Relatório Final de Projeto de Iniciação Científica

Título: **Projeto de Pesquisa**	Tecnologia da Informação e Comunicação no Monitoramento de Pacientes Portadores de Hipertensão e Diabetes
Título: **Trabalho de Iniciação**	Emprego de aparelho de TV na Telemedicina Domiciliar utilizando o Microsoft Kinect
Área do Conhecimento:	Engenharias
Referência da Chamada:	(X) PIBIC ( ) PIVIC
Bolsa: Agência	(X) CNPq ---- ( ) Ifes ---- ( ) Facitec ( ) Fapes ---- ( ) Voluntariado ---- ( ) CNPq-Ações Afirmativas
Orientador do Projeto:	Rodrigo Varejão Andreão
Endereços para contato:	Eletrônico: Telefônico:
Estudante/ Bolsista:	Felipe Küster de Freitas
Endereços para contato:	Eletrônico: felipekf@outlook.com Telefônico: 27 99624-6733
Campus/Coordenadoria:	Vitória / Engenharia Elétrica
Data:	29/06/2015

1. Resumo

O tratamento de pacientes com doenças crônicas tais como hipertensão e diabetes requer acompanhamento médico constante. O objetivo deste projeto é utilizar os aparelhos de televisão, um dos eletrodomésticos mais comuns, em auxílio deste paciente crônico. Utilizando-se o Microsoft Kinect é possível criar uma interface de usuário de alta interatividade. São obtidas imagens que, ao processadas, retornam dados sobre os movimentos realizados e permitem executar eventos, testes e comandos.

Ainda, neste projeto investiga-se a viabilidade da implementação de soluções tecnológicas embarcadas de baixo custo nas quais o usuário recebe um sistema de computação embarcado junto com os dispositivos de detecção de imagem e movimento em tamanho portátil.

2. Palavras-chave

Microsoft Kinect ®, *Processing*, Linux, doenças crônicas, *skeleton tracking*. TLD, Rockchip, Raspbery Pi

3. Introdução e Justificativa

Cuidados com pacientes crônicos podem ser significantemente melhorados ao se estabelecer um método de cuidado residencial. Pacientes com diabetes devem coletar e monitorar dados como o nível de glicose sanguíneo, o número de doses de insulina aplicadas, peso, rotinas de exercícios e de pressão sanguínea [1].

O número de pacientes com doenças crônicas está aumentando em todo o mundo em função do envelhecimento da população. Muitos deles também se encontram em condição de recuperação pós-hospitalar com doenças crônicas condicionadas pela diabetes, como hipertensão. Para este grupo, é essencial a prática de exercícios físicos. Com este fim, desenvolveu-se neste trabalho uma forma de exercitação assistida (*Assisted Exercising* [2]) com os dados do usuário em tempo real retornados pelo Kinect.

A introdução destes hábitos na rotina do paciente requer um cenário com o qual este esteja familiarizado, pois a coleta de dados requer uma interface homem-máquina. Utilizar-se do aparelho de TV para este objetivo é uma alternativa para realizar esta interface. Com o auxílio do Microsoft Kinect e do uma CPU para o processamento dos dados é possível gerar e tratar eventos controlados por gestos e expressões do usuário através da análise de imagens em tempo real.

Em junho de 2011, a Microsoft publicou o primeiro kit de desenvolvimento, conhecido como Kinect for Windows SDK Beta 1. A resposta do mercado foi positiva. A programação utilizando o Kinect deixou de ter como objetivo apenas a produção de jogos.

O escopo do trabalho aqui apresentado envolve o estudo das bibliotecas de processamento de dados e imagens do Microsoft Kinect, o uso destas ferramentas na programação de dois algoritmos: i) de detecção e simulação de um cursor virtual para facilitar a interface com pacientes acamados; ii) de monitoramento de juntas do esqueleto humano (*skeleton tracking*, Figura 1), e a sua aplicabilidade em sistemas embarcados domiciliares.

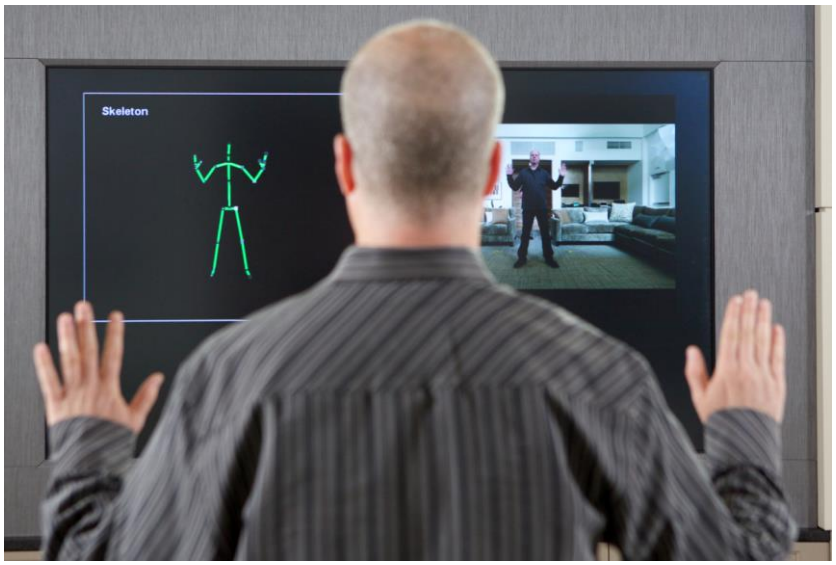


Figura 1: Exemplo de Skeleton Tracking à esquerda em comparação com a imagem em RGB do usuário, à direta.

4. Objetivos

O objetivo principal deste projeto é o estudo, a pesquisa e o desenvolvimento de um sistema de telemonitoramento do estado de saúde, hábitos de vida e de parâmetros vitais e biológicos de pacientes com diagnóstico de condições crônicas de adoecimento, particularmente hipertensão arterial sistêmica e diabetes mellitus, acompanhados por equipes de saúde multiprofissionais que atuam na Atenção Primária a Saúde.

Nesse escopo, destaca-se o desenvolvimento de uma solução para equipar um aparelho de TV com recursos de interatividade (execução de aplicativos e acesso à rede sem fio) e o reconhecimento de gestos com o objetivo de auxiliar o paciente no autocuidado.

5. Material e métodos

Por ser uma tecnologia relativamente nova, os primeiros passos do projeto incluem o estudo e o aperfeiçoamento na utilização das ferramentas e materiais relacionados ao Kinect. Uma vez em posse dos requisitos, escolheu-se a linguagem de programação mais apropriada para a utilização do equipamento no Linux, que é o sistema operacional base para a maior parte dos dispositivos móveis.

Os tópicos a seguir descrevem objetivamente todos os passos do desenvolvimento e dos experimentos executados.

5.1. Busca de bibliotecas e *drivers* de acesso ao Kinect nas plataformas Linux, Windows e Android

Há três bibliotecas de desenvolvimento do tipo *Software Development Kit* (SDK) que possibilitam o uso do Microsoft Kinect :

i) *Microsoft SDK*: Fornece reconhecimento de voz em inglês e espanhol, *Skeleton Tracking* e o melhor kit de desenvolvimento. Porém, está disponível apenas na plataforma nativa Microsoft Windows. Maior suporte e maior comunidade de desenvolvimento.
ii) *OpenNI/ SimpleOpenNI:* Biblioteca de código aberto multi-plataforma. Funciona com Linux, Windows e Android. O OpenNI estabelece um conjunto de aplicações e rotinas (Application Programming Interface) cujo objetivo é não envolver em detalhes da implementação direta com o Hardware. O SimpleOpenNI é um envelopador que torna mais direto o desenvolvimento de aplicações. Por exemplo, torna mais simples o desenho dinâmico do esqueleto do usuário sem desenvolver um algorítimo para processar a imagem obtida pela câmera.
iii) *libfreenect*: É um conjunto de drivers/wrappers de nível baixo e não sistemas prontos para se utilizar, ou seja, é necessário programar a partir do zero utilizando todas as técnicas de processamento de imagem, embora haja códigos públicos para algumas tarefas como o *Skeleton Tracking.*

5.2. Descrição dos recursos e limitações respectivas aos softwares e bibliotecas

A principal limitação encontrada para o desenvolvimento do projeto é a falta de opções para reconhecimento de áudio. Dentre os quatro possíveis conjuntos de bibliotecas apenas o *Microsoft SDK* dispõe do recurso, embora este não esteja disponível em português. Uma alternativa é utilizar um sistema adjacente de reconhecimento de voz que pode ser integrado ao programa utilizando o Kinect apenas como um microfone padrão.
É importante destacar também a dificuldade de desenvolvimento multiplataforma.

5.3. Descrição das características de hardware do Kinect e especificações de funcionamento

São dispositivos componentes do equipamento são descritas abaixo e representadas na Figura 2.

1. Conjunto de quatro microfones. O arranjo destes microfones no dispositivo possibilita a detecção da distância entre o usuário e o aparelho por voz;
2. Um emissor e um detector de profundidade infravermelho. Possibilitam a captura de dados de profundidade. Estes dados são os principais a serem capturados uma vez que todo o processamento de gestos e sinais é feito tendo-os como base. Toda a informação é

armazenada em um vetor cujos valores representam a distância de um ponto representado por um pixel e o aparelho. Este vetor contém geralmente 307200 posições [3];

3. Uma câmera capturadora de cor;

As características físicas do equipamento geram alguns limites. São eles:

- Ângulo de visualização horizontal: 57º;
- Ângulo de visualização vertical: 43º;
- Intervalo de distância do usuário para o funcionamento correto: de 0,4 (Função Near Mode, disponível apenas no Kinect for Windows) a 4m;
- Intervalo de profundidade: 400 mm (Near Mode) a 8000 mm (modo convencional);
- Faixa de temperatura de 5 a 35 graus Celsius. O aumento da temperatura influencia no comprimento das ondas infravermelho, prejudicando o funcionamento.
- Necessidade de uma fonte externa de energia mesmo quando conectado ao computador;
- Resoluções:
- 640 x 480 a 30 FPS usando formato RGB;
- 1280 x 960 a 12 FPS usando formato RGB;
- 640 x 480 a 15 FPS usando formato YUV.

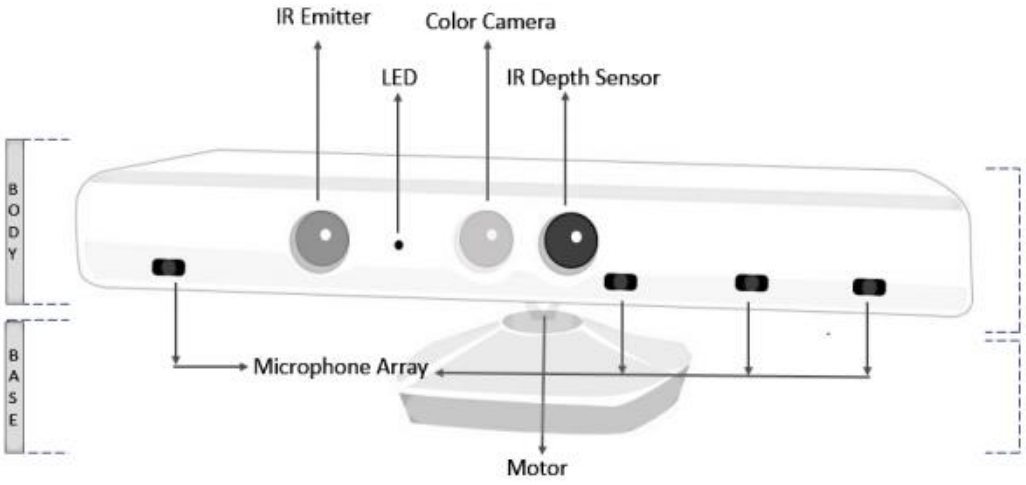


Figura 2: Disposição dos recursos de Hardware do Microsoft Kinect.

5.4. Escolha da linguagem de programação

A integração dos dados brutos obtidos pelos sensores é feita por um *framework:* o OpenNI. Os dados de profundidade são interpretados e assumem padrões que possibilitam identificar juntas do corpo e reconhecer gestos. A seguir, os dados processados pelo OpenNI são envelopados para Processing pelo SimpleOpenNI, cujo é embarcado em um mini-PC que também se comunica com um glicosímetro e com um medidor externo de pressão arterial. O fluxo de dados segue a Figura 3.

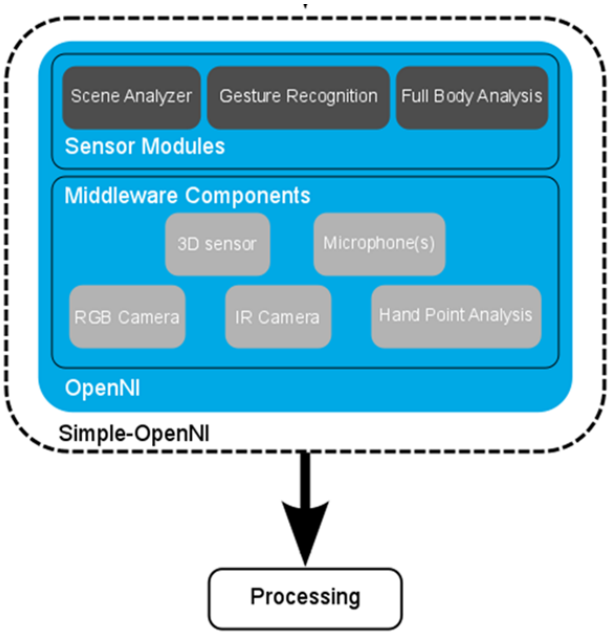


Figura 3 : Divisão em blocos dos componentes e envelopadores utilizados no trabalho.

5.5 Algoritmo de Cursor Virtual

Uma vez escolhidas e testadas as ferramentas do projeto, pode-se desenvolver o objetivo principal: os algoritmos.

Para construir formas usáveis de interação com o Kinect, é necessário escrever programas que respondem ao cenário de uma forma que as pessoas achem clara e objetiva. Todas as pessoas tem um sentido inerente dos objetos e do espaço à sua volta. Por causa deste sentido tão aguçado, interfaces físicas são radicalmente mais fáceis de intender do que as que possuem milhares de botões e janelas na tela. Especialmente no caso em que as interfaces que proporcionam um *feedback* na tela.

Tendo-se uma câmera de profundidade, é possível escrever interfaces como essa. Através do simples processamento dos dados de profundidade vindos do Kinect, pode-se fazer o usuário controlar um objeto da tela.

Há duas formas de se realizar esta tarefa: seguindo uma das juntas do mapa do esqueleto do usuário; e seguindo o ponto mais próximo do Kinect. A primeira opção é de fácil implementação mas gera um problema para possíveis pacientes crônicos acamados. É necessário que o usuário esteja em pé e a cerca de um metro do Kinect.

O Kinect converte os dados de profundidade em números de 0 a 255 que representam tons de cinza. Cada cor na imagem de profundidade de saída representa uma distância de forma proporcional tal que 0 são os pontos mais afastados (8000mm) e 255 os pontos mais perto (400mm) do Kinect.

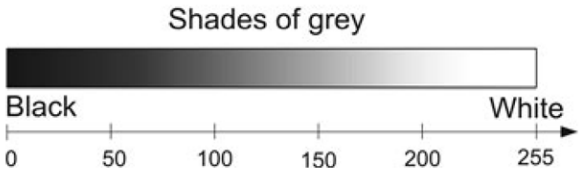


Figura 4: Divisão em tons de cinza

Porém, valores de 0 a 255 deixariam buracos e foi necessário acessar os dados brutos de profundidade do Kinect que são capturados com resolução de 11 bits, aumentando consideravelmente a precisão. A função *depthImage()* do pacote SimpleOpenNI retorna o vetor de 640x480 posições com esses valores já convertidos para distância física, em milímetros.

A função do algoritmo é encontrar qual é o ponto de menor profundidade dentro deste vetor a cada ciclo de *frame* de imagem. Para melhorar o resultado, algumas condições foram adicionadas:

i. O ponto deve estar entre 400mm e 650mm do Kinect;
ii. A variação instantânea da menor distância atual e a ultima distância não pode ser maior do que 100mm;
iii. Interpolação entre os dois últimos resultados.

5.6 Algoritmo de exercitação assistida

Agora, ao invés de realizar *loops* por pontos de profundidade para descobrir a posição do usuário, utilizou-se o recurso do OpenNI que detecta partes do corpo humano. Uma vez que o OpenNI detecta o usuário, ele retorna a posição das juntas do usuário que estejam visíveis à câmera: cabeça, pescoço, ombros, cotovelos, mãos, tórax, cintura, joelhos e pés.

Os movimentos físicos podem ser simulados com o Kinect através do monitoramento dos ângulos entre as juntas do usuário e das distâncias das respectivas juntas até outras juntas. O usuário realiza o movimento e o programa calcula o quão próximo o movimento que ele está realizando está perto do ideal. Uma vez cumpridas as exigências mínimas, o exercício é dado como concluído e move-se para o próximo exercício da lista. Dessa forma, a equipe médica pode definir uma lista de exercícios para cada paciente. O comando *map()* do Processing foi utilizado para gerar uma barra de progresso e realizar um feedback na tela para o usuário. Em alguns dos movimentos foi utilizado como referência o ângulo entre duas juntas e o eixo de pixels, cujos eixos positivos são leste o sul do plano cartesiano.

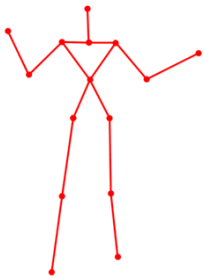
Foram definidas quatro posições no programa, as quais devem atender as exigências à seguir:

- Posição Normal:

a. Distância entre a mão direita e quadril direito: 10cm;
b. Distância entre a mão esquerda e quadril esquerdo: 10cm.

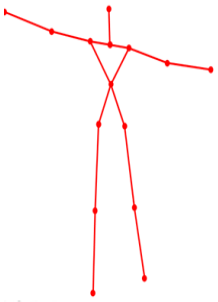
- Posição Psi:

a. Distância entre a mão direita e quadril direito: 90cm;
b. Distância entre a mão esquerda e quadril esquerdo: 90cm;
c. Ângulo do cotovelo esquerdo: 90°;
d. Ângulo do cotovelo direito: 90°.



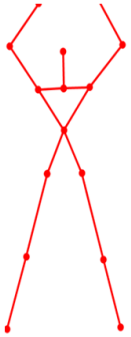
- Posição Pi:

a. Distancia entre mão esquerda e mão direita: 1500 cm;
b. Ângulo entre mão direita e mão esquerda (com relação ao eixo de pixels): 55°.



- Posição Delta:

a. Posição Psi;
b. Ângulo entre as pernas: 22°.



5.7 Alternativas de processamento de imagem em sistemas embarcados

As bibliotecas nativas do Kinect e o OpenNI dispõem de ferramentas de alto nível para processamento de imagens. Porém, o custo computacional é muito alto para sistemas embarcados. Além disso, o SimpleOpenNI está disponível apenas para sistemas de arquitetura 64bits, pouco comuns em sistemas embarcados.

Algumas alternativas foram levantadas. Uma delas é o uso direto da câmera de profundidade para realizar aproximações em 3D. É possível realizar estimações para as juntas do corpo com uma base de dados e extrair os pontos principais com uma variante do algoritmo de Dijkstra [5].

Outra alternativa testada foi o uso do algoritmo OpenTLD. A diferença neste caso é que o TLD não necessita de treinamento off-line, sendo necessário para o início do seu funcionamento apenas a seleção do objeto de interesse. Na Figura 4, apenas o rosto do usuário foi selecionado e o TLD manteve-se constantemente selecionado, mesmo com o rosto em movimento, o que pode ser útil para o desenvolvimento do cursor virtual.

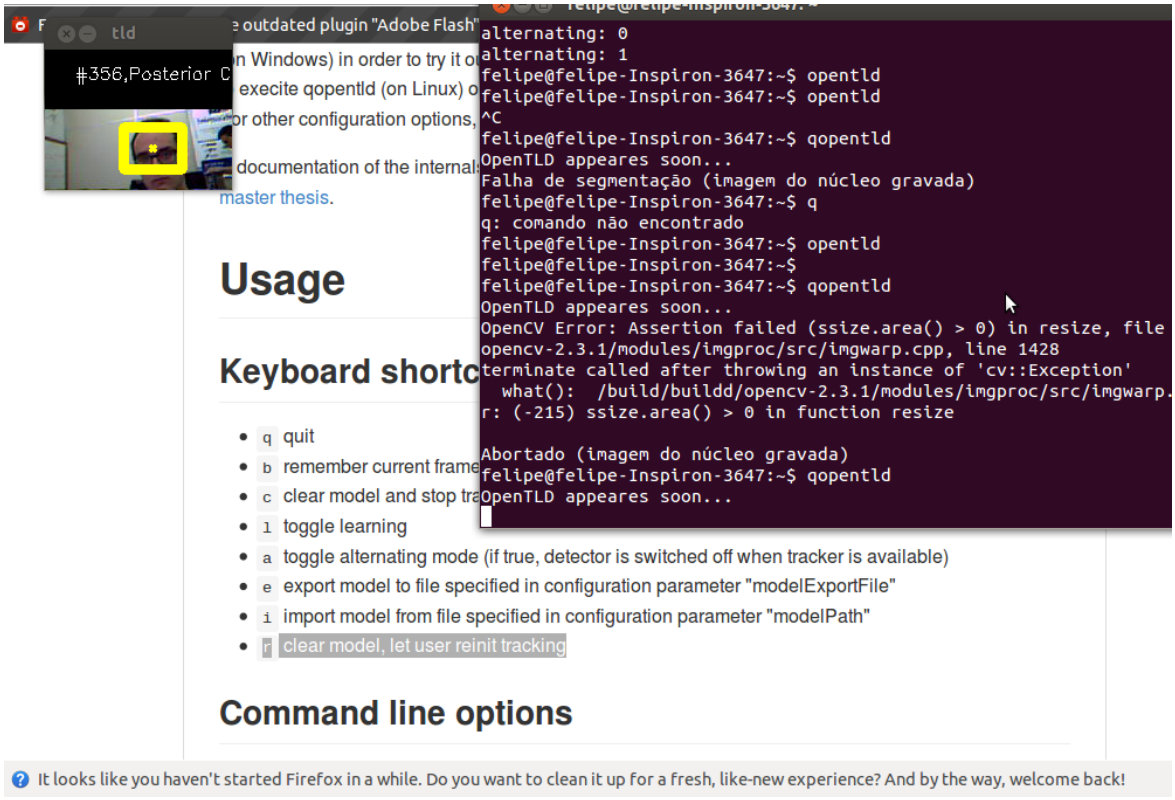


Figura 4: Exemplo de detecção de um ponto de interesse utilizando o OpenTLD. A caixa amarela é exibida enquanto o ponto é detectado.

5.8. Estendendo o reconhecimento de gestos para uma abordagem por sistemas embarcados

As grandes desvantagens da utilização de um computador para processar as informações de vídeo e movimento são o encarecimento do projeto e a menor portabilidade. Já é possível encontrar dispositivos com poder de processamento suficiente para as tarefas de análise de imagem que são menores, de mais fácil transporte e mais baratos do que um computador.

A etapa seguinte do projeto foi a de analisar a viabilidade destes dispositivos e a compatibilidade com os hardwares e softwares necessários. Ainda neste contexto, a preferência dos sistemas utilizados é dada àqueles cujas licenças são do tipo aberta ou não limitantes para usos científico, distributivo ou comercial.

5.8.1. Diferenças básicas na transição das plataformas

As diferenças entre as duas plataformas começam no nível mais baixo de hardware: a arquitetura dos processadores.

Para microcomputadores, existem basicamente duas arquiteturas universais: *i386* e *amd64*. A primeira refere-se ao padrão de microprocessadores de 32 bits criado pela Intel em 1985 que faz uso

do conjunto de instruções x86 CISC (Complex Instruction Set Computing) enquanto amd64 (ou X86-64) é uma versão de 64 bits deste mesmo conjunto de instruções que tornou os programas capazes de armazenar quantidades maiores de dados na memória.

Em 2015, a memória RAM média disponível de fábrica para microcomputadores do tipo *desktop* é cerca de 8GB, quantidade que não pode ser endereçada por um sistema operacional mais comum (Windows) em arquiteturas do tipo i386. Dessa forma, adm64 é o padrão.

Para os sistemas embarcados, porém, a arquitetura de processador dominante é a *arm* desenvolvida pela ARM Holdings e cujo conjunto de instruções é do tipo RISC (Reduced Instruction Set Computing).

Um exemplo da diferença entre arquiteturas RISC e CISC é o número de ciclos de *clock* necessários para completar uma instrução. Em CISC, as instruções são complexas e podem demorar mais do que um ciclo para serem completadas. Na arquitetura RISC todas as instruções devem tomar um número de ciclos de *clock* fixo (1 ciclo, por exemplo).

Já que todas as instruções na arquitetura RISC executam um intervalo de tempo uniforme, é possível realizar a divisão de tarefas do processador em *pipelining*. Além disso, menos transistores do espaço de hardware são necessários para acomodar as instruções RISC, tornando possível reservar mais espaço para os registradores de uso geral.

Apesar das vantagens citadas acima, os processadores de arquitetura RISC (*arm)* ainda são menos difundidos do que aqueles que herdaram os conjuntos de instrução x86 por um motivo: a incompatibilidade de software.

Cada um dos itens listados na seção 5.1 possui um conjunto de dependências de bibliotecas. O *Microsoft Kinect SDK*, por exemplo, depende do .NET Framework enquanto o *OpenNI* requer FreeGLUT3. Essas dependências são limitadas pelo tipo de instrução do processador e/ou pelo sistema operacional. O .NET Framework é exclusivo do S.O. Windows enquanto o FreeGLUT3 é um pacote de ferramentas e extensões para a execução e criação de programas gráficos baseados em OpenGL que não está disponível na arquitetura *arm*.

Alternativas para as bibliotecas citadas na seção 5.1 e ao Kinect foram abordadas. Uma vez que o problema está no nível mais baixo de hardware, a sua solução deve ser encontrada através do nível mais baixo de software.

Para dar continuidade ao projeto e criar um ambiente de trabalho, foi necessário abrir mão de todas as bibliotecas não compatíveis com os processadores RISC e buscar soluções semelhantes que possam ser compiladas com o kernel *arm.*

5.8.2. Criação de um ambiente de trabalho nos sistemas embarcados

As soluções do projeto foram baseados em dois hardwares: Mini-PC MK 809 III e o Raspeberry Pi 2, cujas instalações são abordadas nas duas seções a seguir. Embora hoje seja possível encontrar

sistemas *Embedded Windows* e *Windows 10 IoT* embarcáveis, a escolha do sistema operacional foi a do tipo Linux devido às facilidades enquanto aos termos de licenciamento e a customização.

5.8.2.1 MK 809 III

O dispositivo MK809 III é um mini-pc da fabricante dongle que possui altos recursos de fábrica para o processamento de imagens. O sistema funciona como um computador que, quando conectado a um aparelho de TV pela entrada HDMI, utiliza este como monitor.

Recurso	**Especificação**
CPU	Rockchip RK3188, Cortex A9. Quad Core. 1.8Ghz
O.S.	Android 4.4.2
Aceleração Gráfica	Mali400, 1080p FULL HD
HDMI	1
USB	1x USB 2.0 Host, 2x Micro USB
Armazenamento	Micro SD
Wifi	802.11b/g/n
RAM	2GB

O primeiro passo para a utilização do MK 809 III foi a troca do sistema operacional. É necessária uma plataforma Linux para a instalação de bibliotecas de desenvolvimento e processamento de imagem.

Para cumprir este objetivo, existem duas alternativas: o *boot* do sistema pelo cartão micro-sd e o *image flashing* da memória interna do dispositivo que sobrescreve o sistema operacional Android e torna o Linux o *boot* padrão. No segundo caso, não é possível realizar *dual-boot* entre Linux e Android.

O primeiro procedimento realizado foi o de *boot* pelo cartão SD. A imagem de sistema operacional é fornecida por uma iniciativa livre (*Linuxium)* e deve ser gravada no cartão SD de forma que este se torne uma nova partição de sistema. Em seguida, é necessário apenas inserir o cartão na entrada micro-sd do mini-pc.

Após feita a montagem, realizou-se testes de compatibilidade de arquivos e bibliotecas, onde foi possível diagnosticar um problema. A imagem pré-compilada do sistema operacional não incluía uma parte essencial do sistema: as ferramentas de compilação de driver fornecidas pela fabricante do processador, também conhecidas como *kernel build-essentials*.

Ou seja, não era possível compilar drivers de vídeo, internet e câmera dentro do sistema operacional o que impossibilitou os testes de performance do *opentld*.

Sendo assim, abordou-se a segunda alternativa. A fabricante Radxa Rock do processador RK3188 fornece um guia de instalação ou recuperação do sistema operacional [6]. De acordo com o manual, este processo pode ser realizado quantas vezes forem necessárias pois o sistema continuará em funcionamento, a não ser em caso de dano físico no Hardware.

Neste procedimento, as imagens de kernel, sistema operacional, configuração e restauração são gravadas separadamente em trechos da memória. Na primeira opção, por exemplo, a distribuidora *Linuxium* uniu todas as imagens em uma única imagem de sistema. Possivelmente, a partição *kernel.img* foi corrompida ou se tornou incompatível após este processo de união.

Porém, o manual da fabricante faz referência às suas placas completas (*Radxa Rock Board*) e o mini-pc faz uso apenas do processador RK3138. Por este motivo, encontrou-se incompatibilidade entre sistema operacional fornecido para gravação e o mini-pc, que aloca a memória de sistema operacional em uma região diferente de memória. A opção a seguir foi adotar o outro tipo de Hardware disponível.

5.8.2.2 Raspberry Pi 2

Recurso	**Especificação**
CPU	Broadcom BCM2836 Arm7 Quad Core @ 900Mhz
O.S.	Wheezy Raspbian
USB	4 x USB 2.0 + Micro USB Power
Armazenamento	Micro SD
Wifi	Não. 10/100 Ethernet
RAM	1GB

No Raspberry Pi 2, a maioria dos problemas de compilação foram resolvidos principalmente porque o sistema operacional Wheezy Raspbian fornecido pela fabricante já o sistema operacional padrão da placa.

Ou seja, as ferramentas de compilação de driver estão presentes e a maioria dos drivers necessários, como o *gspca* para captura de vídeo, já são previamente instalados.

Assim, foi possível instalar, compilar e testar os softwares necessários: *opencv e opentld*.

5.8.3 Algoritmo de reconhecimento de imagens: o *TLD*

O termo *TLD* refere-se a “Tracking-Learning-Detection” e é uma alternativa proposta para investigar objetos desconhecidos em um fluxo de vídeo de longo prazo [7]. Um objeto de interesse é definido em um único frame de vídeo e, em cada frame subsequente, o processo determina a localização do objeto na imagem, caso exista, ou indica a sua ausência.

Assim, é possível seguir o objeto na imagem. O detector atualiza todas as aparições do objeto no fluxo de vídeo e ajusta a localização do objeto, se necessário.

O sistema *TLD* se baseia nos três componentes a seguir.

5.8.3.1 Rastreamento de Objetos (*Tracking)*

O rastreamento de objetos ou padrões em imagens é uma tarefa importante no campo da visão computacional. O aumento dos recursos computacionais aumentou a viabilidade deste processo que já pode ser realizado com câmeras e computadores de baixo custo.

Basicamente, o rastreamento pode ser definido como o problema de estimar a trajetória de um objeto num plano de imagem à medida em que ocorre o fluxo de vídeo. Em outras palavras, um rastreador associa rótulos consistentes aos objetos rastreados em diferentes frames de vídeo. [8]

A tarefa de rastrear objetos pode ser complexa devido aos seguintes fatores:

- Perda de informação na projeção do mundo em 3D num plano 2D;
- Ruídos nas imagens (Ex.: pontos de alta luminosidade);
- Movimentos complexos do objeto (Ex.: velocidade de movimento próxima ou maior que o número de frames por segundo);
- Natureza dinâmica e articulada dos objetos;
- Ocultação parcial ou total do objeto;
- Mudanças de iluminação de fundo;
- Necessidade de processamento em tempo real.

5.8.3.2 Detecção de objetos (*Detection)*

A detecção de objetos é a tarefa de localizar objetos em uma imagem. A definição de “um objeto” pode variar entre um único grupo de pontos a uma classe de objetos [7]. A detecção de objetos é tipicamente baseada em Características Locais, que são descritores da vizinhança de uma imagem local processados com múltiplos pontos de interesse.

As aproximações baseadas em Características geralmente seguem a seguinte ordem de trabalho:

1. Detecção das características;
2. Reconhecimento das características;
3. Modelagem.

5.8.3.3 Aprendizado de Máquina (*Learning)*

Os detectores de objetos são tradicionalmente treinados assumindo que todos os dados de treinamento são classificados.

Para viabilizar o reconhecimento de diferentes objetos em tempo real, a abordagem de aprendizado de máquina utilizada foi a de Expectativa-Maximização (EM). Este é um método iterativo utilizado para encontrar a probabilidade máxima posteriori Bayesiana (MAP).

O autoaprendizado começa com a classificação inicial de um modelo de treinamento. No programa *OpenTLD*, esta classificação é feita através de uma imagem que pode ser desenhada pelo usuário em tempo real em um frame do vídeo ou importada por uma imagem modelo.

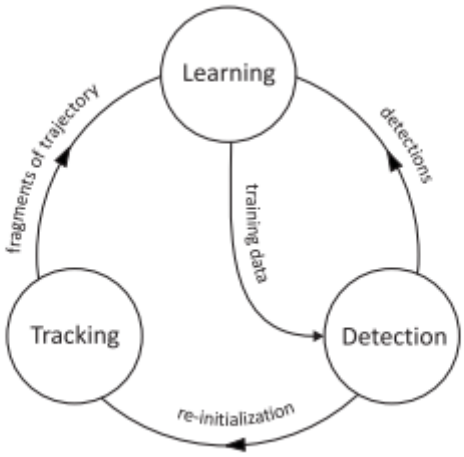


Figura 5: *Diagrama de Blocos TLD. Fonte: "Tracking-Learning-Detection" [7].*

A principal característica da ferramenta TLD é a aprendizagem por P-N, cujo objetivo é realizar a detecção do objeto em tempo real. Essa aprendizagem utiliza dois detectores de erro, um para identificar falsos negativos e outro para identificar falsos positivos.

O número de falsos positivos e falsos negativos são classificados iterativamente por equações recursivas.

Qualitativamente, é possível criar uma matriz em função dos indicadores de erro. Os diferentes autovalores dessa matriz tornam possível a recuperação de informações sobre o comportamento do aprendizado do objeto (Figura 6).

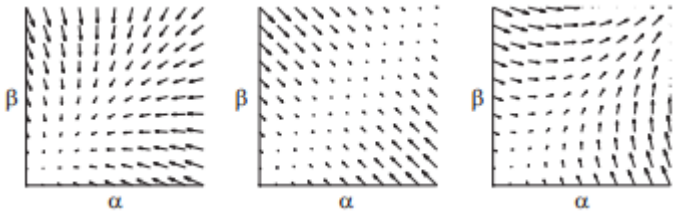


Figura 6: *Evolução dos erros: Erro decrescente (esquerda), estático (meio) e crescente (direita). α e β são, respectivamente, os falsos negativos e positivos. Fonte: "Tracking-Learning-Detection" [7].*

A formação destes gradientes de erro ao mesmo tempo que possibilita a detecção de objetos complexos em uma imagem, dificulta a detecção de objetos com pouca complexidade.

No caso deste trabalho, é desejável detectar o contorno de uma mão humana para discretizar o movimento de um mouse. Porém, quando há baixa variabilidade de detalhes no objeto de interesse, os coeficientes α e β, que são recursivos, irão convergir sempre para um mesmo valor pois a escala de cinza de uma mão pode ser facilmente confundida com a cor de uma parede por este método.

6. Atividades realizadas no período

I) Busca de bibliotecas e *drivers* de acesso ao Kinect nas plataformas Linux, Windows e Android;

II) Descrição dos recursos e limitações respectivas aos softwares e bibliotecas;

III) Descrição das características de hardware do Kinect e especificações de funcionamento;

IV) Participação em grupos de discussão sobre o desenvolvimento de aplicativos Kinect;

V) Aquisição de equipamentos;

VI) Escolha da linguagem de programação;

VII) Testes com os recursos de *skeleton tracking* e de imagens de profundidade (*depth images)*;

VIII) Criação de um algoritmo de cursor virtual *hands-free* em tempo real;

IX) Criação de um algoritmo de exercitação assistida;

X) Apresentação dos resultados em palestras/workshop.

XI) Análise dos algoritmos em sistemas embarcados e soluções para implementação de processamento de imagens em sistemas de baixo custo.

XII) Instalação e Testes de Hardware e Software: Mini-Pc

XIII) Instalação e Testes de Hardware e Software: Raspberry Pi 2

XIV) Análise do algoritmo TLD. Aplicação em reconhecimento de movimento de uma mão.

7. Cronograma do Projeto

Atividade (2014)	JAN	FEV	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV	DEZ
I	X											
II	X											
III		X										
IV		X	X									
V			X	X								
VI			X	X								
VII					X							
VIII						X	X					
IX								X	X	X		
X											X	
XI											X	X

Atividade (2015)	JAN	FEV	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV	DEZ
XII	X	X	X									
XIII				X	X							
XIV					X	X						

8. Resultados e discussão

O algoritmo de detecção reconhece as quatro posições apresentadas na Figura 7. É possível identificar padrões com base nos ângulos relativos a duas juntas do esqueleto que são vistas como pontos no espaço de profundidades. A precisão é ajustável mudando coeficientes de pesos das contribuições das distâncias euclidianas entre duas juntas e dos ângulos característicos dos membros do esqueleto em cada movimento. O sensor deve estar a 1,50m do usuário, dado que pode variar de acordo com a envergadura do mesmo.

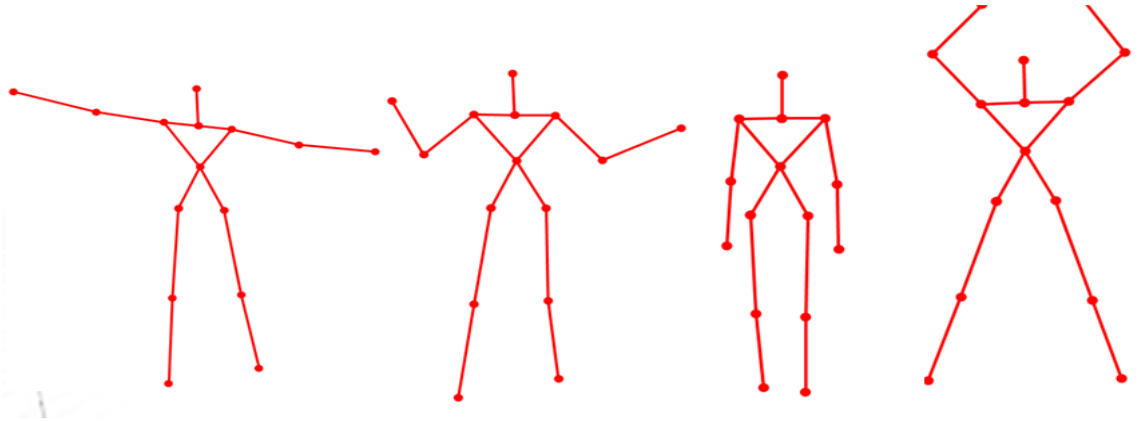


Figura 7: Sequência de movimentos de exercício.

A movimentação do mouse virtual é dada de duas formas: rastreando a junta de uma das mãos do usuário, se este estiver afastado do sensor ou rastreando-se o ponto mais próximo do Kinect, caso no qual o usuário aproxima a mão ao sensor. A amostragem discreta dos pontos mais próximos é corrigida por interpolação linear resultando em um movimento contínuo. Os pontos mais próximos à câmera são os mais claros na Figura 4. Um círculo é desenhado tendo como origem o pontos mais próximo do sensor.

Comparando-se com os resultados obtidos em [4] a precisão do algoritmo de simulação do cursor deste projeto é consideravelmente melhor porque as características de discretização foram amortecidas pela interpolação.

O maior desafio do projeto é o processamento dos dados do sensor em um mini-PC. Os softwares padrão de reconhecimento de imagem dependem de uma taxa de atualização maior do que 20 FPS. A performance nos hardwares tipo mini-PC não atingiu velocidade suficiente. Sendo assim, a próxima

etapa do projeto é a criação de um *software* próprio que se ligue diretamente aos Componentes de Middleware utilizando o OpenCV. O fator dificultante na realização desta tarefa é a diferença de arquiteturas entre a plataforma PC e chips Arm que torna difícil a utilização do Kinect. Uma nova possibilidade é a abordagem do sistema operacional Microsoft IOT para Raspberry Pi 2, que traz consigo o ambiente .NET Framework da Microsoft embarcado.

A aplicação da técnica de TLD pode ser aplicada com sucesso para superfícies com riqueza de detalhes: uma face, um indivíduo andando pela rua ou um modelo de veículo específico em meio ao trânsito. A detecção de um objeto com poucas características é dificultada pelo método P-N de aprendizado.

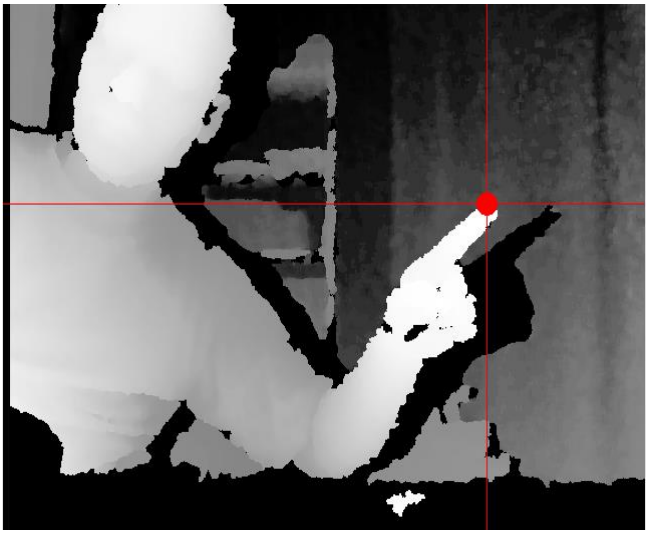


Figura 8: Exemplo de cursos virtual.

9. Referências

[1] Celić, L., Trogrlić, D., Paladin, I.,Prašek, M., Magjarević, R. “Integration of measurement devices supporting diabetic patients into a remote care system.” IFMBE Proc. 2011, pp. 39-42.

[2] Magjarevick, R.; Celic, L.; Drobjnak, S.; “Integration of Physical Activity Monitoring into eHealth Care for Chronic Patients**”.** IEEE.

[3] G. Borenstein. “Making Things See: 3D vision with Kinect, Processing, Arduino and MakerBot”. Vol1, no.1, pp. 43-299, 2009.

[4] CATUHE, D. “Programming with the Kinect for Windows Software Development Kit”. Microsoft Press. 224 p. 2013.

[5] Baak, A.; Müller, M.; Bharaj G.; Seidel H.; Theobalt C.; “A Data-Driven Approach for Real-Time Full Body Pose Reconstruction from a Depth Camera”. 2011 IEEE International Conference on Computer Vision, ICCV 2011; Barcelona; Spain; 6 November 2011 through 13 November 2011; Code 88388. 1092-1099p.

[6] http://radxa.com/Rock/flash_the_image. Como acessado em 20/06/2015

[7] Kalal, Z.; Mikolajczyk, K.; Matas, J.; “Tracking-Learning-Detection”. IEEE Transactions on pattern analysis and machine intelligence. Vol. 6, No. 1. 2010.

[8] Yilmaz, A., Javed, O., and Shah, M. 2006. “Object tracking: A survey”. ACM Comput. Surv. 38, 4, Article 13 (Dec. 2006), 45 p.

[9] http://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/. Como acessado em 16/06/2015

10. Participação em eventos técnico-científicos

I) Palestra:

Sistemas de análise de movimentos aplicados à Engenharia Biomédica.

Prof. Dr. Anselmo Frizera Neto, UFES

Data e Hora: 19/03 | 14:00

II) Workshop:

Workshop Internacional de Engenharia Biomédica

@ http://www2.ele.ufes.br/~anselmo/Workshop/index.html

Data: 12 a 14 de Novembro de 2014

11. Outras atividades

ANEXO I

AVALIAÇÃO DO ORIENTADOR QUANTO AO DESEMPENHO DO BOLSISTA

No quadro abaixo, o orientador deve avaliar o desempenho do bolsista, indicando o grau de atendimento de suas expectativas. Caso as expectativas originais não tenham sido atendidas, indicar o(s) provável (eis) fator(es) responsável(eis) pelo baixo desempenho do bolsista (por exemplo, falta de tempo, falta de conhecimento, falta de iniciativa). Apagar estas observações em vermelho após preenchimento.

PARECER

Data

Assinatura do orientador(a)