UM QUADRO INTERACTIVO: COMPARAÇÃO DE ALGORITMOS DE VISÃO PARA DETECÇÃO DE INTERACÇÕES

RESUMO: *Na era digital actual, a adopção de interfaces naturais entre o homem e a máquina torna-se cada vez mais pertinente. Na educação, em particular, a utilização de ferramentas interactivas para melhorar as práticas pedagógicas, auxiliar a compreensão de conceitos complexos e permitir o trabalho colaborativo constitui uma vantagem inequívoca. A panóplia de soluções comerciais disponíveis é vasta, mas o custo associado é geralmente elevado para equipar de forma generalizada os estabelecimentos de ensino, principalmente nos países de economia mais débil. Neste contexto, este trabalho propõe um sistema de quadro interactivo de código aberto, com requisitos de hardware muito pequenos (i.e., um computador com uma câmara de vídeo WEB e um videoprojector) que poderá permitir massificar o seu uso. Neste artigo é apresentada a estrutura física e lógica do sistema proposto e efectuado um estudo comparativo do desempenho de diferentes algoritmos de detecção de pontos de interesse nas imagens captadas pela câmara de vídeo. Estes algoritmos fazem parte de um dos módulos centrais da arquitectura do sistema do quadro interactivo.*

Palavras-chave: Quadro Interactivo, Algoritmos de Visão, Detecção e *Tracking* de BLOBs, OpenCV.

INTRODUÇÃO

Ao longo dos anos a evolução tecnológica tem sido notável, no entanto, nem todas as áreas têm seguido o mesmo ritmo. Na educação, por exemplo, o objecto central da sala de aula é ainda o quadro que, na maioria dos casos, não faz parte da era digital. Não pela inexistência de quadros interactivos digitais no mercado, mas principalmente pelos elevados custos que estes acarretam, impedindo a sua adopção generalizada. Contudo, os quadros interactivos têm uma série de qualidades inexistentes nos seus predecessores; promovendo uma maior interacção e discussão dos conceitos em sala de aula por via da utilização de recursos multi-mediáticos. Por outro lado, é possível gerar documentos das sessões projectadas nesses mesmos dispositivos, melhorando a aquisição de apontamentos.

O desafio deste trabalho consiste na criação de uma ferramenta de software que implemente um quadro interactivo com recurso a um computador normal equipado com uma câmara de vídeo Web comum. Pretende-se que esta ferramenta utilize algoritmos de visão na detecção das interacções com o quadro (via apontadores infravermelhos) e que seja independente do sistema operativo. A plataforma ajusta-se às condições do ambiente de projecção e retorna as coordenadas dos pontos de interacção para qualquer aplicação que delas faça uso. O objectivo é mostrar que, com hardware comum e software apropriado é possível obter um sistema de quadro interactivo genérico, barato e útil em sala de aula. Um dos aspectos mais importantes neste tipo de sistemas é o suporte à detecção de interacções, através da análise das imagens recolhidas pela câmara de vídeo. Esta detecção é efectuada por algoritmos de detecção de *regiões* (BLOBs), i.e. conjuntos de píxeis com características comuns que se conseguem isolar numa imagem. Para além de analisar estes algoritmos, o artigo apresenta também uma avaliação comparativa de algoritmos para a detecção de cliques e movimentos do apontador de infravermelhos (IR) com aplicação possível no sistema de quadro interactivo proposto.

Este artigo encontra-se organizado em 4 secções. A primeira, define o enquadramento e os objectivos do projecto. A segunda secção apresenta um estudo dos projectos relacionados realçando elementos importantes a reutilizar no sistema apresentado. A arquitectura do projecto e os algoritmos que a acompanham são descritos na terceira secção. Finalmente é efectuada uma avaliação dos diferentes algoritmos estudados e é apresentada uma breve conclusão.

PROJECTOS RELACIONADOS

Pretende-se nesta secção, em primeiro lugar, identificar bibliotecas ou ferramentas de suporte ao desenvolvimento de quadros interactivos e, em segundo lugar, analisar e comparar os objectivos propostos em projectos de quadros interactivos afins.

Bibliotecas e Ferramentas de Suporte

O OpenCV [1], é uma biblioteca de código aberto (licença BSD) que contém rotinas básicas de processamento de imagem apropriadas ao desenvolvimento de aplicações na área de Visão Computacional. Com base nesta biblioteca é

possível adquirir imagens através de uma câmara e efectuar o seu processamento de forma a obter informação sobre pontos de interesse nas mesmas. A área de visão computacional tem registado evoluções consideráveis nos últimos anos, com aplicações na área médica e na industria, através da robótica e na engenharia aeroespacial na concepção de veículos autónomos. Em [1] é realçada a mais valia de usar o OpenCV em detrimento de outras bibliotecas quando se trata de analisar imagens em tempo real.

Existem já algumas plataformas com características semelhantes às do sistema aqui apresentado. São os casos da Touchlib [6] [7] e da Tbeta [8]. Estas plataformas utilizam o OpenCV para efectuar detecção dos pontos de interesse nas imagens captadas pela câmara. Esta decisão, como veremos, tem consequências na carga de processamento necessária. Verificamos, em concreto, essa situação com o Touchlib que usa a função *cvFindContours* de forma linear acompanhado de salto nas linhas (y). Destacamos a mais valia de reutilizar o protocolo existente TUIO [9] desenvolvido para estes sistemas, o qual permite reportar coordenadas dos pontos de interesse (PI), ou BLOBs através de mensagens num formato comum.

A figura 1 [11] ilustra como é possível, usando uma câmara, reportar BLOBs de radiação infra-vermelha. O sistema pode ser divido em duas máquinas distintas através do uso de mensagens TUIO podendo ter uma máquina a fazer detecção de pontos e outra dedicada à projecção e interacção com o utilizador.

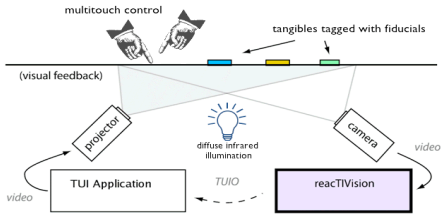

Figura 1 - Funcionamento do projecto TUIO.

Por fim, destaca-se o projecto desenvolvido por Jonhy Lee [10] como sendo um dos pioneiros na concepção de quadros interactivos de baixo custo reutilizando um comando da Nintendo, o *Wiimote*. Esse dispositivo permite a detecção de pontos IR e, por bluetooth, reporta as respectivas coordenadas ao dispositivo emparelhado. Este tipo de sistema possui algumas limitações. A primeira é que o dispositivo constitui propriedade intelectual do criador da Nintendo, permanecendo os algoritmos usados desconhecidos dos programadores. Um segundo inconveniente é o facto da precisão da detecção estar limitada pela resolução da câmara embutida na *Wiimote*: apenas 128x96. Por outro lado, a utilização da câmara de vídeo disponível na maioria dos computadores recentes é sempre preferível à utilização de um periférico adicional.

Projectos Abertos de Quadros Interactivos

Nesta secção pretende-se identificar e perceber os objectivos e as dificuldades encontradas em projectos afins. Em [2] os autores realçam a necessidade de evolução do modelo tradicional do uso de um quadro. Os autores propõem uma solução baseada numa câmara, um microfone e um computador para conseguirem armazenar num documento digital as várias partes de uma reunião. Esse documento é pesquisável permitindo, assim, rever e ouvir partes da reunião que sejam do nosso interesse. O objectivo é permitir aos participantes focarem-se exclusivamente nos objectivos da reunião, sem se preocuparem em tirar notas e evitando o esquecimento de tarefas ou ideias discutidas.

Os autores de [3] pretendem uma migração suave entre o mundo digital e os sistemas tradicionais. Será possível para uma câmara distinguir a nova informação que foi escrita, quer essa esteja num quadro interactivo (meio digital) ou num quadro com giz (meio tradicional) essa é a problemática do artigo. Podemos com este modelo ter um suporte para ambos os métodos, quer estes sejam mais inovadores ou tradicionais. Algumas técnicas abordadas neste artigo podem revelar-se importantes, nomeadamente as técnicas de calibração, que permitem, por sua vez, gerar previsões. No artigo, os autores conseguem prever em que posição e de que forma a imagem projectada irá ser visualizada pela câmara. Esta informação é importante porque permite isolar a informação conhecida dos novos dados, permitindo ao algoritmo focar sua atenção em determinadas áreas consideradas de interesse. O artigo deixa saliente a necessidade de recorrer a calibração. É necessário calibrar a cor, porque quando uma imagem é projectada ela pode-se tornar mais escura ou mais clara dependendo das condições de luminosidade ou mesmo do projector e a geometria também difere consoante a distância entre o projector e a área projectada, sendo necessário criar uma escala que nos permita mapear a resolução entre a imagem real e a projectada.

A utilização de quadros interactivos em ambiente de trabalho colaborativo e aprendizagem é defendida em [4] e [5]. O primeiro artigo revela um estudo efectuado pela XEROX PARC avaliando de que forma os seus funcionários recorriam ao quadro interactivo e qual o tipo de informações

anotadas no mesmo. Esta analise permite conceber um sistema mais próximo das necessidades do futuro utilizador.

SISTEMA PROPOSTO

Nesta secção, apresentamos inicialmente a arquitectura e funcionalidade da ferramenta de suporte ao quadro interactivo que foi desenvolvida. De seguida descrevemos os algoritmos utilizados na detecção de BLOBs que serão comparados posteriormente.

Arquitectura do Quadro Interactivo

O sistema é composto pelo software, de código aberto, instalado num vulgar computador, por uma câmara Web normal (ou qualquer outro tipo de câmara genérica), por um dispositivo apontador com uma fonte infravermelha incorporada e por um mecanismo de projecção (videoprojector). A robustez e a simplicidade de utilização foram dois importantes requisitos tidos em conta de modo a que o sistema seja facilmente usado por utilizadores sem conhecimentos de programação. A câmara deverá capturar, maioritariamente, a radiação infravermelha. Uma forma muito simples de obter esse efeito consiste em colocar, em frente à sua lente, um filtro de luz que só deixe passar luz infravermelha, como por exemplo, película fotográfica. Usamos, como dispositivo apontador, uma caneta com um led IR montado na extremidade, alimentado por uma pilha acomodada no interior da caneta. Segue-se depois um fase de calibração a fim de estabelecer um factor escala entre a resolução do quadro é a do seu computador, para mapear os movimentos. No final obtemos um sistema que nos permite interagir com o rato do nosso computador somente através dos deslocamentos do apontador na projecção. Sendo que o sistema reconhece a posição e a acção que o utilizador pretende fazer no mesmo ponto.

Podemos ver, no diagrama de blocos da figura 2, os 4 processos mais importantes efectuados pelo sistema: primeiro dentro deles é a aquisição de uma imagem feita por parte da câmara, depois nessa mesma imagem serão aplicados alguns filtros a fim de melhorar a qualidade da mesma, na terceira fase entram os algoritmos de detecção e seguimento, os quais abordaremos à frente, e por fim reportamos as coordenadas do ponto encontrado, caso esse não exista será retornado o ponto (-1,-1). No passo 2, utilizando funções do OpenCV o sistema pode, opcionalmente, melhorar a qualidade da imagem recolhida pela câmara, baseando-se em técnicas de remoção de fundo. Ele permite-nos construir um modelo do *background*, que podemos usar para subtrair a cada uma das novas imagens, obtendo como resultado final apenas o *foreground* de cada imagem.

Figura 2 - Ciclos dos processos do sistema.

Algoritmos Detecção e Seguimento

Esta secção descreve o modo como é realizada a análise de cada imagem para a detecção da posição de interacção no formato de coordenadas cartesianas (x, y).

Um vídeo é composto por uma sequência de imagens consecutivas. Cada imagem representa uma matriz de píxeis. Cada píxel é formado por três valores, representando as componentes vermelho, verde e azul do espaço de cor RGB. Convertendo uma imagem para uma escala de cinzentos (*grayscale*), passamos a ter apenas um valor por cada píxel. No processamento em tempo real de cada imagem, esta transformação revela-se fundamental sendo que a analise pode ser até três vezes mais rápida do que com imagens a cores. Por outro lado, a aplicação fica impossibilitada de reconhecer e distinguir cores diferentes. Outro aspecto a ter em conta são as coordenadas do PI. Estas são reportadas tendo como origem o ponto (0,0) do canto superior esquerdo de cada imagem adquirida da câmara de vídeo, e a coordenada máxima é definida por (largura, altura). Dependendo da resolução da câmara, podemos trabalhar sobre imagens com diferentes tamanhos.

Segue-se uma breve descrição dos 5 algoritmos de detecção e seguimento dos BLOBs gerados pela caneta IR. Podemos ver na figura 3 a legenda que será usada e que será comum a todos eles.

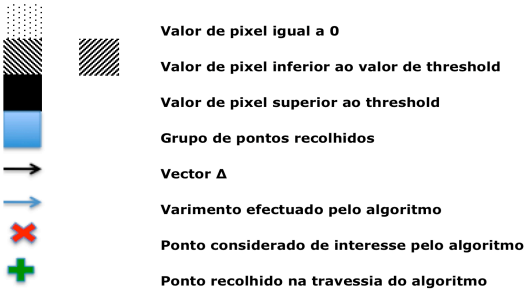

Figura 3 - Legenda usada na definição dos algoritmos.

Algoritmo simples de detecção (A1): neste algoritmo é efectuado um varrimento píxel a píxel da imagem, como é ilustrado na figura 4, de modo

a encontrar um valor superior a um determinado limiar (*threshold*) ou valor máximo aceitável definido pelo utilizador na fase de inicialização. Este algoritmo recolhe as coordenadas de todos os pontos que têm um valor superior ao *threshold* e retorna a média dos valores em x e y.

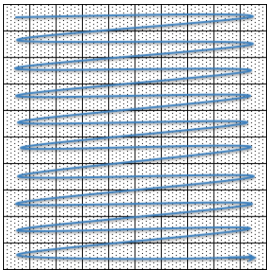

Figura 4 - Algoritmo A1.

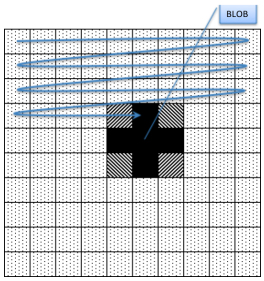

Figura 5 – A2 (1º passo).

Algoritmo simples de detecção com salto (A2): este algoritmo tem um funcionamento semelhante ao anterior, mas permite definir um factor de salto S. Desta forma, entre duas leituras de píxeis, o algoritmo ignora S-1 píxeis. No processamento da coordenada do PI usamos o mesmo método, ou seja, reportamos a média dos valores recolhidos em x e y. Quanto menor for o valor de S, mais preciso será o cálculo do centro do BLOB a detectar.

Algoritmo simples de detecção com salto - versão 2 (A3): as diferenças entre este algoritmo e o anterior visam reduzir o custo de processamento de cada imagem, em casos específicos: quando existe um PI na frame e o PI não esteja na coordenada (largura, altura). O algoritmo começa a travessia da frame usando como valor o salto que foi definido pelo utilizador; quando é encontrado um píxel com valor superior ao *threshold*, interrompe a pesquisa, e aplica nessa área um algoritmo para determinar quais as coordenadas do PI (ver figura 5).

Para encontrar o centro faz-se uma pesquisa nas 4 direcções possíveis com origem no ponto encontrado. Primeiro pesquisa-se na horizontal, e determina-se o máximo e o mínimo nessa linha. Depois aplica-se uma pesquisa na vertical com origem, no eixo dos x, o valor médio entre o mínimo e máximo encontrados anteriormente, e no eixo dos y, o mesmo valor do ponto original. Com a mesma técnica encontra-se um máximo e um mínimo na vertical. Os dois valores médios dão-nos o centro do PI (ver a figura 6).

Algoritmo com previsão e pesquisa em espiral (A4): este algoritmo propõe uma abordagem diferente dos anteriores. Foca-se não tanto na detecção mas mais no seguimento de um PI. Ele deverá apresentar melhores resultados quando um ponto se movimentar (operação de *drag* do dispositivo apontador) ao longo de imagens consecutivas. Usamos neste algoritmo um vector, que guarda a informação sobre o deslocamento, ou seja, a diferença de deslocamento entre duas imagens consecutivas. Consideramos as seguintes três frames consecutivas: F1, F2 e F3. Todas contêm um PI em movimento, que tem por coordenadas, respectivamente, **p1** = (x1,y1), **p2** = (x2,y2) e **p3** = (x3,y3). O valor do vector deslocamento do PI entre F1 e F2, $\Delta\mathbf{s}$, é calculado da seguinte forma:

$$\Delta\mathbf{s} = (\Delta x, \Delta y) = (x2-x1, y2-y1).$$

Podemos então reutilizar $\Delta\mathbf{s}$, juntamente com uma pesquisa em espiral, para encontrar rapidamente as coordenadas do PI em F3. Como $\Delta\mathbf{s}$ tem o valor do ultimo deslocamento, prevemos que o deslocamento se mantenha em frames consecutivas, pressuposto que é válido a não ser que termine, abruptamente, a operação de arrastamento (*drag*). A estimação $\hat{p}_3$ para as novas coordenadas do PI em F3 será calculada da seguinte forma:

$$\hat{p}_3 = p2 + \Delta\mathbf{s} = (x2+\Delta x, y2+\Delta y)$$

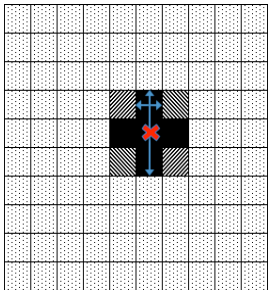

Figura 6 - A3 (2º passo).

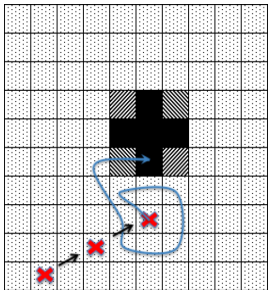

Figura 7 - Algoritmo A4.

Com base nesta previsão, iremos iniciar nossa pesquisa na frame não no tradicional ponto (0,0) mas sim na previsão que acabamos de calcular. Usamos um mecanismo de espiral com a finalidade de aproximar a solução de encontrar um ponto com valor superior ao *threshold* o mais rapidamente possível. O mecanismo para pesquisa em espiral tradicional seria muito drástico no uso do CPU por isso optou-se por pré construir uma tabela (*lookup table*) com os deslocamentos prévios em x e em y definindo o movimento de uma espiral como se pode constatar na figura 7.

Constatamos que quando um ponto desaparece, a expansão da espiral a toda a imagem pode revelar-se ineficiente quando comparada com o método de pesquisa linear. Os motivos são relacionados com armazenamento em memória da imagem. Aquando uma pesquisa em espiral efectuamos repetidamente vários saltos entre zonas de memória, enquanto numa pesquisa linear analisamos zona de memorias consecutivas permitindo que o processo seja desempenhado mais rapidamente. Por esse motivo decidimos limitar a sua expansão com um parâmetro que corresponde ao número máximo N que

corresponde aos níveis (círculos) permitidos. Quando o algoritmo nada encontrar no limite da espiral e ainda permanecer uma área da imagem por percorrer, recorremos a uma abordagem de pesquisa linear a fim de verificar a presença ou não de um PI nos restantes píxeis.

Algoritmo multiponto (A5): este algoritmo permite num simples varrimento linear da imagem recolher informação sobre todos os pontos de informação presentes como podemos ver na figura 8. De seguida ele analisa os valores recolhidos e determina diferentes grupos de forma a saber quantos pontos tem em cada imagem. Por fim ele analisa esses grupos para conhecer quais são as coordenadas dos seus centróides, como é ilustrado na figura 9. O algoritmo descrito foi adaptado do algoritmo descrito por Erik van Kempen [12]. O autor, em [12], salienta a eficiência temporal desta técnica usada para o reconhecimento de vários pontos de interesse num só varrimento de imagem.

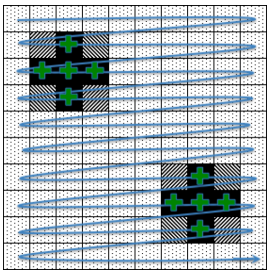

Figura 8 - Travessia e captura dos PI.

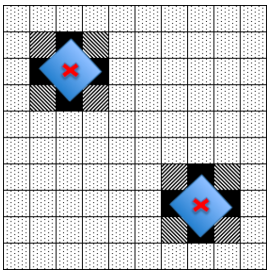

Figura 9 – Definição dos grupos e cálculo do centróide.

AVALIAÇÃO DOS ALGORITMOS

Nesta secção iremos analisar o desempenho de cada um dos algoritmos anteriormente descritos.

Metodologia

Os testes foram realizados sempre na mesma máquina e com as mesmas definições de compilação. Foram criados seis vídeos de teste para serem submetidos aos algoritmos descritos (figura 10). Os vídeos incluem vários tipos de interacções com o apontador de infravermelhos: cliques, movimentos contínuos e uma mistura de ambas as interacções. Foram ainda criados vídeos com e sem ruído, ou seja, com e sem a presença de fontes de ruído na imagem de fundo (e.g., luzes, reflexões, etc.).

Durante a simulação de cliques a interacção de infravermelhos aparece e desaparece em pontos aleatórios da imagem. No movimento o ponto, depois de aparecer, continua a deslocar-se pela imagem em movimentos aleatórios. Na simulação de interacções diversificadas foram conjugados os dois tipos de interacção definidos anteriormente. Por outro lado, foram também introduzidas variações (ruído) nas condições de ambiente, que podem interferir na determinação de um ponto de interesse. Os vídeos com ruído são pré-processados para remover o fundo e isolar o ruído. Os algoritmos descritos anteriormente foram numerados de A1 a A5 de modo a podermos identificá-los nos gráficos decorrentes de cada experiência.

	Simulação	Ambiente
Vídeo 1	Cliques	Sem Ruído
Vídeo 2	Movimentos	Sem Ruído
Vídeo 3	Diversificado	Sem Ruído
Vídeo 4	Cliques	Com Ruído
Vídeo 5	Movimentos	Com Ruído
Vídeo 6	Diversificado	Com Ruído

Figura 10 – Vídeos de teste.

Primeira experiência: no primeiro estudo comparamos o desempenho de cada um dos algoritmos. Na realização destas experiências usamos 500 imagens como limite da amostragem. Primeiro, medimos a taxa de utilização do processador de cada algoritmo em cada um dos vídeos. Depois, calculamos a média desses valores (ver figura 11). Os algoritmos que apresentam melhores médias são o 4 e o 5.

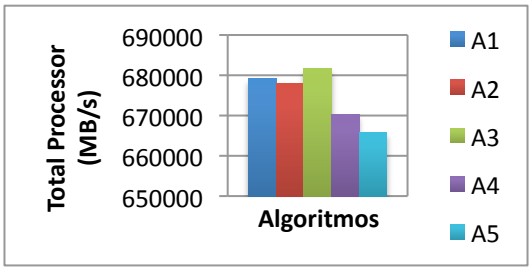

Figura 11 - Taxa média de utilização do CPU.

De forma a medir o desempenho de cada algoritmo foram criados perfis temporais. Os valores apresentados representam em percentagem o tempo que o algoritmo foi executado durante a amostragem. Os parâmetros usados foram, no A2 e A3 S=3 (salto) e no A4 N=5 (níveis). Podemos ver, na figura 12, que os algoritmos com desempenho mais eficiente foram o A3 e o A4 (devido ao seu mecanismo de previsão nos vídeos 2 e 5), seguidos por A2, A1 e A5, sendo esse o menos eficiente.

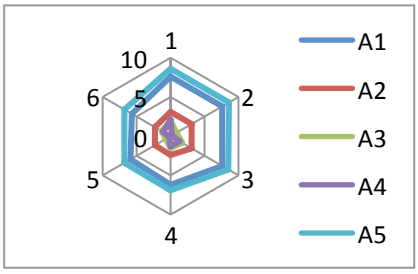

Figura 12 - Avaliação do tempo de execução de cada algoritmo em cada vídeo.

Segunda experiência: esta segunda fase focou-se sobre a precisão na detecção dos prontos de interesse por parte dos algoritmos sobre cada um dos vídeos. Usamos o vídeo 6, com pré-processamento de imagem para melhorar a sua qualidade e recolhemos as coordenadas dos 7 primeiros pontos encontrados. Repetimos estes passos para os 5 algoritmos (ver figura 13). As coordenadas foram calculadas usando a fórmula:

valor = x + (y*largura)

Onde x corresponde à posição estimada pelo algoritmo do ponto na horizontal e y na vertical.

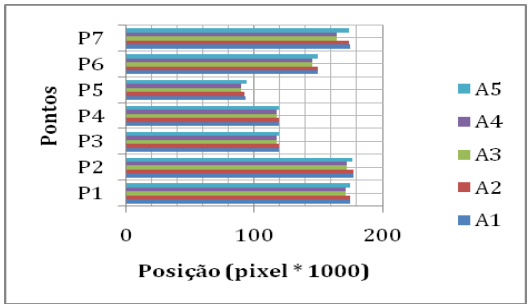

Figura 13 - Avaliação da previsão das coordenadas de um PI.

Terceira experiência: na última fase analisamos o impacto da mudança de duas variáveis nos algoritmos 3 e 4. Nesta experiência usamos o vídeo 3, que contém movimentos e cliques. Possivelmente, este vídeo será o que mais se aproxima dos casos de uso reais.

No algoritmo 3 fazemos variar o valor do salto para verificar o impacto no tempo de execução do algoritmo (através de um perfil temporal da execução). A percentagem de tempo obtida corresponde à razão entre o tempo de execução do algoritmo em relação ao tempo de amostragem total (ver figura 14).

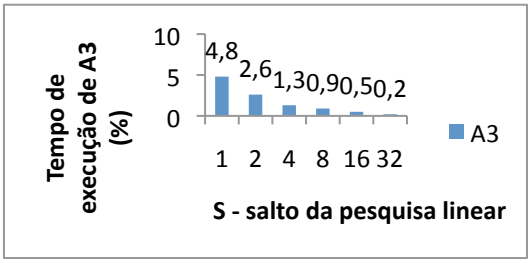

Figura 14 – Tempo de execução de A3 variando o salto**.**

No algoritmo 4 fazemos variar o tamanho da espiral, e verificamos a taxa de sucesso do algoritmo para encontrar o ponto. Podemos ver os resultados na figura 15.

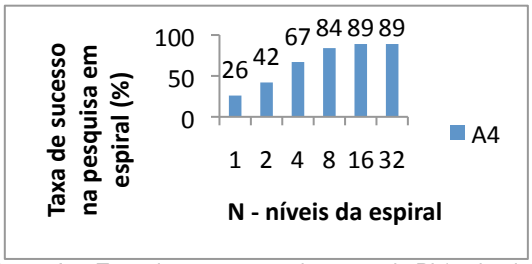

Figura 15 - Taxa de sucesso na detecção de PI (variando o tamanho da espiral).

CONCLUSÃO

Neste trabalho pretendeu-se combinar conhecimentos de múltiplas áreas de saber para obter uma solução simples, económica, eficaz de quadro interactivo que vá ao encontro das necessidades do utilizador final. Este é um trabalho em desenvolvimento pelo que os resultados apresentados são ainda preliminares. Os próximos passos centrar-se-ão no refinamento da implementação da plataforma e na sua utilização em aplicações concretas que possam ser utilizadas em sala de aula de modo a ilustrar o seu funcionamento e a testar e avaliar todo o conjunto em ambientes reais. Outras direcções a explorar constituem a possibilidade de reportar pontos de interesse, para outras aplicações, através do protocolo TUIO ou outro similar e a incorporação plena da tecnologia multi-toque no sistema a implementar.

Algumas das limitações encontradas resultaram do funcionamento das bibliotecas de tratamento de imagem utilizadas (e.g., na fase de aquisição e pré-processamento de uma *frame*), que por vezes dificultam a identificação dos sinais de infravermelhos efectuada. Contudo estes problemas foram superados, permitindo a este projecto oferecer aos programadores, mais uma alternativa às plataformas existentes. Por outro lado, deve-se salientar que o código fonte do projecto será partilhado com toda a comunidade, permitindo desta forma dar-lhe continuidade e melhorá-lo, e eventualmente contribuir para as soluções e plataformas já existentes.

Referências Bibliográficas

[1] Bradski, G.; Kaehler, A., Outubro 2008, Learning OpenCV: Computer Vision with the OpenCV Library, 1º edição, O'Reilly Media, Inc.

[2] Li-Wei He, Zicheng Liu, Zhang, Z., 2003, Why Take Notes? Use the Whiteboard Capture System. Microsoft Research, One Microsoft Way, Redmond, WA, USA.

[3] Hanning Zhou, Zhengyou Zhang, Thomas Huang, 2004, Visual echo cancellation in a projector-camera- whiteboard system. Microsoft Research, One Microsoft Way, Redmond, WA, USA.

[4] Elizabeth D.Mynatt, Takeo Igarashi, W. Keith Edwards, Anthony LaMarca, 2000, Designing an Augmented Writing Interface. IEEE Computer Graphics and Applications.
[5] Yonald Chery, 2000, Bringing the Common Whiteboard into the Digital Age. IEEE Multimédia – Multimedia at Work.
[6] David Wallin. Touchlib: an opensource multi-touch framework. 2006. URL http://www.whitenoiseaudio.com/touchlib/, (14/12/2008).
[7] Touchlib – Home. URL http://www.nuigroup.com/touchlib, (15/12/2008).
[8] Tbeta – The beta release. URL http://tbeta.nuigroup.com/ , (15/12/2008 , 22h00).
[9] TUIO: A Protocol for Tangible User Interfaces. URL http://tuio.lfsaw.de/ , (15/12/2008).
[10] Johnny Chung Lee - Projects - Wii: Low-Cost Multi-point Interactive Whiteboard using the Wiimote. URL http://www.cs.cmu.edu/~johnny/projects/wii/, (15/08/2008).
[11] Reactable - ReactVision. URL http://mtg.upf.edu/reactable/?software (14/12/2008).
[12] Geekblog.nl Blob detection V: growing regions algorithm URL http://geekblog.nl/entry/24 (14/11/2008).