MINERAÇÃO DE COMPORTAMENTO DE CONSUMIDORES ATRAVÉS DA CAPTURA DE MOVIMENTOS VIA MICROSOFT KINECT[1]

Alessandro Tyska Menezes[2] <aletyska@gmail.com>
Stanley Loh[3] <stanley.loh@ulbra.edu.br> – Orientador

Universidade Luterana do Brasil (Ulbra) – Curso de Sistemas de Informação – Campus Canoas
Av. Farroupilha, 8.001 – Bairro São Luís – CEP 92425-900 – Canoas - RS

28 de Novembro de 2013

RESUMO

Este trabalho descreve o desenvolvimento de uma ferramenta para monitoração automatizada de clientes em compras em lojas físicas. Através do sensor 3D *Microsoft Kinect*, serão capturados os movimentos executados pelos clientes. Após essa etapa, a ferramenta utilizará uma implementação do algoritmo Apriori, minerando os padrões sequenciais, a fim de descobrir os movimentos mais executados pelos clientes.

Palavras-chave: Mineração de Dados, *Microsoft Kinect*, Clientes em compras, Algoritmo Apriori.

ABSTRACT

Title: *"Consumer Behavior Mining Through Movement Capture With Microsoft Kinect."*

This paper describes the development of a tool that automatically monitors shopping customers in physical stores. The consumer movements will be captured by using the Microsoft Kinect 3d sensor. After that, it will use the Apriori algorithm implementation, in order to search sequential patterns and find the most executed consumer movements.

Key-words: *Data Mining, Microsoft Kinect, Consumers Shopping, Apriori algorithm.*

1 INTRODUÇÃO

Atualmente, o setor de comércio vem crescendo e sendo um fator chave na economia brasileira, uma vez que movimenta grandes quantias de dinheiro e gera grande número de empregos. Entretanto, segundo estudo publicado pelo Serviço Brasileiro de Apoio às Micros e Pequenas Empresas (SEBRAE, 2011), 26,9% das empresas fecham as portas após dois anos de sua abertura. O estudo mostra, também, que um dos principais fatores para a mortalidade das empresas são as falhas gerenciais. Isso mostra um despreparo das empresas para enfrentar a concorrência que está imposta no cenário brasileiro atual.

Vivemos em uma realidade cada vez mais informatizada no âmbito pessoal, acadêmico e, principalmente, coorporativo. Atualmente, devido à facilidade de acesso, sistemas de informações gerenciais são vistos como obrigatórios para organizações e não mais como um diferencial. Qualquer loja hoje em dia possui um computador com um sistema de controle de vendas, que efetua pagamentos através de cartão. O estabelecimento que não possuir esse recurso será invariavelmente abatido pela concorrência. Além disso, com o crescimento exponencial da internet, as compras online vêm se tornando uma grande concorrência para as lojas físicas. Além dos clientes terem a facilidade de não sair de casa, os sites de venda conseguem monitorar os seus passos durante a compra e assim, personalizar a loja virtual, levando o cliente a comprar mais, ou comprar itens que gerem maior lucro.

Enquanto isso, em lojas físicas, o monitoramento do comportamento de clientes é bastante complicado. Underhill (2008) fez um estudo do comportamento de clientes em compras utilizando a observação. Através de gravações de vídeo e observações feitas em lojas, ele foi capaz de descobrir padrões no comportamento dos clientes e sugerir melhorias para os locais analisados.

---

1 Proposta de Trabalho de Conclusão de Curso em Sistemas de Informação, submetida ao Curso de Sistemas de Informação da Universidade Luterana do Brasil, Câmpus Canoas.

2 Bacharelando em Sistemas de Informação da Universidade Luterana do Brasil.

3 Professor dos Cursos de Computação da Universidade Luterana do Brasil.

Este trabalho tem como objetivo apresentar uma ferramenta de *software* que possibilite a identificação de movimentos executados por clientes em frente a uma gondola de vendas. Para essa tarefa, será utilizado o sensor 3D *Microsoft Kinect*, bem como a sua *Application Programming Interface* (API, daqui por diante). Depois de identificados e capturados, estes movimentos serão minerados para que possam ser identificados padrões estatísticos. Será utilizado o algoritmo Apriori, definido por Agrawal e Srikant (1995), para identificar qual a sequência de movimentos mais executada numa base de movimentos. Por exemplo, a sequência 'cruzar os braços, estica braço direito e estica braço esquerdo', caso seja uma sequência detectada pelo Apriori, pode significar que o cliente cruzou os braços em dúvida e pegou dois produtos.. Além disso, serão feitas algumas estatísticas simples para que seja possível entender melhor a base de movimentos coletada, como por exemplo a média de clientes por dia, os movimentos mais executados, entre outros.

2 REFERENCIAL TEÓRICO

Esta seção apresenta toda a base teórica dos termos, conceitos e tecnologias utilizadas neste trabalho. Primeiramente, na seção 2.1 será falado sobre o conceito de sensores 3d. Na seção 2.2, será comentado o funcionamento do sensor 3D utilizado neste trabalho, o *Microsoft Kinect*. Já na seção 2.3, será explicado o processo de mineração com ênfase no algoritmo Apriori, que extrai as sequências mais frequentes em uma base de transações.

2.1 Microsoft Kinect

O *Kinect* foi originalmente construído para a plataforma de jogos Xbox e posteriormente, foi feita uma versão para *Windows*. Segundo CRAWFORD(2010) o *Kinect* é constituído por câmeras coloridas, sensor infravermelho de profundidade e um *array* de microfones, que faz a leitura do ambiente ao qual ele se encontra. Aliado a um conjunto de algoritmos embarcados no sensor, ele gera diversos tipos de informações como monitoramento dos movimentos do usuário, profundidade e reconhecimento de fala. Estas informações são entregues ao computador através de uma interface USB 2.0, no formato de Streams (MICROSOFT, 2012).

2.1.1 Especificação de Hardware do Kinect

Para captar as informações de movimento, profundidade e fala, o *Kinect* possui uma série de sensores com funções específicas. Através de sua câmera padrão *Red, Green, Blue* (RGB daqui por diante), ele consegue fazer a captura de imagens comuns em uma resolução de 1280x960 a 30 frames por segundo. É através dessa câmera que é feita a detecção de movimentos em duas dimensões, a partir da detecção das 20 principais juntas de movimento do corpo. Há também um emissor de ondas infravermelhas que, aliado ao sensor infravermelho de profundidade, fará a leitura da terceira dimensão. Dessa forma, capta-se movimentos do usuário em todas as três dimensões, sabendo quais partes do corpo estão mais próximas ao dispositivo (MICROSOFT, 2012). Além disso, há um *array* de quatro microfones que fazem a captura do áudio. Por ser um *array,* é possível determinar de qual direção o som é emitido. O IR Emitter emite raios infravermelho para que o IR Depth Sensor possa receber os estímulos gerados. Eles são responsáveis pelos valores de profundidade de um determinado ponto. Já o Color Sensor, é responsável pela imagem do sensor, bem como monitoramento de coordenadas X e Y dos pontos monitorados pelo sensor.

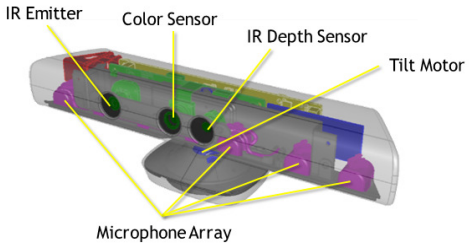

Figura 1 – Sensores do Microsoft Kinect (MICROSOFT, 2012)

O *Kinect* tem um campo de visão vertical de 43,5°, que pode ser alterado através de um motor chamado de *Tilt Motor,* que altera o ângulo vertical de leitura em 27° para ambos os sentidos. Além disso, o sensor consegue monitorar usuários à distância de 0,4m até 3,5m do dispositivo. Todas as leituras feitas pelo *Kinect* são entregues ao computador através de uma conexão *Universal Serial Bus* (USB daqui por diante).

Os dados gerados no sensor são enviados na forma de *data streams*, que será explicado no próximo item (MICROSOFT, 2012).

2.1.2 Data Streams

Todas as informações geradas pelo *Kinect* são disponibilizadas para processamento de um programa através do que é chamado de *Data Streams.* Esse é o termo utilizado para a forma como o *Kinect* libera as informações de áudio, cor, profundidade e esqueleto.

O *audio stream* fornece os dados de áudio que são capturados pelo *array* de quatro microfones embutidos no dispositivo. Por ser um *array*, ele consegue detectar de qual direção o som vem e associar este som com o esqueleto que é monitorado. Isto é importante, pois caso haja dois usuários operando a mesma aplicação, é possível identificar qual usuário disse determinado comando. Ele também possui um avançado mecanismo de reconhecimento de fala, para que possa ser interpretado como comandos (MICROSOFT, 2012).

O *color stream* concede informações de imagem. Esses dados podem ser fornecidos em diversos formatos de cor, resolução e *frame rate*. Resoluções maiores tem um *frame rate* menor, pois mandam uma maior quantidade de informações por frame. Com o formato RGB, pode-se conseguir uma resolução de 640x480, com 15 ou 30 frames por segundo, além da resolução 1280x960, com 15 frames por segundo. Já com o formato YUV, há apenas a resolução de 640x480 a 15 frames por segundo. Além desses, há ainda o formato *Bayer* de cor, que fornece resoluções de 1280x960 a 12 frames por segundo e 640x480 a 30 frames por segundo (MICROSOFT, 2012).

O *depth stream* gera informações de profundidade dos usuários monitorados. Elas são fornecidas através de *frames,* similar ao *color stream.* Cada pixel do frame fornece a distância, em milímetros, que se encontra do *Kinect*. Além disso, cada pixel possui o *Player Segmentation Data*, que é um *id* fornecido para usuário monitorado. Dessa forma, é possível detectar quais partes do corpo de cada usuário estão mais próximas, aumentando a gama de comandos que podem ser interpretados. Além disso, este mapeamento é importante para que o dispositivo possa detectar e descartar objetos parados atrás dos usuários monitorados. O monitoramento de esqueleto utiliza as informações de profundidade. Elas podem ser lidas em diferentes resoluções, sendo a maior e *default* 640x480 (MICROSOFT, 2012).

Por fim, o *Kinect* fornece o monitoramento do esqueleto do usuário, facilitando a leitura dos movimentos, que é chamado de *skeletal tracking.* O esqueleto é gerado através da detecção de vinte pontos de articulações chaves do corpo humano, desde a cabeça, até os pés. O *Kinect* consegue detectar seis usuários, mas consegue monitorar o esqueleto de apenas dois simultaneamente. O monitoramento pode ser feito de dois modos distintos. No modo *default,* o *Kinect* parte do principio que o usuário está em pé e busca monitorar todas as 20 juntas. Já no *seated mode,* ou modo sentado, o *Kinect* parte do princípio que o usuário está sentado. Logo, ele monitora apenas as juntas superiores do usuário, como cabeça, braços e mãos, totalizando dez juntas. Isso possibilita que o dispositivo seja utilizado tanto para jogos quanto para operação de um *software* com interface específica para *Kinect* (MICROSOFT, 2012).

2.1.3 SDK Existentes

Para o desenvolvimento de *softwares* que utilizem *Kinect*, é necessário um *Software Development Kit (*SDK, daqui por diante) para comunicação com o dispositivo. Atualmente, existem duas *SDK* disponíveis para este tipo de desenvolvimento, a *Microsoft SDK for Kinect* e a *OpenNI SDK*.

A *Microsoft SDK for Kinect* (MICROSOFT, 2012) é o SDK oficial do dispositivo, pois foi desenvolvido pelo seu fabricante, a *Microsoft*. Já a *OpenNI* é um SDK alternativo. Basicamente, ambos fornecem suporte para todas as funcionalidades do *Kinect*. A grande diferença é que a *OpenNI SDK* é multiplataforma, ou seja, pode ser executado em outros sistemas operacionais além do *Microsoft Windows*. Entretanto, a *Microsoft* fornece suporte apenas para *softwares* desenvolvidos com seu SDK oficial. Assim, este projeto será desenvolvido utilizando o *Microsoft SDK*.

2.1.4 Exemplos de Utilização Acadêmica

Apesar do apelo comercial, o *Kinect* vem sendo utilizado de diversas formas no meio acadêmico, para as mais diversas finalidades. Serão descritos agora, exemplos pesquisados que mostram as diversas aplicações do sensor.

Está sendo desenvolvido, como tese de mestrado na Pontifícia Universidade do Paraná (PUCPR),

um método que utiliza os sensores do *Kinect* para interpretação e tradução de sinais da Linguagem Brasileira de Sinais (LIBRAS). Ele se utiliza de quatro etapas, que são: aquisição do sinal, identificação da configuração de mão, rastreamento das mãos e classificação do sinal (MENDONÇA, 2012).

Outra interessante utilização do dispositivo foi desenvolvida em Taiwan. Foi feito um *software* – utilizando o *Kinect* – para auxiliar na recuperação de dois jovens pacientes com deficiências motoras, em uma escola pública especializada em alunos deficientes. O *software* pedia para que os alunos executassem uma rotina de movimentos, que conforme fossem cumpridos, gerava uma interação do *software* com o aluno, através de sons e imagens (CHANGA, CHENB e HUANGC, 2011).

No 22º Simpósio Brasileiro de Informática na Educação, foi apresentado um estudo conjunto da Universidade Federal do Sergipe e Universidade Federal da Bahia, de como o *Kinect* pode ser utilizado para as características afetivas de um usuário e de que forma isto pode auxiliar na educação (NUNES et al., 2011).

A seguir, descreve-se a definição de Mineração, bem como qual método é utilizado neste trabalho.

2.2 Mineração de Dados

A utilização em massa dos sistemas de informação fez com que eles fossem evoluindo com o tempo. Nos primórdios da computação, os sistemas eram bastantes rudes, com poucos recursos e poucos usuários, com interfaces que não eram muito amigáveis. Atualmente, as interfaces são bastante intuitivas e, além disso, os recursos fornecidos por sistemas computacionais e o número de usuários cresceram muito. Com isso, a massa de dados manipulada por esses sistemas aumentou exponencialmente, bem como a sua importância para os negócios. Com a Mineração de Dados (MD, daqui em diante) foi possível tornar os sistemas, ferramentas poderosas para o suporte à decisão de níveis táticos e estratégicos das organizações.

Segundo Fayyad, Piatetsky-Shapiro e Smith (1996, apud REZENDE et al., 2003) a definição de MD é "o processo de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis embutidos nos dados". Para melhor compreensão, é interessante olhar para os componentes da definição individualmente.

- Dados são um "Conjunto de fatos ou casos em um repositório de dados" (REZENDE et al., 2003).
- Padrões são abstrações extraídas deste conjunto de dados, dentro de um determinado contexto.
- Conhecimento é "definido em termos dependentes do domínio que estão relacionados" (REZENDE et al., 2003)
- Válido refere-se à confiabilidade do que está sendo extraído.
- Útil diz respeito a sua importância no contexto.
- Novo, pois o conhecimento extraído é uma nova informação em relação aos dados contidos em determinada base.

A partir da compreensão da definição e dos significados de seus itens, é interessante entender o funcionamento do processo de MD que, segundo Cabena et al. (1997), é composto por 5 etapas, sendo elas: Definição de Objetivos, Preparação e Transformação de Dados, Mineração de Dados, Análise dos Resultados e Assimilação do Conhecimento.

2.2.1 Processo de Mineração de Dados

Quando falamos em MD, é comum pensar que se resume apenas a minerar dados. Entretanto, isto não corresponde com a realidade. Autores divergem sobre as características e quantidade de etapas que existem no processo de MD. Segundo Cabena et al. (1997), o processo é dividido em 5 etapas, cada uma com uma função bem definida para que o resultado seja o melhor.

A Definição dos Objetivos de Negócio é a fase onde é definido o escopo que irá ocorrer a MD. Essa fase pode parecer um tanto intuitiva e, por isso, há uma tendência a não ser cumprida. Fazer mineração sem especificar objetivos pode não trazer os resultados esperados.

Já a Preparação e Transformação dos Dados é a fase mais trabalhosa e, portanto, a mais demorada. Definido o escopo do que será minerado, deve-se pensar de onde virão estas informações. A preparação envolve a seleção e o pré-processamento dos dados. A seleção trata da origem dos dados, fazendo um mapeamento das diversas origens dela, formando um subconjunto de dados que então, serão minerados. Já o pré-processamento julga a qualidade destes dados, decidindo se o que foi selecionado é interessante, ou não, ser minerado. Após isso, os dados podem não estar na formatação necessária pelos algoritmos de mineração.

Para ajustar isso, eles passam por um processo de transformação, onde são ajustados para se encaixarem perfeitamente nas necessidades das ferramentas de mineração. Após isso, estarão prontos para serem minerados.

A Mineração de Dados –se os processos anteriores forem bem executados, os dados forem objetivos, úteis e bem modelados – tende a ser uma das mais rápidas, apesar de ser o coração do processo.

Na Análise de Resultados, a saída gerada pela fase de Mineração de Dados é avaliada e interpretada. Desta forma, é importante que a visualização destes resultados seja feita da forma mais simples, fácil e intuitiva possível. Assim, a pessoa que for fazer esta análise irá extrair o maior nível de conhecimento possível.

Por último, a Assimilação do Conhecimento diz respeito ao que foi extraído na fase anterior. É necessário entender de que forma isso pode ser aplicado no caso que foi utilizado. Assim, podem-se fazer melhorias, ou detectar algo que estava oculto outrora.

Esse é apenas um processo de como deve ser feita a MD como um todo. Para a extração de padrões (Fase de Mineração de Dados), há diferentes técnicas e algoritmos de mineração que se utilizam deste processo para obterem sucesso.

2.2.2 Algoritmo Apriori de Padrões Sequenciais

O algoritmo Apriori foi o escolhido para o desenvolvimento deste estudo. Ele foi criado no centro de pesquisa da International Business Machines (IBM daqui por diante), dentro do projeto QUEST, pelos pesquisadores Agrawal e Srikant (1995). Seu objetivo é encontrar os *itemsets* mais frequentes dentro de uma base de dados, com suporte mínimo intitulado *S*.

$$
\begin{aligned}
&\textit{APRIORI}(T,S)\\
&\quad L_1 \leftarrow \{\ \text{Litemset-1 que ocorre mais que } S \text{ vezes}\ \}\\
&\quad k \leftarrow 2\\
&\quad \textit{Enquanto}\ L_{k-1} \neq \varnothing\\
&\qquad C_k \leftarrow \textit{GeraçãoDeCandidatos}(L_{k-1})\\
&\qquad \textit{Para transações}\ t \in T\\
&\qquad\quad C_t \leftarrow \text{Subset}(C_k, t)\\
&\qquad\quad \textit{Para candidatos}\ c \in C_t\\
&\qquad\qquad \text{count}[c] \leftarrow \text{count}[c] + 1\\
&\qquad L_k \leftarrow \{c \in C_k |\ \text{count}[c] \geq S\}\\
&\qquad k \leftarrow k + 1\\
&\textit{return} \bigcup_k L_k
\end{aligned}
$$

Figura 2 – Algoritmo Apriori em Pseudocódigo - Adaptado de Agrawal e Srikant, (1995).

Como é possível ver na figura 2, o Apriori é executado em uma base de dados *T*, onde as sequências serão avaliadas de acordo com o suporte *S*. O suporte diz respeito à quantidade de ocorrências da transação na base de dados. Se o suporte estipulado for de 50% em uma base de 10 transações, o item deverá ocorrer em pelo menos 5 transações. *K* é o controle do número da iteração, também indicando qual é o número de itens que os conjuntos são formados naquela iteração. *K* inicia em 2, já que o primeiro conjunto será formado fora das iterações. Os conjuntos L_k ou *Litemsets-k*, são conjuntos de transações formadas por *k* itens que respeitam o devido suporte. Já os conjuntos C_k são conjuntos candidatos formados por *k* itens, formados pela combinação dos itens do conjunto L_{k-1}.

A seguir são citados os trabalhos relacionados que fazem uso do Apriori.

2.2.3 Trabalhos Acadêmicos Relacionados

Existem diversos meios em que o algoritmo Apriori pode ser útil. Ele é bastante utilizado por supermercados e lojas, buscando entender quais itens são mais comprados por determinados clientes e qual a ordem disso. Além disso, ele também é utilizado por lojas *online* para definir o comportamento dos clientes, tanto no ato na compra, quanto na observação dos produtos. Pelo fato de ser um algoritmo de mineração de padrões sequenciais, pode ser utilizado em qualquer situação onde haja banco de dados registrando

ocorrências de um evento e necessidade de descobrir padrões dentro destes registros. Kohari (2009) utiliza o algoritmo para encontrar grupos de produtos que são encomendados em conjunto, dentro de uma base de dados de 21705 transações, de uma organização de comércio de materiais elétricos.

Ramos (2010) desenvolveu um estudo sobre a mineração de padrões sequenciais. Nele, apresenta o processo de descobrimento de padrões, e logo após, um *software* de Mineração de Dados desenvolvido por ele, que implementa o algoritmo Apriori. Apresenta também uma avaliação da ferramenta através de execuções do processo, mostrando as iterações executadas pelo algoritmo para o descobrimento dos padrões.

A próxima seção apresenta a solução que foi desenvolvida.

3 APRESENTAÇÃO DA SOLUÇÃO

Esta seção apresenta o *software* desenvolvido capaz de, em tempo real, identificar quais movimentos predeterminados são executados, a partir dos dados gerados pelo sensor *Kinect*. Após isto, num segundo momento, o *software* irá possibilitar a identificação da(s) sequência(s) mais executada(s) dentro da base de movimentos coletados, além de algumas estatísticas sobre os movimentos capturados.

O monitoramento de clientes pode ser tratado como um assunto que possui diversos níveis de significado. O primeiro nível diz respeito à identificação dos movimentos executados a partir dos dados gerados pelo *Kinect*. Num segundo nível, pode-se ser atribuído significado ao movimento, como por exemplo, 'erguer o braço', pode significar tocar o produto. Pode haver ainda um terceiro nível, onde um agrupamento de itens gera um novo significado, como por exemplo, erguer o braço, que pode significar tocar o produto, junto de um movimento de abaixar o braço, podendo significar que o produto foi colocado num cesto de compras. Este trabalho se preocupa apenas com o primeiro nível, que é a identificação dos movimentos executados através dos dados gerados pelo *Kinect*. Junto disto, para auxiliar na triagem desta base de movimentos, é proposto a implementação do Algoritmo Apriori, além das estatísticas simples.

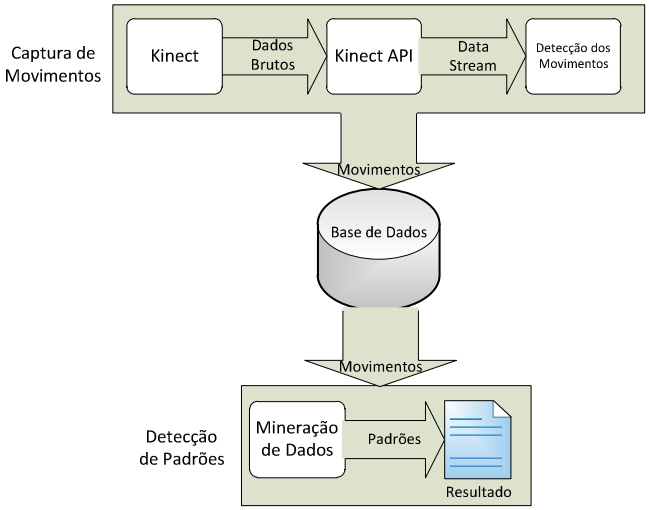

Figura 3 – Etapas Lógicas do Software

A figura 3 mostra o fluxo das informações, dividindo o *software* em duas grandes fases: Captura de Movimentos e Detecção de Padrões. Elas são descritas com diversos itens. Quanto a Captura de Movimentos temos:

- *Kinect*: Esta etapa é relacionada a *hardware*. O *Kinect* é o sensor que fará o monitoramento, coletando os sinais de vídeo e infravermelho do ambiente e traduzindo-os em Dados Brutos.
- Dados Brutos: É a tradução dos sinais de vídeo e de infravermelho em dados quantificáveis. Eles são brutos, pois ainda não possuem estrutura nenhuma e não podem ser acessados diretamente via *software*. Os dados brutos que o *Kinect* gera não são acessíveis por *software*. Apenas a API se comunica com o *Kinect*, para que possa interpretar estas informações, gerando assim informações estruturadas.
- *Kinect* API: A API é responsável por receber os dados brutos gerados pelo *Kinect* e coloca-los

dentro de uma estrutura conhecida e previamente definida. Ou seja, ela é a responsável por tornar os dados brutos em *DataStreams.*

- *DataStream*: São as informações quantificáveis e estruturadas, coletadas do ambiente. Existem vários tipos de *Streams* que a API entrega, mas neste trabalho é utilizado o *SkeletonStream*.
- Detecção de Movimentos: Essa fase diz respeito à interpretação das informações recebidas através do *SkeletonStream.* Assim, faz-se necessário a predefinição de uma lista de movimentos, para poder interpretar as informações recebidas e detectar quais destes movimentos foram executados. Essa etapa terá como saída os movimentos executados por um determinado cliente.
- Movimentos: É a saída da fase de Detecção de Movimentos. São os movimentos executados por um cliente. É importante ressaltar que todo o movimento é ligado a um cliente. Eles são salvos em uma base de dados, para posterior análise.
- Base de Dados: Local onde são armazenados os clientes e seus movimentos. Ela é a etapa final da fase de Captura de Movimentos e a etapa inicial da fase de Detecção de Padrões.

Já a fase de Detecção de Padrões possui estes itens:

- Base de Dados: Esta é a única etapa presente tanto na fase de Captura de Movimentos quanto na Detecção de Padrões. Ela fornece para a fase de Mineração de Dados todos os movimentos executados por cada cliente, além das informações de início e fim de exposição ao monitoramento.
- Mineração de Dados: Esta fase é basicamente a execução do algoritmo Apriori e das demais sumarizações pertinentes à solução. Para isso, o usuário informa a data de início e a data final do período que ele deseja que seja minerado. Além disso, o usuário determina qual o percentual de suporte que ele deseja para o algoritmo Apriori.
- Padrões: Compreende ao término do processo de Mineração de Dados. Ou seja, corresponde às sequências de movimentos mais executadas e às sumarizações do período.
- Resultados: Documento final, com os padrões devidamente formatados para serem exibidos ao usuário.

Na figura 4 temos o diagrama *Unified Modeling Language* (UML daqui por diante) de atividades da solução. Na imagem, é possível visualizar melhor as responsabilidades de cada fase dentro da ferramenta. O sensor *Kinect* capta os sinais do ambiente que está monitorando, interpreta e transforma isso em dados quantificáveis. Esses, são repassados para a API, que gera uma estrutura para esses dados, chamada de *DataStream.* Eles nada mais são do que classes em C# que fornecem as informações estruturadas do esqueleto do usuário que está sendo monitorado.

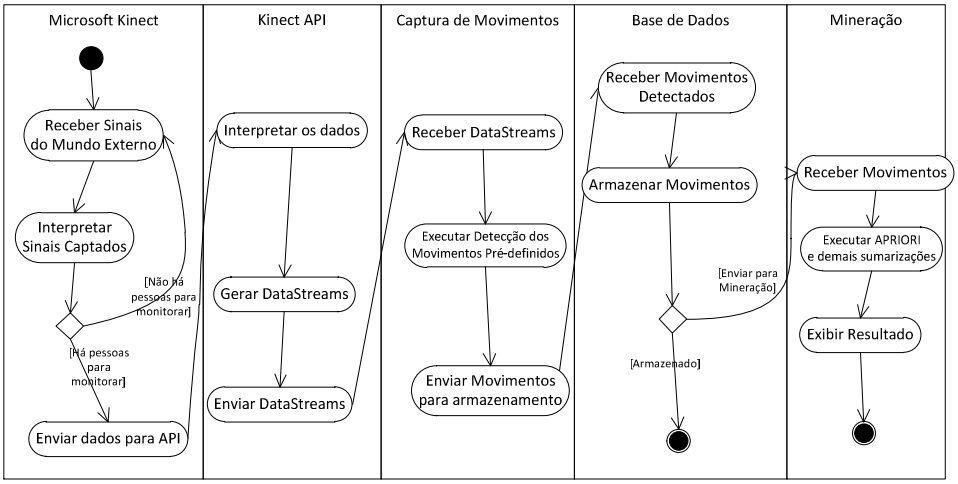

Figura 4 – Diagrama UML de Atividade

A seguir, são apresentadas todas as etapas de funcionamento do *software* e como os itens acima descritos interagem dentro deste fluxo.

3.1 Captura de Movimentos

A seção 3.1.1 explica como o *Kinect*, através da sua API, fornece as informações para serem consumidas através da aplicação. Já na seção 3.1.2, é explicado como é montada a interpretação dos dados

gerados pelo sensor e de que forma são identificados os movimentos predeterminados. Por último, na seção 3.1.3, é explicada a estrutura de banco de dados onde são armazenados os movimentos identificados, para futura recuperação.

3.1.1 Formato do Data Stream Gerado pelo Kinect

Conforme dito anteriormente, o *Kinect,* através de sua API, entrega as informações do monitoramento através do dos *Data Streams* (MICROSOFT, 2012). É possível ler as informações de cor, esqueleto, profundidade e áudio. Para este trabalho será utilizado o *Skeleton Stream,* responsável pelo monitoramento do esqueleto da pessoa exposta ao sensor. Ele envia, *frame* por *frame,* as informações do que capturou do ambiente, como se fosse um vídeo. Ou seja, a cada segundo se tem 30 frames de informação sobre os pontos monitorados, o que possibilita boa precisão para verificar os movimentos desejados.

Figura 5 – Pontos Monitorados pelo sensor (MICROSOFT, 2012)

Esse *Stream* entrega o monitoramento de vinte e três pontos do corpo humano, considerados as principais juntas responsáveis pela maioria dos movimentos. Entre esses pontos destacam-se punhos, cotovelos, ombros, pescoço, joelhos e tornozelos. A figura 5 mostra detalhadamente todas as juntas que o sensor monitora.

Para conseguir ler os dados gerados pelo sensor, são necessárias algumas ações no que concerne a código. É interessante ressaltar que a linguagem utilizada será o C#. Via código, é necessário realizar a ativação do *Skeleton Stream* daquele sensor. Neste trabalho, os demais *streams* (áudio, cor) ficam desabilitados. Além de habilitar o stream desejado, é necessário registrar o método *listener* do *skeleton stream*. O método kinect_*SkeletonFrameReady* é onde são executadas as detecções de movimento e seu armazenamento. Entretanto, ele é um método passivo. Ele somente é executado quando o sensor obtiver a captura dos movimentos de um usuário.

Feito isso, é possível acessar as informações do esqueleto através do método *kinect_SkeletonFrameReady,* que é previamente registrado. Dentro dessa classe deve ser aberto o *frame* que o *Kinect* envia e copiado os dados para um objeto criado do tipo *Skeleton.* A classe *Skeleton* possui uma propriedade chamada *Joints,* que é a coleção de todas as juntas monitoradas do usuário. Para acessar a junta desejada deve ser utilizada o conjunto de enumeração *JointType.* Por exemplo, para acessar as informações da junta, deve ser utilizado *JointType.Head,* deixando a linha de comando da seguinte forma: *Skeleton.Joints[JointType.Head]*. Dentro desse item, há as coordenadas da junta. O sensor fornece as informações das juntas através de coordenadas dos eixos X, Y e Z. Os eixos X e Y são coordenadas cartesianas, ou seja, X representa a posição horizontal e Y a posição vertical. Desta forma, já é possível ter a localização 2D do ponto. Já o eixo Z fornece a distância do ponto em relação ao sensor, ou seja, fornece a profundidade do ponto. A figura 6 mostra um exemplo de acesso às coordenadas X, Y e Z da mão direita, dentro de um frame de dados.

Figura 6 – Exemplo de Acesso as Coordenadas X, Y e Z no Código

Além disso, é utilizado a propriedade *TrackingID* da classe *Skeleton.* Ela possui um código único do esqueleto que é monitorado, gerado de forma automática pela API. Essa propriedade torna-se importante para o armazenamento dos movimentos. Para garantir que seja um *ID* único, quando for salvo na base de dados, este *ID* será concatenado com a data, hora, minuto e segundo em que começa a exposição ao sensor.

3.1.2 Interpretação dos Dados para Identificação dos Movimentos

Conforme foi descrito no item 3.1.1, o *Kinect* fornece coordenadas cartesianas (x,y,z) sobre os 23 pontos monitorados do esqueleto, a 30 frames por segundo. Ou seja, a cada segundo, a API fornece 30 fotos dos pontos, com as coordenadas respectivas. Em cima destas informações, é necessário fazer um modo com que um movimento pudesse ser detectado, evitando ser duplicado ou confundindo com outro movimento.

Para entender melhor como o sensor se comporta, foram executados os movimentos predeterminados perante o *Kinect* para que pudessem ser observados como os valores das coordenadas referentes aos pontos que interessavam ao movimento se comportam. Os valores possuem 8 casas decimais, ou seja, uma precisão bastante grande. Entretanto, dependendo das condições de iluminação, posicionamento do sensor ou obstáculos entre o sensor e um dos pontos, esta precisão fica um pouco comprometida, e o sensor acaba estimando o valor da coordenada de acordo com as posições anteriores.

A melhor maneira encontrada para identificar um movimento sem duplicidade ou ambiguidade foi dividir o movimento em fases. Assim, conforme as fases progridem, chega-se mais perto de ser considerado como executado, o movimento. Desta forma, é possível dizer quando ele foi executado, bem como quando o movimento foi desfeito, o que implica na regressão das fases. Isto é importante para evitar a duplicidade. Quando um determinado movimento é executado, uma *flag* booleana é passada para o *status* verdadeiro. Se nos frames seguintes, o mesmo cenário se mantiver, mas a *flag* estiver com o valor verdadeiro ainda, não será considerado um novo movimento, mas apenas a manutenção do mesmo. Conforme as fases dele forem sendo desfeitas, a *flag* será passada para falso novamente e nesse momento, poderá ser considerado um novo movimento.

Em linhas gerais, todos os movimentos são divididos por fases. Mas para melhor compreensão do processo de identificação do movimento, será exemplificado o passo a passo da identificação do movimento de Erguer o Braço Direito, que é um dos movimentos predefinidos propostos a serem identificados pelo *software* desenvolvido neste trabalho. Para este movimento, foram utilizados os pontos do ombro direito (na API *ShoulderRight*), cotovelo direito (na API *ElbowRight*) e mão direita (na API *HandRight*). Este movimento foi dividido em 3 fases. É considerada alcançada a primeira fase, quando o valor da coordenada do eixo Y da mão direita é superior ao valor da coordenada do eixo Y do ombro direito. Ou seja, a mão direita passou da altura do ombro. Neste momento, uma *flag* é configurada com o valor verdadeiro.

A próxima fase tem a mesma ideia, mas os pontos de referência são o ombro direito e o cotovelo direito. Quando a coordenada do eixo Y do cotovelo direito é maior que a coordenada do eixo Y do ombro direito, é considerada alcançada a segunda fase e uma nova *flag* é passada para verdadeiro.

A última fase envolve os pontos da mão direita e do ombro direito. Entretanto, desta vez, é avaliada a coordenada do eixo Z, que diz respeito à proximidade do ponto em relação ao sensor. É verificada se a coordenada do eixo Z da mão direita é maior que a do ombro direito, ou seja, se a mão direita está mais próxima do sensor do que o ombro. Isto é verificado para certificar que a mão está projetada para frente, como se estivesse tentando alcançar alguma coisa. É possível notar na figura 7, que a representação do ponto branco da mão direita está maior que o do ombro direito, como se estivesse mais próximo do observador. Para isso, é estipulado um valor mínimo que esta diferença deve estar, para que assim a o braço esteja bem projetado para frente.

Na figura 7 é possível observar, fase a fase o movimento de erguer o braço direito. É importante ressaltar que o *Kinect* é preparado para que o usuário fique de frente para o sensor. Logo, os movimentos digitalizados e impressos na tela funcionam como se fossem um espelho.

Na fase 1, pode-se perceber que o valor da coordenada do eixo Y para a mão direita é -0,02, ou seja, é maior que a do ombro direito, -0,05. Neste momento, então, é atribuído o valor verdadeiro para a *flag* da fase 1. Já na fase dois é esperado que a coordenada do eixo Y do cotovelo direito passe da coordenada do ombro direito. Na imagem da fase 2 é possível notar isso, pois o cotovelo direito possui o valor de 0,142, enquanto o ombro está com -0,0003. Enquanto isso, a mão direita possui um valor ainda maior para o eixo Y. Neste momento a *flag* da fase 2 é configurada como verdadeira. Na fase 3 por sua vez, é considerado o eixo Z. Os pontos considerados são o Ombro direito e a mão direita.

Figura 7 – Identificação do Movimento de Erguer o Braço direito, fase a fase

Como foi dito anteriormente, é necessário configurar um valor mínimo para que possa ser considerado que o ponto da mão está projetado para frente. Este valor foi fixado em 0,4. Na imagem, vemos que a coordenada da mão é 1,3 enquanto a do ombro é 1,8. Ou seja, há uma diferença de 0,5 entre as coordenadas, onde a mão está mais próxima do sensor que o ombro. Desta forma, a fase 3 é completada, e assim o movimento é considerado executado.

Os outros movimentos seguem a mesma lógica de fases, entretanto utilizando pontos pertinentes a estes movimentos e verificações específicas particulares àquele tipo de ação. Todos os processos de identificação dos movimentos foram definidos neste trabalho após observações feitas de como os movimentos são executados e como o *Kinect* reage a eles. Além disso, é importante ressaltar que novos movimentos podem ser definidos futuramente, para outras aplicações.

Uma vez identificado o movimento, ele é salvo em um banco de dados, vinculado ao *ID* do usuário que o executou.

3.1.3 Geração da Base de Movimentos

Os movimentos identificados são armazenados em uma base de dados. A estrutura de tabelas necessárias para este armazenamento é bastante simples e está ilustrada na figura 8. A opção de utilizar um *software* de banco de dados para esta tarefa se dá devido às facilidades que ele oferece para armazenar e recuperar estas informações. É necessário armazenar em local seguro os movimentos para que depois, possa ser recuperado de forma consistente e ágil, para executar o Apriori e as demais sumarizações propostas.

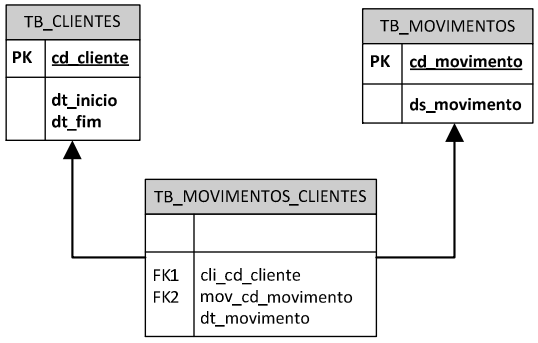

Figura 8 – Diagrama ER da Base de Dados

Na tabela Tb_Clientes, são registrados os clientes monitorados, com um código único para cada um. Este código consiste na concatenação do *TrackingID* gerado pelo *Kinect*, com o a data e hora em que o

usuário entrou no monitoramento do *Kinect*. Ou seja, se o *TrackingID* for 30, e a data e hora for 01/01/2013 15:01:01, o *id* registrado na tabela seria 302013010115010101. É feito desta forma para que não haja duplicidade de *ids* na tabela. Além disso, armazena a data (dia, mês e ano) e hora (hora, minuto e segundo) de início e fim de exposição ao sensor.

Na tabela Tb_Movimentos, estão previamente cadastrados os 10 movimentos predefinidos a serem monitorados pelo *software*. Na tabela 1 é possível visualizar a relação entre o código e a descrição do movimento. Eles são associados a códigos para facilitar no registro do movimento, bem como na hora de executar o Apriori. Estes movimentos foram definidos para testar a proposta. Como já foi dito anteriormente, é possível definir novos movimentos, conforme necessidade.

Tabela 1 – De-para de Códigos e Descrição dos movimentos

Código	Descrição
1	Erguer Ambos os Braços
2	Erguer Braço Direito
3	Erguer Braço Esquerdo
4	Colocar Ambas as Mãos na Cabeça
5	Colocar a Mão Direita na Cabeça
6	Colocar a Mão Esquerda na Cabeça
7	Esticar Ambos os Braços
8	Esticar o Braço Direito
9	Esticar o Braço Esquerdo
10	Cruzar os Braços

Já na tabela Tb_Movimentos_Clientes, são armazenados todos os movimentos executados pelo cliente monitorado. Ela é fruto de uma relação n para n das tabelas de clientes e movimentos. Nela é registrado o *id* do cliente que está executando determinado movimento, o *id* do movimento que foi identificado, e a data de execução do movimento. O campo de data é primordial, pois ele dará a sequência cronológica de execução dos movimentos, o que é fundamental para o algoritmo Apriori.

3.2 Detecção de Padrões

Esta seção irá falar sobre a fase de detecção de padrões. Na seção 3.2.1, será descrito como foi feita a implementação do algoritmo Apriori. Já na seção 3.2.2 serão descritas as sumarizações que foram desenvolvidas e de que forma.

3.2.1 Detalhamento e Implementação do Apriori

O Apriori é executado em uma base de dados, onde as são avaliadas as sequências de acordo com um valor de suporte. Cada iteração do algoritmo, possui uma variável que armazena o número da iteração. Ele também indica qual é o número de itens que os conjuntos são formados naquela iteração. O algoritmo só irá parar quando o conjunto de candidatos for nulo, ou seja, mais nenhum conjunto pode ser criado, então ele para. Para possibilitar a implementação, fez-se necessário dividir as funcionalidades, e fazer métodos C# específicos para as funcionalidades do Apriori. Primeiramente será visto a definição e detalhamento do Apriori. Logo após, serão apresentados os métodos e a divisão feita pela implementação do Apriori neste trabalho.

Segundo De Amo (2004), as diversas iterações executam primordialmente três etapas. São elas: a Geração de Conjuntos Candidatos, Poda de Candidatos e Cálculo do Suporte e, por último, a Validação do Suporte, as quais serão explicadas a seguir.

- Geração de Candidatos: é gerado um conjunto de candidatos C_k de *itemsets*, que não são necessariamente frequentes na base de dados. Ele usualmente é gerado a partir do conjunto L_{k-1}, exceto quando $K=1$. Nesse caso, ele é gerado a partir da base de dados. Após, é feita a poda dos candidatos. A geração destes candidatos é feita conforme o algoritmo em pseudocódigo ilustrado na figura 9.

Figura 9 – Geração dos Candidatos - Adaptado de De Vasconcelos e De Carvalho (2004).

- Poda de Candidatos: Para entender como funciona a poda dos candidatos é necessário, antes, entender a propriedade do algoritmo Apriori, chamada de Antimonotonia da Relação ⊆. Ela diz: "Sejam I e J dois *itemsets* tais que I ⊆ J. Se J é frequente então I também é frequente" (DE AMO, 2004). Na prática, isso quer dizer que um *itemset* de tamanho *K* de C_k só será considerado frequente se seus *subitemsets* de tamanho *K-1* fizerem parte do conjunto L_{k-1}. A partir desta poda, é gerado um conjunto C'_k . A execução da poda dos candidatos é ilustrada na figura 10.

Figura 10 – Poda dos Candidatos - Adaptado de De Vasconcelos e De Carvalho (2004).

- Cálculo de Validação do Suporte: Nessa fase é calculado o suporte para cada *itemset* armazenado em C'_k. Assim, são avaliadas quantas vezes cada *itemset* ocorre na base de dados. Após isso, este número é comparado com o valor do suporte *S* definido pelo usuário. Caso esteja abaixo do suporte definido, o *itemset* é descartado. Depois de descartados todos os *itemsets* com ocorrências menores que o suporte definido, é gerado o conjunto L_k, ou *Litemset-*$_k$, que é o conjunto de sequências frequentes de tamanho *K*. O cálculo do suporte é a única fase do algoritmo em que é necessário acessar a base de dados, uma vez que as outras fases utilizam apenas as estrutura de conjuntos candidatos frequentes, da iteração corrente e da iteração anterior.

São executadas quantas iterações forem necessárias e, em cada uma delas, implementado a lógica das fases descritas. O algoritmo encontrará seu ponto de parada quando L_{k-1} = nulo, ou seja, quando o conjunto de *itemsets* frequentes anterior à iteração corrente for nulo. Isso significa que não há mais avaliações a serem feitas, uma vez que não haverá mais *itemsets* que se enquadrem nas regras do algoritmo.

Neste trabalho, as funcionalidades do Apriori foram divididas por métodos C#. Os métodos criados para implementar o algoritmo foram os seguintes:

- PopulaBancoDeTransacoes: Como a implementação feita neste trabalho possibilita a restrição da base através de datas de início e fim, é necessário filtrar os dados da base de movimentos. Este método é o primeiro de todos a ser executado. Caso não retorne nenhuma transação no período executado, o usuário será notificado e o Apriori não será executado.
- Apriori: Este método é o algoritmo em si. Ele simplesmente faz o encadeamento dos demais métodos, para que o objetivo possa ser cumprido. Além disso, é neste método que ficam controles como iteração corrente e o dispositivo de parada, ou seja, quando não houver mais conjuntos candidatos. Neste momento, o algoritmo termina, e pode imprimir o conjunto resultado na tela, para visualização do usuário.
- GeracaoConjuntoCandidato: Este método, é responsável por duas tarefas. A primeira é mais importante, pois dá o nome do método. Aqui são criados os novos conjuntos candidatos. Ou seja, esse método é um dos primeiros a serem executados dentro do algoritmo. Conforme a definição do Apriori, esee método consulta a base de transações apenas na primeira iteração para achar o conjunto de candidatos. Após a primeira tarefa, os conjuntos são feitos a partir do conjunto de itens frequentes da iteração anterior. Além disto, o método possui uma segunda tarefa. Nela é

calculado o valor do suporte de cada *itemset*. Não é efetuado nenhum tipo de corte, nem de poda. Apenas é calculado o valor do suporte, para posterior uso pelo método de Efetuar o corte através do valor de suporte.

- EfetuaPoda: Este método é responsável pela poda de candidatos. Ele é a aplicação da regra de Antimonotonia, descrita acima na definição do algoritmo. Após sua execução, o conjunto ainda é considerado candidato, pois ainda não foi executada o corte através do valor do suporte. Isto é feito no próximo método.
- EfetuaCorteSuporte: É o último método a ser executado. Depois dele, um conjunto deixa de ser considerado candidato, para ser considerado um conjunto de itens frequentes. Ele compara o valor de suporte de cada *itemset* com o valor de suporte estipulado pelo usuário. Caso o valor estipulado pelo usuário seja menor que o valor de suporte do *itemset*, ele é cortado do conjunto.

3.2.2 Estatísticas Simples

Além do Apriori, foram feitas algumas estatísticas simples, para que seja possível entender melhor a base de movimentos capturados. É interessante frisar que esta etapa é executada filtrando os dados da base através de parâmetros de data inicial e data final.

As estatísticas são:

- Total de Movimentos: O número total de movimentos identificados pela ferramenta dentro do período selecionado pelo usuário.
- Média de Movimentos por Dia: Média de movimentos identificados por dia pela ferramenta dentro do período selecionado.
- Total de Clientes: Total de clientes que foram monitorados pelo sensor dentro do período selecionado. Aqui entram clientes que foram monitorados, mas que por ventura não tiveram nenhum movimento identificado.
- Média de Clientes por Dia: Média de clientes que foram monitorados pelo sensor dentro do período selecionado pelo usuário.
- Tempo Médio de Exposição: Tempo médio que os clientes ficaram na frente do *Kinect* sendo monitorados. Este item é relevante, pois pensando em uso real do *software*, seria interessante saber qual o tempo médio que um cliente fica em frente a uma prateleira.
- Movimentos Mais Executados: Quais os movimentos mais executados dentro do período.

3.3 Desenvolvimento do Software

Esta seção descreve a parte técnica da solução desenvolvida. Na seção 4.3.1 são descritas as tecnologias utilizadas para o desenvolvimento deste *software*. Já na seção 4.3.2 são descritas as principais classes desenvolvidas neste *software* e quais suas principais funções. Por último, na seção 4.3.3, é feito um comparativo entre o *Kinect for Xbox* e o *Kinect for Windows*.

3.3.1 Tecnologias Utilizadas

O *software* foi desenvolvido todo em C#, linguagem esta que é orientada a objetos. Além disso, foi utilizado o *framework Windows Presentation Foundation* (WPF daqui por diante) para a parte gráfica. Apesar de a API do *Kinect* possibilitar o desenvolvimento de aplicações com C++ e *Visual Basic*, C# foi escolhido devido à documentação para esta linguagem ser bem mais vasta que para as outras linguagens.

Para armazenamento das informações, foi utilizado uma base de dados *Oracle 11G*, onde foram criadas as estruturas de tabelas. Como forma de conexão foi utilizado o ODP.NET (Oracle Data Provider for .NET), que é um driver desenvolvido pela *Oracle* para acesso a base de dados *Oracle* a partir de plataformas .NET.

3.3.2 Principais Classes do Software

Devido a estrutura do *framework* WPF, cada tela possui um arquivo na forma *Extensible Application Markup Language* (XAML, daqui por diante) para marcação do *layout*, além de um arquivo de extensão .cs, que contém a classe da tela, onde será codificado o que for necessário. Estas classes possuem as funções mais básicas de programação, no que é pertinente à tela. Mas há duas classes que foram criadas especificamente para este *software*.

Na classe MonitoredSkeleton ocorre todo o processo de identificação dos movimentos executados. Ela possui como propriedades as *flags* das fases de todos os movimentos que são propostos a serem

identificados. Além disso, possui um método para cada movimento a ser identificado. Na tela é chamado o método desejado, passando o objeto que possui as informações do usuário monitorado. Dentro do método, estas informações são lidas e tratadas, para saber se algum movimento ou fase foi executada.

A classe MovementMining é responsável pela parte de identificação dos padrões dentro da base de movimentos. Nela estão situados os métodos do Apriori, que foram descritos na seção 4.2.1 deste artigo. Além disto, ela possui também um método onde são executadas todas as sumarizações descritas na seção 4.2.2 deste trabalho.

Por último, a classe DBOperations é responsável por todo e qualquer acesso à base de dados. A classe MonitoredSkeleton utiliza a DBOperations para fazer o registro dos movimentos identificados na base de dados. Já a MovementMining utiliza ela para recuperar os movimentos salvos, para que seja possível executar o Apriori e extrair as estatísticas.

4 VALIDAÇÃO E TESTE

Esta seção aborda como foi feita a validação do processo de identificação de movimentos, bem como de qual a forma foi testada a implementação do algoritmo Apriori. Primeiramente, na seção 4.1 é apresentado o plano de validação da parte de identificação dos movimentos e depois, na seção 4.2 são descritos os resultados desta validação. Já na seção 4.3, é descrito o plano de teste para a implementação do algoritmo Apriori e na seção 4.4, os resultados obtidos.

4.1 Plano de Validação para a Captura de Movimentos

Conforme já foi dito, a captura de movimentos consiste em coletar as informações geradas pelo sensor *Kinect* e identificar quando um movimento predefinido foi executado. Entretanto, para validar a proposta deste trabalho, de identificação e captura destes movimentos, é necessário aferir o percentual de acerto da identificação destes movimentos.

A validação foi feita através de testes assistidos. Foram colocadas pessoas na frente do *Kinect* e solicitados ao usuário a execução de movimentos. Neste momento, é anotado o movimento executado pela pessoa, ao mesmo tempo em que o *software* identifica o movimento em questão, caso ele seja um dos movimentos predefinidos. Não é identificado nada, caso não seja um movimento predefinido. Após isso, são cruzadas as informações das anotações com o que o *software* registrou na base de dados, procurando verificar se algum movimento deixou de ser identificado, foi duplicado, ou foi identificado como outro movimento (falso-positivo). Ao final disso, é extraído um percentual de acerto dos movimentos. Algumas metas foram estipuladas antes de executar a validação:

- Mínimo de participantes: 5 pessoas
- Tota de movimentos: 150

Além disso, foram levantadas observações gerais quanto ao comportamento do *software* em relação aos movimentos executados, tentando entender o por que de um movimento ter sido detectado errado ou não ter sido detectado.

4.2 Resultados da Validação para a Captura de Movimentos

Conforme foi descrito no plano de validação da captura de movimentos, foram utilizadas 5 pessoas diferentes para fazer o teste. As 5 pessoas foram, uma por vez, solicitadas a ficar em frente ao *Kinect*. Em seguida, foi solicitado às pessoas que simulassem movimentos que fazem corriqueiramente em frente a uma prateleira de itens em alguma loja. No total, cada pessoa executou 30 movimentos, totalizando 150 movimentos.

Tabela 2 – Resultado dos Totais da Validação da Captura de Movimentos

Descrição	Valor
Total de Movimentos	150
Total de Acertos	122
Total de Erros	28
Percentual de Acertos	81,3%

Percentual de Erros	18,7%

De acordo coma a tabela 2, do total de 150 movimentos, o *software* foi capaz de identificar corretamente 122 movimentos, totalizando assim um percentual de acerto de 81,3%. 28 movimentos foram identificados erroneamente, ou não foram identificados, deixando assim um percentual de 18,7% de erros.

Além disso, se for verificado os percentuais por cada pessoa, é possível observar que o *software* se manteve em uma média interessante em todas as pessoas, conforma mostra a tabela 3.

Tabela 3 – Resultado Por Pessoa da Validação da Captura de Movimentos

Pessoa	Percentual Acerto	Percentual Erro
Pessoa 1	86,7%	13,3%
Pessoa 2	70%	30%
Pessoa 3	86,7%	13,3%
Pessoa 4	76,6%	23,4%
Pessoa 5	90%	10%

Das cinco pessoas que participaram do teste, apenas duas ficaram abaixo dos 80% de acerto. O percentual de acerto mais baixo foi aferido na pessoa 2, com 70% de acertos e o maior, na pessoa 5, com 90%.

Esta validação teve seu foco nos 28 movimentos detectados de forma incorreta, buscando entender por que aconteceram estes erros. Conforme podemos ver na tabela 4, esta foi a distribuição dos erros.

Tabela 4 – Erros de Captura por Movimento

Movimento	Erros
Mão Direita na Cabeça	9
Mão Esquerda Na Cabeça	6
Ambas as Mãos na Cabeça	2
Cruzar os Braços	2
Ergue Braço Direito	2
Ergue Braço Esquerdo	2
Estica Ambos os Braços	2
Estica Braço Direito	2
Estica Braço Esquerdo	1

A maior quantidade de erros ficou concentrada nos movimentos de Mão Direita e Mão Esquerda na Cabeça. Fazendo uma análise dos movimentos identificados com erro, foi possível levantar os seguintes pontos:

- Nos movimentos de Erguer o Braço Direito, Erguer o Braço Esquerdo, Erguer Ambos os braços, Estica Braço Direito, Esticar Braço Esquerdo e Esticar Ambos os braços é levado em consideração a diferença de profundidade entre o ponto da mão e do ombro, para ver se o braço está projetado para frente. Na maioria das vezes em que esse movimento não foi detectado a pessoa não estava projetando o braço o suficiente para satisfazer o limiar do *software*. Ou seja, seria interessante reavaliar o limiar que foi configurado no *software*. Nos primeiros erros, observou-se que o *Kinect* inferiu valores devido à má iluminação do local em que ocorreram os testes. Após melhorar a iluminação local, a detecção dos pontos pelo *Kinect* ficou mais efetiva, tornando o *software* mais preciso.
- As duas vezes em que o movimento de Cruzar os Braços foi marcado com erro, foi possível perceber que a velocidade com que o movimento é executado interfere na sua detecção, bem como a forma como a pessoa cruza os braços. Se a pessoa fizer o movimento muito rápido, o

software não consegue capturar o momento. Além disso, por haver uma sobreposição de pontos, o *Kinect* infere algumas coordenadas. A inferência do *Kinect* pode distorcer muito a real coordenada do ponto que está obstruído, confundindo a identificação do movimento.

- Os dois movimentos com mais erros foram os de Mão Direita na Cabeça e Mão Esquerda na Cabeça. Esse problema ocorreu, em quase todas as vezes, por confusão com o movimento de Cruzar os Braços. Como já foi dito, quando há uma sobreposição de pontos, o ponto que fica fora do alcance da monitoria do *Kinect*, tem seus valores inferidos pelo sensor, para que a leitura pelo *software* possa continuar. Entretanto, na maioria das vezes, esta inferência acaba por distorcer muito o que realmente a pessoa está fazendo. Nas 9 vezes em que o movimento de Mão Direita na Cabeça foi identificado como errado, a pessoa estava, na realidade, cruzando os braços. Isso aconteceu pois ela deixou a mão direita atrás do braço esquerdo. Neste momento, o *Kinect* não sabe mais onde está a mão direita e tenta inferir os valores dela. Foi observado que os valores inferidos pelo *Kinect* deixavam o ponto da mão direita próximo à cabeça do usuário, fazendo com que o *software* entendesse que o movimento de mão direita na cabeça foi executado. A mesma coisa ocorreu quando a mão esquerda ficava encoberta pelo braço direito. Não foi encontrado na API do *Kinect* como diminuir ou detectar estes eventos.

4.3 Detecção de Padrões Sequenciais Com Base Simulada

Para testar a implementação do algoritmo Apriori, foi utilizado como base o artigo Ferramenta para Mineração de Sequências de Eventos Temporais (RAMOS, 2010). Nele, foi feito um estudo sobre o algoritmo Apriori e suas características. Nele também consta um exemplo de base de transações e o resultado da execução do algoritmo contra esta base.

Para testar a implementação feita neste trabalho, foi utilizada a mesma base do artigo de RAMOS e foi executado o Apriori desenvolvido neste artigo. esperáva-se que a saída, ou seja, a(s) sequência(s) mais frequente(s), fossem iguais às descritas no artigo.

No trabalho de RAMOS (2010), o Apriori foi executado contra uma base de transações comerciais, onde cada linha representa os itens comprados por um cliente em um estabelecimento comercial. Também era descrito os significados de cada id. Neste trabalho isto não será levado em conta, pois o intuito desta comparação é comprovar a funcionalidade do algoritmo Apriori. A base era a seguinte: {(1,2,3,4,5,6),(1,3,7),(1,2,4),(3,4),(5),(6,2,1),(3,4,1,2),(7,6,2),(6,2,1,3,4),(5,3,8)}

No trabalho de RAMOS (2010), o suporte mínimo utilizado foi de 2. Com isso, o Apriori resultou nas seguintes sequências como mais frequentes: {(1,2,4),(1,3,4),(2,3,4)}

Foi então simulada a mesma situação no ambiente desenvolvido neste trabalho. Foi cadastrada a mesma base de transações. Além disso, foi configurado o valor mínimo de suporte para 2 e executado o Apriori. No final da execução, foram encontradas as mesmas sequências descritas no trabalho de RAMOS (2010).

A seguir é relatado o resultado da detecção de padrões em uma base gerado por um ambiente real.

4.4 Detecção de Padrões Sequenciais em Ambiente Real

Além da base simulada para comparação, foi simulado um ambiente de lojas com prateleiras. Foi montado um ambiente controlado, com diversos itens em um armário, simulando uma prateleira de uma loja. Foi solicitado aos clientes que fizessem interações com os itens que estavam colocados no armário. Com essa base gerada, foram rodados o algoritmo Apriori e as demais estatísticas simples, para que sejam levantados padrões.

O teste coletou movimentos de 20 pessoas diferentes, interagindo com os itens expostos no armário que estava simulando uma prateleira de uma loja. Com o Apriori, foi possível encontrar 2 sequências de 4 movimentos, como as mais frequentes, sendo configurando o valor mínimo de suporte para 3.

1. Erguer Braço Direito , Erguer Braço Esquerdo , Colocar a Mão Direita na Cabeça ,Colocar a Mão Esquerda na Cabeça - Suporte: 3
2. Erguer Braço Direito ,Erguer Braço Esquerdo ,Colocar a Mão Esquerda na Cabeça ,Cruzar os Braços - Suporte: 3

A primeira sequência, diz que foi erguer o braço direito, depois o braço esquerdo, depois mão direita na cabeça seguida da mão esquerda na cabeça. Como exemplo, uma das interpretações possíveis é que o

cliente está comparando produtos, uma vez que a mão na cabeça pode significar que ele está com os produtos que pegou, no movimento de erguer o braço, próximo ao rosto.

Já na sequência 2, pode-se interpretar, por exemplo, que o cliente pegou dois produtos, com os movimentos de erguer braço direito e esquerdo, entretanto, visualizou somente um. Isto pode significar que o primeiro produto já é conhecido do usuário e ele está conhecendo um produto novo. Por último, há o movimento de cruzar os braços, que pode indicar uma dúvida do cliente entre um produto já conhecido e um produto novo.

Além das sequências geradas pelo Apriori, há também as estatísticas simples coletadas da mesma base de testes. O teste foi feito em 2 dias, então as informações da tabela 5 mostram a distribuição da coleta dentro deste período.

Tabela 5 – Estatísticas Simples

Estatística	Valor
Total de Movimentos	144 Movimentos
Total De Clientes	20 clientes
Média de Movimentos por Dia	72 movimentos/dia
Média De Clientes por Dia	10/dia
Tempo Médio de Exposição	2 Minutos

Conforme mostra na tabela 5, foram coletados 144 movimentos de 20 clientes diferentes. Uma informação interessante é o tempo médio em que os clientes ficaram em frente à prateleira. Neste experimento, este tempo médio foi de 2 minutos, o que é um tempo considerável, mostrando que provavelmente os clientes analisaram os itens antes de tomar alguma ação e não simplesmente passaram em frente à prateleira.

Há ainda os movimentos mais executados dentro da base. Na tabela 6 é possível visualizar a distribuição destes movimentos. Como líder de execuções está o movimento de Colocar a Mão Direita na Cabeça, seguido do movimento de Colocar a Mão esquerda na cabeça. Com isto podemos, por exemplo, interpretar que houve bastantes visualizações de itens. As sequências do Apriori nos mostram que estes movimentos estão atrelados aos movimentos de erguer e esticar o braço. Mas nem todos os movimentos estão atrelados a isso, logo, podemos, por exemplo, interpretar que esta diferença significa uma dúvida do cliente, uma vez que o movimento de cruzar os braços está em terceiro colocado nesta lista.

Além disso, os movimentos de Erguer e Esticar os braços estão bem distribuídos, mostrando que os items estavam bem dispersos onde estavam expostos.

Tabela 6 – Movimentos Mais Executados

Movimento	Valor
Colocar a Mão Direita na Cabeça	32
Colocar a Mão Esquerda na Cabeça	24
Cruzar os Braços	20
Erguer Braço Direito	13
Colocar Ambas as Mãos na Cabeça	11
Esticar o Braço Esquerdo	11
Erguer Braço Esquerdo	9
Esticar o Braço Direito	9
Esticar Ambos os Braços	8
Erguer Ambos os braços	7

5 CONCLUSÃO

Em vista de tudo até aqui exposto, pode-se concluir que é possível a identificação de movimentos via *software*, utilizando o sensor 3D *Kinect*. Conforme a validação feita, foi possível atingir um percentual de 81,3% de acerto dos movimentos. Além disso, foi identificada uma série de pontos que podem gerar melhorias no processo de identificação que foi desenvolvido neste trabalho. Além disso, foi possível fazer uma implementação do algoritmo Apriori, para possibilitar a mineração da sequência de movimentos mais executada dentro de uma base de movimentos.

Esta ferramenta desenvolvida pode trazer vantagens para qualquer local que gostaria de entender o padrão de movimentos de pessoas perante a um ponto de referência. O foco maior desta ferramenta são lojas, onde há prateleiras com produtos em que o consumidor escolha o produto, ou até mesmo vitrines. O *software* possibilita uma discussão acerca de quais produtos são alcançados pelo cliente, quando o cliente fica em dúvida e até mesmo quantos cliente interagem com os produtos e quantos passam pela prateleira sem interagir.. Na situação de vitrines, é possível discorrer para onde o cliente apontou, qual produto chamou atenção e também o tempo o cliente fica em frente à vitrine antes de seguir seu rumo, para que seja possível deixa-la mais atraente ao público alvo.

Apesar de ter sido pensado na realidade de lojas, com prateleiras e vitrines, é possível aplicar este *software* em outras situações, dependendo dos movimentos que forem definidos.

5.1 Dificuldades Encontradas

Durante o processo de desenvolvimento do *software* de identificação de movimentos, foi possível levantar alguns pontos referentes ao sensor *Kinect*, que devem ser observados neste trabalho. Primeiramente, será falado sobre os fatores externos ao sensor que podem trazer uma perda de precisão na sua detecção, e logo após, será comparado os dois tipos de sensores disponíveis.

5.1.1 Fatores Externos

O *Kinect* detecta 23 pontos do corpo humano utilizando basicamente 2 câmeras. Logo, alguns fatores do mundo externo são importantes para que o sensor consiga obter seu melhor resultado.

- Iluminação: Por utilizar câmeras, a iluminação do ambiente é vital para uma maior precisão do *Kinect*. Um local mal iluminado pode distorcer muito os valores, ou até mesmo impossibilitar a detecção dos pontos do esqueleto.
- Distância: Se o usuário estiver muito próximo do sensor, o *Kinect* não consegue detectar, com perfeição, os pontos. Maiores detalhes sobre distância serão descritos na seção 6.1.2, quando será feito um comparativo entre o *Kinect for Xbox* e o *Kinect for Windows*
- Altura do Sensor: Foi identificado também que a altura de onde o sensor fica posicionado influencia na detecção. O ideal é manter o sensor na altura dos olhos das pessoas, ou acima disto. Caso fique acima, é necessário ajusar o ângulo vertical do sensor.

5.1.2 Kinect for Xbox x Kinect for Windows

Existe, no mercado, dois tipo de sensores *Kinect*. Um deles, o mais conhecido, é o *Kinect for Xbox*, que é o sensor utilizado na console de jogos da *Microsoft*, o *Xbox*. O outro é o *Kinect for Windows*, sensor específico para ser utilizado em computadores. Foi utilizado neste trabalho o *Kinect for Xbox*, devido ao seu custo muito menor em comparação ao *Kinect for Windows*. Entretanto eles possuem algumas diferenças.

- *Release* de Aplicação: Em uma aplicação que utilize *Kinect*, enquanto for rodada dentro da Integradted Development Enviroment (IDE, daqui por diante) de desenvolvimento, o Visual Studio, não há problemas, ambos os sensores são aceitos. Entretanto, neste momento já há um aviso que caso seja gerada uma *release* do aplicativo, o *Kinect* for Xbox não será considerado um sensor válido, uma vez que ele não foi feito para este fim.
- Distância Mínima: O *Kinect for Windows*, promete uma distância mínima de 0,3 metros para que possa monitorar um usuário. Já o *Kinect para Xbox* pede pelo menos 1,5 metros para que essa detecção funcione bem. Logo, pelo perfil deste trabalho, seria interessante utilizar o *Kinect for Windows*. Mas, como já foi relatado, o custo é muito maior, o que acabou inviabilizando sua utilização.

5.2 Trabalhos Futuros

Este trabalho é apenas uma das possibilidades a serem feitas. Foi idealizado, durante seu desenvolvimento, o monitoramento do rosto do usuário, para que fosse possível detectar emoções, como um sorriso. Entretanto, devido à grande complexidade disso, não foi possível incluir essa ação no trabalho em questão.

Além disso, como já foi dito no início deste artigo, esse trabalho não busca dar sentido nenhum aos movimentos. Como trabalho futuro, também seria interessante atrelar um sentido a cada movimento, para que seja possível entender melhor o comportamento de uma pessoa que foi monitorada pelo *software*.

AGRADECIMENTOS

Agradeço a todas as pessoas que cruzaram meu caminho e de alguma forma, me ensinaram alguma coisa. Pelas ideias, ensinamentos e atalhos dados, meu orientador Stanley. Pelo apoio, incentivo e carinho incondicional minha mãe Madalena, meu padrasto Nilo, meu pai Lélio e minha madrasta Sabina. Por superar meu mau humor com um alto astral radiante, agradeço muito ao amor da minha vida, Tamiris. E por último, as minhas inspirações maiores, que não conseguirão presenciar eu me formar, pois estão olhando para mim lá das estrelas, meu avô Gabriel e minha prima Maíris.

REFERÊNCIAS

AGRAWAL, R.; SRIKANT, R. **Mining Sequential Patterns**. In: Proceedings of the International Conference on Data Engineering, 11º, 1995, Taipei, Taiwan, p. 3-14.

CABENA, P. et al. **Discovering Data Mining**: From Concept to Implementation. Estados Unidos: Ed. Prentice Hall, 1997. p. 63-88

CHANGA, Y.; CHENB, S.; HUANGC, J. **A Kinect-based system for physical rehabilitation**: A pilot study for young adults with motor disabilities. *Research in Developmental Disabilities*, Estados Unidos, v. 32, f. 6, p. 2566–2570, 2011.

CRAWFORD, Stephanie. **How Microsoft Kinect Works**. How Stuff Works, 2010. 5 f. Disponível em: < http://electronics.howstuffworks.com/microsoft-kinect.htm>. Acesso em: 18 mar. 2013.
DE AMO, Sandra. **Técnicas de Mineração de Dados**. In: Jornada de Atualização em

DE VACONCELOS, L.M.R.; DE CARVALHO, C.L. **Aplicação de Regras de Associação para Mineração de Dados na Web**. Brasil, Universidade Federal do Rio Grande do Sul, 2004. p. 11-14

KOHARI, A.M. **Uma Aplicação Do Algoritmo Apriori Em Transações Envolvendo Materiais Elétricos**. Brasil, Universidade Estadual de Maringá, 2009. 30 p.

MENDONÇA, Vinícius Godoy de. **Reconhecimento de Sinais de LIBRAS usando Sensor 3D**. PUCPR, 2012. Disponível em: < http://www2.pucpr.br/reol/semic/trabalho.php?dd0=6986&dd90=07d65d945b>. Acesso em 26 mar. 2013

MICROSOFT. **Kinect for Windows SDK**. MSDN, 2012. Disponível em : < http://msdn.microsoft.com/en-us/library/hh973075.aspx> . Acesso em 18 mar. 2013.

NUNES, Maria Augusta S.N. et al. **Uso do Kinect para a extração de características afetivas do usuário** In: Simpósio Brasileiro de Informática na Educação e Workshop de Informática na Escola, 22º e 17º, 2011, Aracaju, Workshop: Towards Affective Computing in Education: how to enhance the student affective experience to foster learning, 2011, p. 1808-18015.

ORACLE. **Oracle 11g PL/SQL Technical Information**. 2013. Disponível em : < http://www.oracle.com/technetwork/database/features/plsql/index.html> . Acesso em 05 jun. 2013.

RAMOS, C.D.A. **Ferramenta para Mineração de Sequências de Eventos Temporais**. Brasil, Universidade Luterana do Brasil, 2010. 20 p.

REZENDE, S.O. et al. **Sistemas Inteligentes**: Fundamentos e Aplicações. Barueri, SP: Ed. Manole, 2003. p.

307-333

SEBRAE. **Taxa de Sobrevivência das Empresas no Brasil**. Coleção Estudos e Pesquisas, 2011. 30 f. Disponível em: < http://www.biblioteca.sebrae.com.br/bds/BDS.nsf/45465B1C66A6772D832579300051816C/$File/NT00046582.pdf>. Acesso em: 01 abr. 2013.

UNDERHILL, Paco. **Why We Buy**. Estados Unidos: Ed. Simon and Schuster, 2008. p. 3-27.