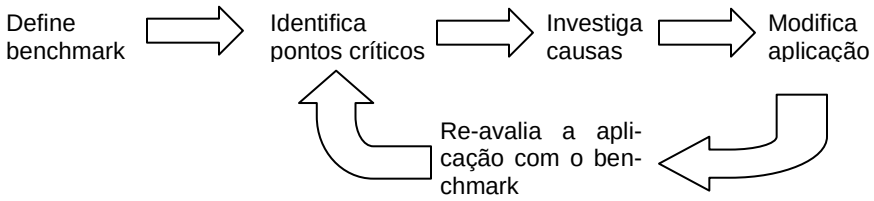


Figura 39 – Processo de otimização (Geber 2002)

Um passo importante é a escolha do *benchmark*. Ele será o índice que iremos utilizar para otimizar a aplicação, a sua escolha representa o que esperamos da aplicação. O *benchmark* pode ser desde uma simples medida de tempo até um índice agregado que inclua fatores incompatíveis, representando um balanceamento dos fatores a serem otimizados.

Os seguintes atributos são importantes na escolha de um *benchmark* (Geber 2002):

- **Repetível**: Um *benchmark* que forneça resultados diferentes para um aplicativo que esteja rodando sobre as mesmas condições, não é útil. O *benchmark* deve ser repetível trazendo sempre o mesmo resultado, quando executado sobre as mesmas condições. Um ponto a ser considerado neste processo é a existência de fatores transientes na execução de um *benchmark* como acessos iniciais a disco, preenchimento de cachês de memória, etc, que podem comprometer a qualidade do *benchmark*. O *benchmark* deve prever estes fatores e ajustar a sua construção ou a sua execução de forma que se elimine ou reduza o impacto destes fatores.
- **Representativo**: O *benchmark* deve ser representativo da aplicação que ele está avaliando. Testar casos de erro, secundários ou atípicos não representam a operação diária do aplicativo. O *benchmark* deve avaliar o caso comum no qual usuário vai utilizar o aplicativo e nas condições mais próximas do real que for possível simular.
- **Fácil de executar**: O *benchmark* deve ser fácil de rodar, coletar os dados e proceder com a análise. A Figura 39 mostra que o processo de otimização é cíclico. Ao se tornar o *benchmark* fácil de rodar, podemos passar pelo ciclo de forma mais rápida e apurar os resultados das modificações freqüentemente.
- **Verificável:** O *benchmark* deve ser verificável. Deve ser possível verificar se o resultado que o *benchmark* produz está de acordo com a realidade.

No caso da 5Cam, o *benchmark* utilizado é o tempo em milisegundos necessário para processar um frame. O objetivo é que o vídeo seja exibido com no mínimo 15 quadros por segundo, o que resulta em um valor objetivo máximo de 67 milisegundos por quadro.

Para auxiliar no processo de otimização foi utilizado um *software profiler*, que é um aplicativo que caracteriza a execução do software. Com ele é possível identificar os pontos de gargalo em sua execução. Quanto mais conhecimento dos detalhes de hardware da arquitetura utilizada, mais eficientemente o *software profiler* pode atuar. Existem duas formas de atuação dos *software profilers*:

- **Amostragem –** Neste caso o *software profiler* periodicamente interrompe a execução do sistema e coleta as informações de desempenho necessárias. As informações pode ser genérica independente de arquitetura, como posição do ponteiro de instrução, número de instruções executadas, identificador de processo ou de *thread* ou específica como número de instruções executadas por *pipeline* do processador, índices internos do cachê, etc.
- **Instrumentação** – Neste caso o software instrumenta o programa, inserindo código para auxiliar no processo de caracterização. Com este tipo de instrumentação é possível fazer a árvore de chamada de funções que indica o caminho crítico na execução da aplicação, número de chamadas de cada função, entre outros.

O *VTune Desempenho Analyser* da Intel foi utilizado como *sotfware profiler* na 5Cam.

O processo de otimização é descrito a seguir desde do seu ponto inicial até o até o seu ponto final. Vamos nos referir o ponto inicial como linha base do processo de otimização. O computador utilizado neste processo foi um Pentium HT 3.0 GHZ com 512 Mb de memória RAM. Em cada ponto do processo são descritos o diagnóstico e a modificação. Cada ponto também acompanha a seguinte tabela resumo:

- **Revisão de código**: Número da revisão do código no repositório *subversion*[11] da versão de código com o resultado obtido.

[11] O código fonte do projeto sempre esteve armazenado em um sistema de controle de versão. Ao final do projeto foi utilizado o Subversion. (http://www.subversion.com)

- **Valor Original**: Utiliza o *benchmark* de milisegundos por quadro com os limites inferiores e superiores de um intervalo de confiança de 95%. O formato utilizado é xxx/yyy/zzz ms, aonde xxx é o limite inferior do intervalo de confiança, yyy é a média e zzz é o intervalo superior. (Jain 1991)
- **Resultado obtido**: Resultado obtido no mesmo formato utilizado no diagnóstico.
- **Melhoria**: Valor da melhoria percentualmente

4.3.3.1 Linha Base

Linha de Base do código completo sem nenhuma otimização do compilador ou linkeditor.

Revisão do Código	10
Valor Original	266,05/267,43/268,80 ms
Resultado obtido	266,05/267,43/268,80 ms
Melhoria	0%

Tabela 5 - Linha Base

4.3.3.2 Otimização de compilador e linkeditor

Compilação do código utilizando o modo do compilador e linkeditor configurados para máximo desempenho.

Revisão do Código	11
Valor Original	266,05/267,43/268,80 ms
Resultado obtido	166,68/167,16/167,67 ms.
Melhoria	37,49 %

Tabela 6 - Otimização de compilador e linkeditor

4.3.3.3 Paralelização de *Threads*

O 5Cam tem a *thread* que gera a imagem panorâmica e as *threads* de captura de imagem conforme mostrado na Figura 34. A arquitetura inicial não permitia que estas *threads* fossem executadas simultaneamente na máquina, pois eles compartilhavam o mesmo *buffer* de processamento da imagem. As *threads* de captura da imagem da câmera escreviam no *buffer*, ao mesmo tempo em que a de geração da imagem panorâmica efetuava a leitura do *buffer* para processamento. A análise mostrou que o processo de captura da

imagem da câmera consome em média 18,3% (30,59 ms) do tempo total de construção de um quadro e a *thread* de geração consome em média 81,7% (136,56 ms) conforme a Figura 40

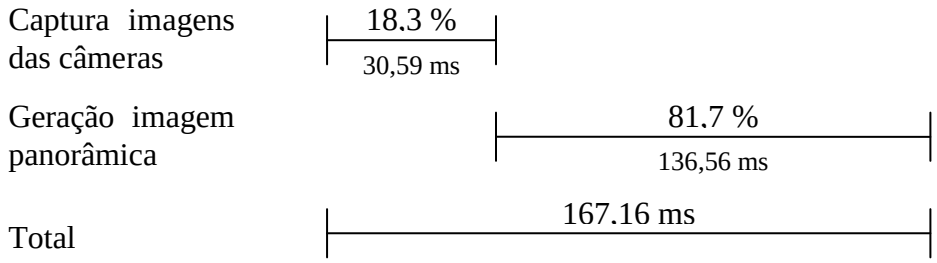


Figura 40 - Relação dos *threads* no 5Cam sem otimização.

Para paralelizar o processo foi adicionado um *buffer* à arquitetura da Figura 34. Enquanto as *threads* de captura da imagem da câmera estão preenchendo um *buffer*, a *thread* de geração da imagem panorâmica está utilizando o outro. O resultado desta mudança pode ser visto na Figura 41.

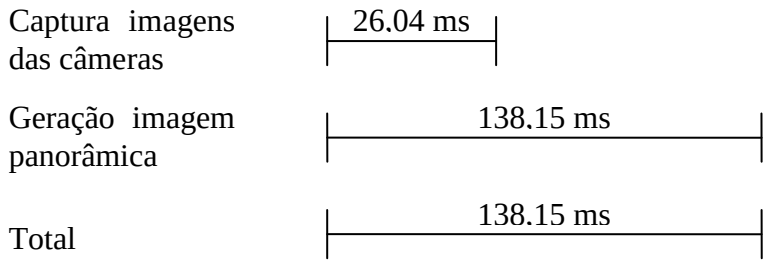


Figura 41 Relação dos *threads* no 5Cam com otimização.

A Tabela 7 resume o aumento de desempenho obtida.

Revisão do Código	14
Valor Original	166,68/167,16/167,67 ms.
Resultado obtido	137,85/138,15/138,45 ms.
Melhoria	17,36 %

Tabela 7 - Paralelização de *Threads*

4.3.3.4 Mudança de Double→Float

Parte do código da 5Cam utiliza ponto flutuante. Durante a criação do código todos os elementos de ponto flutuante tinha precisão dupla com o uso do double como tipo de dado, com a conversão dos elementos para o tipo de dados float, foi possível obter o resultado abaixo:

Revisão do Código	15
Valor Original	137,85/138,15/138,45 ms.
Resultado obtido	133,70/134,53/135,36 ms.
Melhoria	2,62 %

Tabela 8 - Mudança de Double->Float

4.3.3.5 Otimização usando VTune / Utilização de SIMD

Depois de executadas as otimizações globais, foi iniciado o processo de otimização com o auxílio do VTune. O primeiro processo de análise utilizou a função de amostragem do Vtune coletando os seguintes parâmetros:

- **Instructions Retired**: Indica o número de instruções executadas completamente. Não incluem instruções que foram retiradas para execução e foram descartadas por erros de predição de caminhos de execução no código.
- **Clockticks**: É a únidade básica de tempo reconhecida pela arquitetura Intel.
- **Clockticks per instruction retired (CPI)**: Este indicador é considerado o primeiro a ser avaliado em uma análise de desempenho. Um número alto (>4) indica que o processador não está executando com sua máxima capacidade, seja por problemas de predição de pulos, falhas no acesso de cache, entre outros.

O processo de amostragem mostrou que a função *WriteBuffer* era o ponto de gargalo. A primeira otimização foi à conversão do trecho de cálculo da interpolação bilinear para

instruções SIMD (*Single Instruction Multiple Data*). Neste tipo de instrução é possível realizar a mesma operação em um conjunto múltiplo de dados (Intel 2001).

A Tabela 9 mostra o resultado da conversão, constantando uma queda de desempenho.

Revisão do Código	16
Valor Original	133,70/134,53/135,36 ms.
Resultado obtido	141,60/142,25/142,90 ms.
Melhoria	-6,02 %

Tabela 9 - Mudança da rotina de interpolação bilinear para SIMD.

Analisando o código, depois da conversão para SIMD, com o auxílio do Vtune foram detectados 4 pontos de gargalo onde o processador tinha que esperar pela escrita completa de um dado na memória antes de poder dar prosseguimento à execução. Este fenômeno é chamado *Store Forward Blocked* (Intel 2005). O código foi reescrito para retirar esta dependência chegando ao resultado da Tabela 10.

Revisão do Código	17
Valor Original	141,60/142,25/142,90 ms.
Resultado obtido	137,85/138,15/138,45 ms.
Melhoria	17,36 %

Tabela 10 - Retirada dos *Store Forward Blocked*.

4.3.3.6 Conversão YUV→RGB, inserção de *lookup table*

Uma nova análise indicou a função que realiza a conversão do sistema de cores YUV[12] para RGB (Gonzales and Woods 2000) como outro grande ponto de gargalo. As câmeras entregam as imagens codificadas em um formato YUV e o Windows necessita que o *buffer* seja preenchido com o formato RGB, necessitando desta função de conversão. Neste caso foi inserida uma *lookup table* eliminando uma série de operações de multiplicação no processo de conversão. O resultado pode ser observado na Tabela 11

[12] No (Gonzáles and Woods 2000) o formato YUV é chamado YIQ.

Revisão do Código	19
Valor Original	137,85/138,15/138,45 ms.
Resultado obtido	131,27/131,61/131,96 ms
Melhoria	4,73 %

Tabela 11 - Conversão YUV->RGB, inserção de lookup table

4.3.3.7 Reescrita função mx3d_transform para SIMD

O próximo gargalo analisado foi à função *mx3d_transform*. Esta função realiza a conversão do ponto (*x*,*y*) no quadrado da imagem panorâmica destino para o correspondente no quadrilátero de origem. Nesta etapa além de converter a função para SIMD a chamada da função foi eliminada incorporando a funcionalidade na função *WriteBuffer*. O resultado está na Tabela 12

Revisão do Código	23
Valor Original	131,27/131,61/131,96 ms
Resultado obtido	128,66/129,00/129,33 ms
Melhoria	1,98 %

Tabela 12 - Reescrita função mx3d_transform para SIMD

4.3.3.8 Lookup substituindo o recálculo do *mx3d_transform*

Analisando os retornos decrescentes e a falta de novos hotspots detectados pelo *Vtune*, foi tomada a decisão de analisar o problema do ponto de vista algoritmo modificando a estrutura do código. Neste caso foi eliminado do loop de criação da imagem panorâmica todo o cálculo das transformadas dos quadrados em quadriláteros. Este cálculo passou a ser executado somente uma vez na inicialização da 5Cam e a função de geração da imagem panorâmica utiliza os valores pré-calculados. Com esta abordagem os resultados obtidos foram mais satisfatórios conforme a Tabela 13.

Revisão do Código	28
Valor Original	128,66/129,00/129,33 ms
Resultado obtido	93,62/93,81/94,00 ms
Melhoria	27,27 %

Tabela 13 - Lookup substituindo o recálculo do mx3d_transform

4.3.3.9 Remoção de chamadas GDI (*Graphics Device Interface*)

Neste ponto o Vtune indicou que o tempo gasto pelo aplicativo nas chamadas GDI do Windows já era maior que o tempo utilizado pelo próprio aplicativo no seu processamento. A GDI (*Graphics Device Interface*) é a interface de programação para a parte gráfica do Windows. Ela é extremamente genérica suportando qualquer tipo de dispositivo gráfico que possa ser utilizado com o Windows e abstraindo completamente os detalhes destes dispositivos para o programador. Esta generalidade a torna lenta, pois as chamadas devem checar todas as possíveis situações. Neste caso foi construída uma função para escrever os valores de cor na imagem, que acessa diretamente as estruturas de memória no Windows. O resultado final pode ser observado na Tabela 14

Revisão do Código	32
Valor Original	93,62/93,81/94,00 ms
Resultado obtido	33,25/33,31/33,37 ms
Melhoria	64,49 %

Tabela 14 - Remoção de chamadas GDI (Graphics Device Interface)

Com esta etapa chega ao fim do processo de otimização chegando ao objetivo de um quadro processado em menos de 64 milisegundos. O Figura 42 resume o ganho de desempenho de cada etapa que ao final conseguiu uma melhora global de 87,54%.

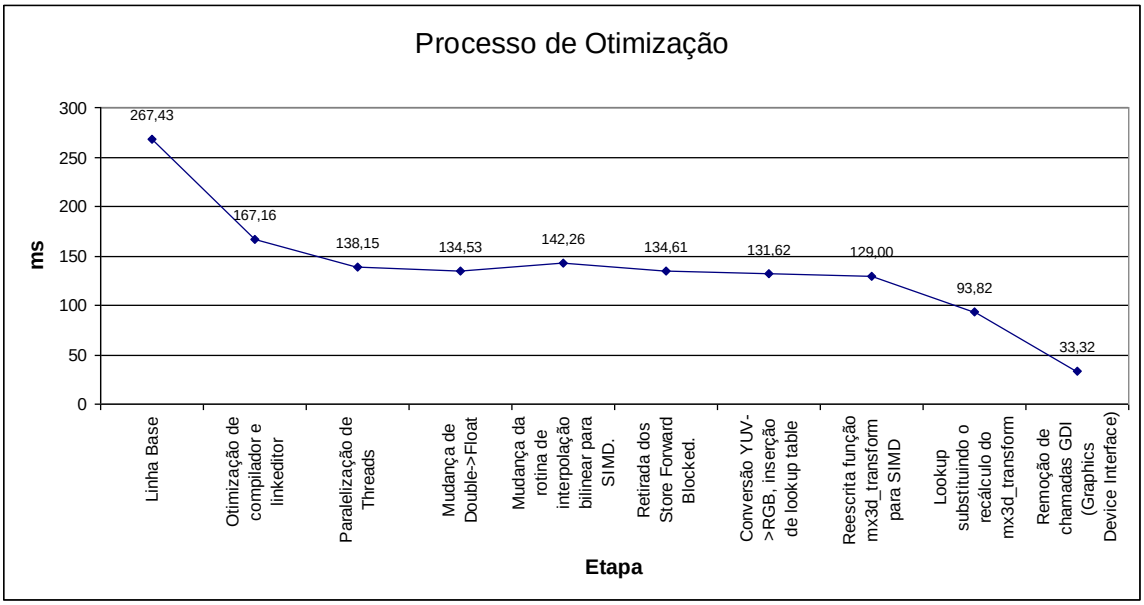


Figura 42 - Resumo Processo Otimização

A principal conclusão do gráfico acima é que os grandes ganhos de desempenho foram obtidos a partir de mudanças estruturais do código do que mudanças pontuais. A otimização utilizando o SIMD, apesar de ter um ganho de performance não foi o ponto crucial na otimização.

Capítulo 5 CONCLUSÃO

Nos capítulos anteriores, realizamos uma revisão sobre o tema câmaras omnidirecionais, apresentamos o sistema 5Cam, sua arquitetura, soluções para a implementação e a otimização do seu código.

O objetivo principal desse trabalho foi construir uma câmera panorâmica para captura de reuniões, com custo reduzido e taxa mínima de 15 quadros por segundo para o vídeo capturado. No decorrer do texto diversos tipos de câmeras omnidirecionais foram apresentadas, estudadas e ao final foi adotado o modelo que utiliza múltiplas câmeras para a construção de um protótipo. Este modelo foi o que mais se adequou às necessidades de captura de uma reunião com baixo custo e de fácil fabricação, requisito inicial desse trabalho. No âmbito nacional não foi encontrada solução similar ao protótipo de câmera panorâmica do tipo múltiplas câmeras apresentado neste trabaho. Uma solução para geração de imagem panorâmica com utilização de uma câmera um espelho foi apresentado por (Junior 2002).

A principal contribuição deste trabalho é o protótipo da câmera e do seu sistema de calibração, assim como os softwares correspondentes 5Cam e 5CamCap. Os diversos problemas relativos à distorção inserida pelas lentes, a não existência de um único centro de projeção, entre outros foram solucionados resultando uma boa qualidade das imagens geradas. Uma arquitetura foi desenvolvida com a utilização de padrão de mercado que foi o *DirectShow* da Microsoft. Algoritmos para fusão das imagens foram estudados e implementados, tendo como base os trabalhos de (Heckbert 1989). Finalmente o Sistema 5Cam teve o seu código otimizado aumentando a sua desempenho total em 87% da implementação original. Esse fato foi fundamental para permitir a geração de um *stream* de vídeo com taxa aproximada de 30 quadros por segundo, o dobro da meta inicial.

5.1 Perspectivas

As melhorias da 5Cam podem seguir alguns caminhos aqui listados:

Melhoria no hardware: Incluir um array de microfones na base da câmera para gravar paralelamente o áudio como em (Cutler, Rui et al. 2002), melhorar a construção da base para aproximar os seus centros de projeção diminuindo os efeitos de paralaxe ou uma melhor forma de fixação das câmeras para impedir pequenos desvios na sua manipulação.

Melhoria no software 5Cam: melhorar o processo de correção do fator Gamma (Gonzales 2000), automatizar o processo de calibração usando técnicas como em (Szeliski 1996), implementar correção de brilho ou cor no hardware do equipamento como em (Nanda and Cutler 2001) seriam os principais pontos a serem melhorados.

Utilização do vídeo capturado: Em (Neto 2005) foi criado um software para anotação de vídeo. Poderia se construir um software específico para anotação de um vídeo de reunião com a semântica adequada como em (Cutler, Rui et al. 2002), (Lee, Erol et al. 2002)

5.2 Lições Aprendidas

A experiência de construção de um protótipo que envolva um hardware e que interaja com o mundo físico é sempre interessante. Caso você contrarie as leis da ótica, da física ou qualquer outra, o protótipo simplesmente não funciona. Quando se está programando um sistema que não tem estas restrições do mundo real, uma má implementação ou análise se pode levar a um processo mal feito e ineficiente ou uma interface com o usuário não adequada. No caso deste trabalho, ao não se entender um determinado conceito o protótipo simplesmente não funciona ou funciona de forma inadequada..

Junto com todos os percalços existentes na construção de um protótipo como mal contatos, detalhes de hardware entre outros, o mais interessante neste processo foi conviver com estas restrições impostas pela interação com o mundo real.

REFERÊNCIAS

Anton, H. and C. Rorres (2001). Algebra linear com aplicações, Bookman.

Association., T. (2001). IIDC 1394-based Digital Camera Specification Version 1.30.

Baker, S. and S. K. Nayar (2001). Single Viewpoint Catadioptric Cameras. Panoramic Imaging: Sensors, Theory and Applications. B. a. Kang.

Chapman, D., A. Deacon, et al. (2002). "An omnidirectional imagiing system for the reverse engineer of industrial facilities." The International Archives of Photogrametry, Remote Sensing and Spatial Information Sciences 34.

Chen, S. E. (1995). {QuickTime {VR}} --- An Image-Based Approach to Virtual Environment Navigation. SIGGRAPH 95.

Cutler, R., Y. Rui, et al. (2002). Distributed meetings: a meeting capture and broadcasting system Proceedings of the tenth ACM international conference on Multimedia Juan-les-Pins, France ACM Press: 503-512

Davies, G. (2000). An Introduction to *DirectShow* Parser Filters. 2005.

DeSouza, G. N. and A. C. Kak (2002). "Vision for Mobile Robot Navigation: A Survey." IEEE Transactions on Pattern Analysis and Machine Intelligence 24(2): 237-267.

Devernay, F. and O. Faugeras (2001). "Straight Lines Have to Be Straight." Machine Vision and Applications 13: 14-24.

Feynman, R. P., R. B. Leighton, et al. (1989). The Feynman lectures on physics. Redwood City, Calif., Addison-Wesley.

Foote, J. and D. Kimber (2000). FlyCam: practical panoramic video Proceedings of the eighth ACM international conference on Multimedia Marina del Rey, California, United States ACM Press: 487-488

Gaspar, J., N. Winter, et al. (2000). "Vision Based Navigation and Environmental Representations with an Omndirectional Camera." IEEE Transactions on Robotics and Automation 16(6).

Geber, R. (2002). The Software Optimization Cookbook, IEEE Press.

Glasbey, C. A. and K. V. Mardia (1998). "A Review of Image Warping Methods." Journal of Applied Statistics 25: 155-171.

Gonzales, R. C. and R. E. Woods (2000). Processamento de Imagens Digitais, Editora Edgard Blucher.

Heckbert, P. S. (1989). Fundamentals of Texture Mapping and Image Warping, University of California at Berkeley.

Houaiss, A. and M. d. S. Vilar (2001). Dicionário Houaiss da Lingua Portuguesa, Objetiva.

Intel (2001). Desktop Performance and Optimization for Intel® Pentium® 4 Processor, Intel.

Intel (2005). IA-32 Intel® Architecture Optimization Reference Manual, Intel Press.

Jain, R. (1991). The Art of Computer Systems Performance Analysis, John Wiley & Sons.

Junior, V. G. (2002). Sistema de Visão Omnidirecional Aplicado no Controle de Robôs Móveis. Departamento de Engenharia Mecatrônica. São Paulo, Universidade de São Paulo: 85.

Lee, D.-S., B. Erol, et al. (2002). Portable Meeting Recorder. ACM Multimedia, France.

Lowe, D. (2004). Cameras and image formation. 2005.

Maita, Y., K. Hashimoto, et al. (2005). A New TV Conference System with Flexible Middleware for Omni-Directional Camera. 16 International Workshop on Database and Expert System Applications.

Majumder, A., W. B. Seales, et al. (1999). Immersive teleconferencing: a new algorithm to generate seamless panoramic video imagery. ACM Multimedia.

Microsoft (1998). COM: Component Object Model. 2005.

Microsoft (2005). *DirectShow* SDK, Microsoft. 2005.

Mittal, A. and D. Huttenloche (2000). Scene Modeling for Wide Area Surveillance and Image Synthesis. International Conference on Computer Vision and Pattern Recognition.

Murphy, J. R. (1995). Application of panospheric imaging to a teleoperated lunar rover. IEEE International Conference on Systems, Man and Cybernetics.

Nanda, H. and R. Cutler (2001). Practical Calibrations for a real-time digital omnidirectional camera. Computer Vision and Pattern Recognition.

Nayar, S. K. (1997). Omnidirectional Vision. International Symposium on Robotics Research.

Nayar, S. K. and V. Peri (1999). Folded Catadioptric Cameras. IEEE Computer Vision and Pattern Recognition Conference.

Neto, A. N. R. (2005). Coyote Annotation: Um Framework para Anotação de Vídeos Digitais. Programa de Pós Graduação em Redes de Computadores. Salvador, UNIFACS.

Oh, S. J. and E. L. Hall (1988). Guidance of a mobile robot using an omnidirectional vision navigation system. SPIE Conference on Mobile Robots.

Onoe, Y., N. Yokoya, et al. (1998). Visual Surveillance and Monitoring System Using an Omnidirectional Video Camera. 14th International Conference on Pattern Recognition.

Pesce, M. D. (2003). Programming Microsoft *DirectShow* for Digital Video and Television, Microsoft Press.

Szeliski, R. (1996). "Video Mosaics for Virtual Environments." IEEE COMPUTER GRAPHICS AND APPLICATIONS 16(2): 22-30.

Unibrain (2005). FireWire bandwidth use of cameras, Unibrain. 2005.

Winters, N. (2001). A Holistic Approach to Mobile Robot Navigation using Omnidirectional Vision. Department of Computer Science,, University of Dublin,: 157.

Woods, A., T. Docherty, et al. (1993). Image Distortions in Stereoscopic Video Systems. SPIE.

Yu, J. and L. McMillan (2004). General Linear Cameras. LECTURE NOTES IN COMPUTER SCIENCE. SPRINGER-VERLAG: 14-27.