Real-time Manipulation Software

Diogo Daniel Soares Ferreira Nº 76504
Luís Davide Jesus Leira Nº 76514
Turma prática P2 – Computação Visual

Resumo – **O presente relatório apresenta a descrição detalhada de um trabalho realizado no âmbito da unidade curricular de Computação Visual. O resultado é uma aplicação de manipulação de vídeo, tendo como base a programação em C++, com recurso à library OpenCV.**

O relatório começa por introduzir os objetivos do trabalho em questão e a apresentação da aplicação. De seguida, são apresentados os ficheiros que constituem o projeto e é exposta uma breve descrição sobre cada um deles. As funcionalidades são depois descritas com o objetivo de familiarizar o leitor com a aplicação. Depois descrevem-se detalhes sobre a implementação das funcionalidades referidas anteriormente. A seguir são apresentados alguns resultados obtidos através da utilização da ferramenta criada e a discussão dos mesmos. Finalmente, são apresentadas observações e limitações técnicas verificadas.

I. INTRODUÇÃO

Neste relatório iremos explicar as principais funcionalidades do projeto *Real-Time Manipulation Software*. O projeto foi desenvolvido no âmbito da cadeira de Computação Visual do 4º ano do Mestrado Integrado em Engenharia de Computadores e Telemática da Universidade de Aveiro.

A ideia inicial para este projeto teve origem na possibilidade de criar uma aplicação que pudesse gravar vídeos e aplicar efeitos aos mesmos, em tempo real. Posteriormente, verificámos que também seria interessante carregar vídeos armazenados previamente e manipulá-los. Para conseguir a fácil manipulação dos efeitos no vídeo por parte de um utilizador e para a interface ser intuitiva, foi desenhada uma interface gráfica baseada no Qt.

Assim sendo, foram definidos 36 efeitos possíveis (descritas na secção III. Funcionalidades) que a aplicação necessita de suportar, bem como permitir ao utilizador aplicar o mesmo efeito mais do que uma vez, e definir a ordem dos efeitos aplicados no vídeo.

II. DESCRIÇÃO BREVE DO CONTEÚDO DOS FICHEIROS

O projeto apresenta um conjunto de ficheiros necessários à execução da aplicação. Nesta secção apresentamos uma breve descrição de cada ficheiro:

- **main.cpp -** Neste ficheiro encontra-se o a função que inicia a janela de Qt utilizada para executar a aplicação.
- **RealTimeManipulationSoftware.h** e **RealTimeManipulationSoftware.cpp** – Este ficheiro contém a classe da janela principal utilizada para o projeto. Contém os dois *players* que são usados para reproduzir os vídeos e as ações a efetuar quando são selecionados botões na janela.
- **Player.h** e **Player.cpp** – Este ficheiro contém as funções necessárias para reproduzir um vídeo de OpenCV no Qt. Contém ainda funções para carregar um vídeo, filmar, parar a gravação, guardar o vídeo no ficheiro, e aplicar as transformações ao vídeo reproduzido.
- **Aux_functions.h** e **Aux_functions.cpp -** Neste ficheiro encontram-se as funções que aplicam as transformações a um frame e retornam um frame modificado.
- **ParameterInserting.h** e **ParameterInserting.cpp** - Neste ficheiro encontram-se as funções que criam uma nova janela para receber do utilizador um número variável de parâmetros, de acordo com a função.
- **Transformations.h** - Este ficheiro contém uma classe base: *Transformation*, que têm uma função virtual *applyTransformation()*. Todas as outras classes derivadas armazenam uma função e os seus argumentos, e chamam a função armazenada c argumentos armazenados. É utilizado para criar uma lista de transformações, onde todas podem ser aplicadas da mesma maneira, apenas utilizando a função *applyTransformation()*.

III. FUNCIONALIDADES

Nesta secção iremos explorar o modo de funcionamento da aplicação, bem como detalhar todas as funcionalidades relevantes.

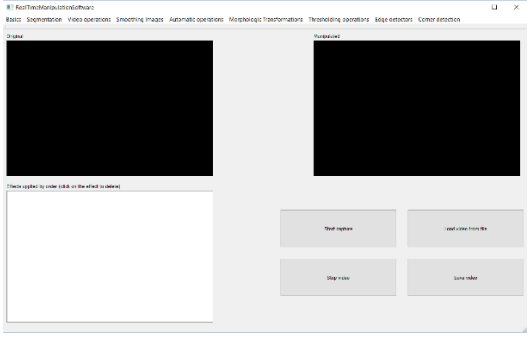

Figura 1 – Imagem da aplicação.

Ao iniciar a aplicação, o utilizador pode ver uma janela com dois retângulos negros. No lado esquerdo, irá ser reproduzido o vídeo original. Do lado direito, irá ser reproduzido o mesmo vídeo com as transformações aplicadas. No canto superior esquerdo, pode ser vista uma lista com o nome das transformações aplicadas pelo utilizador (inicialmente está vazia). Do lado direito, o utilizador tem acesso a quatro botões.

O botão "Start capture" irá iniciar a captura de um vídeo a partir da câmara do computador (se existir). Ao iniciar a reprodução de um vídeo, são criadas duas janelas com histogramas, uma com o histograma do vídeo original e outra com o histograma do vídeo com as transformações aplicadas. Após o início da reprodução do vídeo, é possível parar a sua reprodução no botão "Stop video".

Também é possível reproduzir um vídeo a partir de um ficheiro, no botão "Load video from file", sendo necessário depois selecionar o vídeo no formato ".avi". Finalmente, também é possível guardar o vídeo com as transformações aplicadas num ficheiro, sendo necessário carregar no botão "Save video" antes de o iniciar, e escolhendo a pasta e o nome do novo ficheiro, que será armazenado no formato ".avi".

É possível selecionar transformações a aplicar ao vídeo original no menu superior, estando divididas por categorias. Algumas transformações aceitam parâmetros para alterar o seu funcionamento. Os parâmetros podem ser inseridos através de uma nova janela, que contém a indicação dos valores aceitáveis e a descrição para cada parâmetro.

As transformações são aplicadas por ordem de aplicação do utilizador. A transformação mais recente será a última a ser aplicada. Para apagar transformações aplicadas no vídeo de saída é necessário carregar no nome do efeito, na lista de efeitos do lado esquerdo. Irá abrir uma janela que permitirá confirmar a remoção da transformação do vídeo.

Os efeitos que permitem a manipulação dos vídeos encontram-se divididos nos seguintes grupos:

A. Basics

Nesta secção de efeitos são apresentados alguns efeitos básicos ao processamento de imagens:

- **Red Image** – Apresenta apenas os tons vermelhos da imagem.
- **Blue Image** - Apresenta apenas os tons azuis da imagem.
- **Green Image** - Apresenta apenas os tons verdes da imagem.
- **Grey Image** – Converte a imagem para uma escala de cinzentos.
- **Invert Image** – Inverte os valores de cor da imagem para o seu simétrico.
- **Change Color** – Permite alterar o valor específico de uma cor para outra cor definida pelo utilizador.
- **Change Brightness and Contrast** – De acordo com dois parâmetros, alpha e beta, altera o contraste e a luminosidade da imagem.

B. Segmentation

Nesta secção de efeitos apenas existe o efeito de segmentação (**Create segmentation**), que efetua duas operações de erosão.

C. Video operations

Como o nome indica, esta secção de efeitos contém efeitos que são aplicáveis apenas a vídeos e não a imagens:

- **Background subtraction using KNN** - Subtração do fundo do vídeo utilizando o algoritmo de machine-learning K-nearest neighbors.
- **Background subtraction using MOG** - Subtração do fundo do vídeo utilizando o algoritmo de machine-learning mixture of gaussian.

D. Smoothing

Esta secção de efeitos contém apenas efeitos de smoothing, nomeadamente:

- **Mean Blur** – Aplica um filtro de média ao vídeo, de acordo com o tamanho do kernel definido pelo utilizador.
- **Median Blur** – Aplica um filtro de mediana ao vídeo, de acordo com o tamanho do kernel definido pelo utilizador.
- **Gaussian Blur** – Aplica um filtro gaussiano ao vídeo, de acordo com o tamanho do kernel e o desvio-padrão da curva gaussiana definidos pelo utilizador.

E. Automatic operations

Esta secção de efeitos contém efeitos que são aplicados de forma "automática" para tentar melhorar o video final:

- **Contrast Streching** – Aplica o contrast stretching ao vídeo, dado o histograma.
- **Equalize Histogram** – Equaliza o histograma do vídeo original, alterando o contraste.
- **Brightness and Contrast Auto** – Tenta normalizar o histograma automaticamente, alterando o contraste e a luminosidade do vídeo de forma automática; para isso, recebe uma percentagem inteira (de 1 a 100) das margens do histograma que irá "suprimir", e irá recalcular a luminosidade e o contraste do vídeo (se a

percentagem for 0, o vídeo mantém-se igual; se a percentagem for 100, apenas se vê uma imagem preta).
- **Back Projection** - Calcula a Back Projection de um frame.

F. Morphologic transformations

Esta secção de efeitos contém transformações morfológicas que são aplicados ao vídeo:

- **Erode** – Aplica erosão ao vídeo, dada uma forma (cruz, elipse ou retângulo), a altura e a largura do elemento estruturante; é possível converter a imagem a preto e branco, dado um threshold, ou aplicar a erosão à imagem a cores.
- **Dilate** - Aplica dilatação ao vídeo, dada uma forma (cruz, elipse ou retângulo), a altura e a largura do elemento estruturante; é possível converter a imagem a preto e branco, dado um threshold, ou aplicar a erosão à imagem a cores.
- **Opening** - Aplica abertura (erosão seguida de dilatação) ao vídeo, dada uma forma (cruz, elipse ou retângulo), a altura e a largura do elemento estruturante; é possível converter a imagem a preto e branco, dado um threshold, ou aplicar a erosão à imagem a cores.
- **Closing** - Aplica fecho (dilatação seguida de erosão) ao vídeo, dada uma forma (cruz, elipse ou retângulo), a altura e a largura do elemento estruturante; é possível converter a imagem a preto e branco, dado um threshold, ou aplicar a erosão à imagem a cores.
- **Morphological Gradient** - Aplica gradiente morfológico (diferença entre erosão e dilatação) ao vídeo, dada uma forma (cruz, elipse ou retângulo), a altura e a largura do elemento estruturante; é possível converter a imagem a preto e branco, dado um threshold, ou aplicar a erosão à imagem a cores.
- **Top Hat** - Aplica top hat (diferença entre o frame original e abertura do frame) ao vídeo, dada uma forma (cruz, elipse ou retângulo), a altura e a largura do elemento estruturante; é possível converter a imagem a preto e branco, dado um threshold, ou aplicar a erosão à imagem a cores.
- **Black Hat** - Aplica black hat (diferença entre o fecho e o frame original) ao vídeo, dada uma forma (cruz, elipse ou retângulo), a altura e a largura do elemento estruturante; é possível converter a imagem a preto e branco, dado um threshold, ou aplicar a erosão à imagem a cores.

G. Thresholding operations

Esta secção de efeitos contém os efeitos de thresholding aplicados aos vídeos:

- **Threshold Binary** – Recebe do utilizador dois valores numéricos: um limite de intensidade (threshold) e um valor máximo na escala dos cinzentos. O efeito irá igualar todos os pixeis da imagem com valor menor do que o threshold a zero, e todos os pixeis com valor superior ficarão com o valor máximo recebido do utilizador.
- **Threshold Binary Inverted** - Recebe do utilizador dois valores numéricos: um limite de intensidade (threshold) e um valor máximo na escala dos cinzentos. O efeito irá igualar todos os pixeis da imagem com valor menor do que o threshold ao valor máximo recebido, e todos os pixeis com valor superior ficarão com o zero.
- **Truncate** - Recebe do utilizador dois valores numéricos: um limite de intensidade (threshold) e um valor máximo na escala dos cinzentos. O efeito irá igualar todos os pixeis da imagem com valor superior ao threshold ao valor máximo recebido, e todos os pixeis com valor inferior manterão o seu valor.
- **Truncate to Zero** - Recebe do utilizador o limite de intensidade (threshold). O efeito irá igualar todos os pixeis da imagem com valor inferior ao threshold a zero. Os restantes pixeis mantêm o seu valor..
- **Truncate to Zero Inverted** - Recebe do utilizador o limite de intensidade (threshold). O efeito irá igualar todos os pixeis da imagem com valor superior ao threshold a zero. Os restantes pixeis mantêm o seu valor..

H. Edge detectors

Esta secção de efeitos contém cinco efeitos diferentes para deteção de arestas:

- **Sobel Operator** – Aplica o operador de Sobel aos frames do vídeo original. Recebe do utilizador dois coeficientes, um para aplicar o gradiente da derivada cada direção (x e y), e recebe também o tamanho do kernel, que poderá ser 1, 3, 5 ou 7.
- **Canny Detector** – Aplica o detetor de arestas Canny, utilizando dois thresholds recebidos do utilizador para distinguir entre arestas fracas e arestas fortes.
- **Detect Edges with Dilate** – Deteta as arestas dos frames recebidos do vídeo através da diferença entre a imagem original e a sua dilatação, ambas numa escala de cinzentos.
- **Laplace Operator** – Utiliza um operador laplaciano, de segunda derivada, para detetar arestas. Recebe do utilizador o tamanho do kernel a utilizar, um fator de escala para os valores calculados e um desvio que pode ser adicionado ao resultado final.

- **Find and Draw Contours** – Aplica o detetor de arestas Canny, e posteriormente desenha as arestas encontradas, de acordo com a cor definida pelo utilizador.

I. Corner detection

Esta secção contém dois efeitos que detetam e desenham os cantos nos frames recebidos do vídeo:

- **Harris Corner Detection** – Deteta os cantos na imagem, utilizando o metodo Harris-Stephens, e desenha os círculos nos cantos detetados no vídeo transformado.
- **Shi-Tomasi Corner Detection** - Deteta os cantos na imagem, utilizando o metodo Shi-Tomasi, e desenha os círculos nos cantos detetados no vídeo transformado.

IV. DETALHES DE IMPLEMENTAÇÃO

Nesta secção apresentamos alguns detalhes importantes da implementação efetuada.

A. Aplicação de transformações aos frames

Quando o utilizador aplica uma nova transformação ao vídeo original, a transformação é adicionada ao final de uma fila de transformações. Quando a imagem está a ser enviada para o *player*, irá percorrer todos os elementos da fila de transformações e aplicá-los.

A fila de transformações contém ponteiros para objetos da classe *Transformation*. A classe *transformation* contém um método virtual *applyTransformation(Mat frame)* que é utilizado para obter o frame com a transformação aplicada. Assim sendo, cada objeto poderá ter a sua classe derivada, onde poderá alterar a lógica da função *applyTransformation(Mat frame)*. Atualmente, as classes derivadas existentes armazenam a função e os seus argumentos, e aplicam a função com os seus argumentos quando a *applyTransformation(Mat frame) é chamada.*

B. *Reprodutor de vídeos*

A framework escolhida para desenvolver a interface gráfica para o projeto foi Qt, principalmente devido a utilizar a mesma linguagem de programação que a biblioteca utilizada para aplicar transformações aos vídeos (OpenCV), C++. Apesar da sua utilização ser recomendada pela própria documentação de *OpenCV*, não foi fácil conseguir integrar as duas bibliotecas e reproduzir vídeo nativo de *OpenCV* no *Qt.*

Para isso, foi necessário criar funções *low-level* que lêm frame a frame de um vídeo e imprimem o resultado num *label* de *Qt*. Especificamente, no ficheiro *player.cpp*, a função *run()* é utilizada para reproduzir um vídeo no *Qt.* Existe um ciclo infinito que irá obter a imagem do vídeo, frame a frame, aplica as transformações presentes na fila de transformações, apresenta o histograma para o frame e espera até ler o próximo frame, à frequência *frame rate.*

Como as transformações podem demorar algum tempo a ser aplicadas, nalguns casos é possível que o tempo de calcular a transformação ultrapasse o tempo disponível para emitir um frame. Nesses casos, o vídeo transformado ficará atrasado em relação ao vídeo original.

V. EXEMPLOS DE APLICAÇÃO E DISCUSSÃO DOS RESULTADOS OBTIDOS

Nesta secção iremos expor alguns resultados obtidos ao utilizarmos a aplicação desenvolvida como meio de prova de conceito, tal como a discussão dos mesmos.

Como esta aplicação se destina à manipulação de vídeos, a sua utilidade depende do propósito do utilizador em relação aos vídeos que pretende manipular. Desta forma, para verificarmos o real impacto da utilização de uma ferramenta deste tipo, foram utilizados alguns vídeos com determinadas características para que a sua manipulação possa vir a originar resultados com interesse, provando desta forma o propósito dos efeitos incorporados nesta aplicação.

A. Inverted Minecraft

O vídeo "Inverted Minecraft" encontra-se com as suas cores invertidas, não permitindo que a sua compreensão seja clara e dificulta a perceção do observador sobre o mesmo. Deste modo, a aplicação permite obter a correção do vídeo apresentado ao aplicar um efeito "Invert Image", o qual pertence ao grupo "Basics". Uma imagem do vídeo original e do manipulado estão presentes como Figura 2 e Figura 3

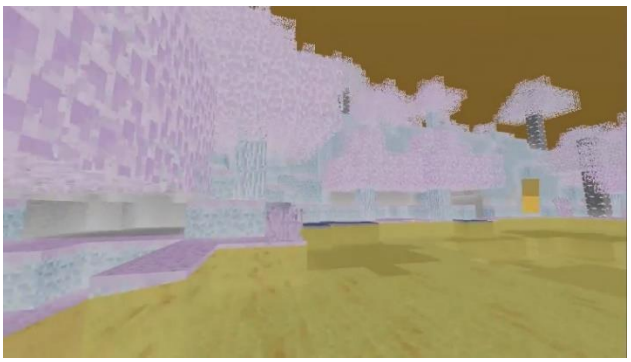

Figura 2 – Imagem do vídeo original "Inverted Minecrasf".

Figura 3 – Imagem do vídeo manipulado tendo como base o vídeo "Inverted Minecraft", após aplicado o efeito "Invert Image".

B. Light

O vídeo "Light" mostra uma fonte de iluminação, neste caso uma lâmpada, num teto de uma casa. Para diminuir a intensidade da lâmpada no vídeo, é possível aplicar o efeito "Truncate" (figura 5), tendo como threshold e como valor máximo na escala dos cinzentos 150.

Figura 4 – Imagem do vídeo original "Light".

Figura 5 – Imagem do vídeo manipulado tendo como base o vídeo "Light", após aplicado o efeito "Truncate".

Se o utilizador desejar filtrar a imagem apenas para obter a lâmpada, uma das opções é aplicar o efeito "Threshold to Zero", tendo como argumento 200, que irá "apagar" o resto do frame, e deixar apenas os pixeis que têm intensidade acima de 200 (figura 6).

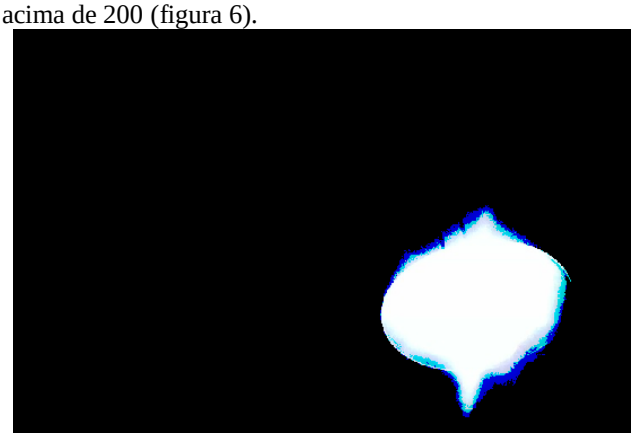

Figura 6 – Imagem do vídeo manipulado tendo como base o vídeo "Light", após aplicado o efeito "Threshold to Zero".

C. RGB HD

O vídeo "RGB HD" mostra três círculos de cores diferentes (vermelho, verde e azul) em movimento. Para obter apenas um dos círculos a partir de determinada cor, podemos aplicar efeitos do grupo "Basics", tais como "Red Image", "Green Image" ou "Blue Image". É possível também mudar uma determinada cor da imagem de através da operação anterior ao aplicar o efeito "Change Color...". Assim, como manipulação do vídeo aplicaram-se dois efeitos. Primeiro o "Red Image" para obter apenas a componente vermelha da imagem, deixando a preto tudo o resto (figura 8), e posteriormente aplicou-se também o "Change color" do preto predominante na imagem (RGB(10,0,0)) para branco (figura 9).

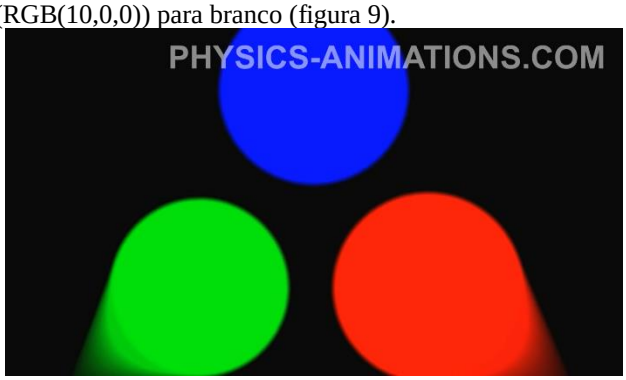

Figura 7 – Imagem do vídeo original "RGB HD".

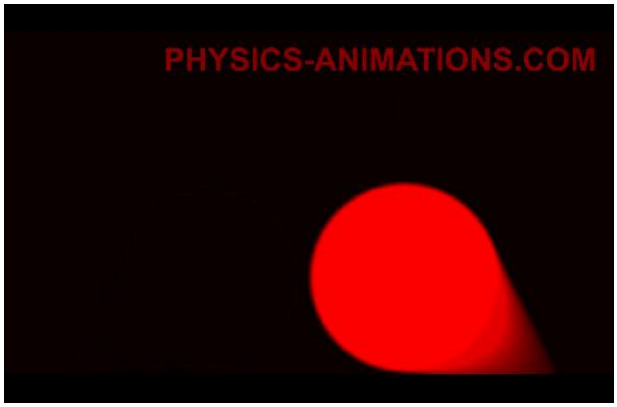

Figura 8 – Imagem do vídeo manipulado tendo como base o vídeo "RGB HD". Foi aplicado o efeito "Red Image".

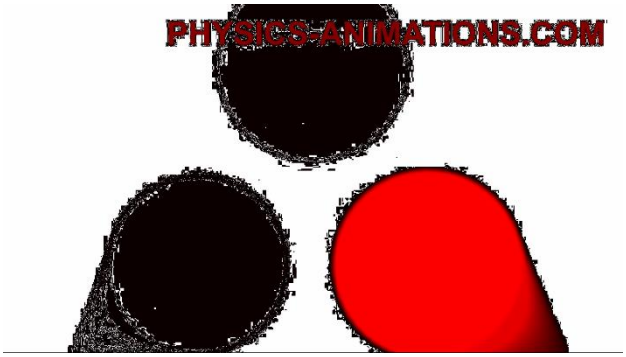

Figura 9 – Imagem do vídeo manipulado tendo como base o vídeo "RGB HD". Após ser aplicado o efeito "Red Image", foi aplicado o "Change Color" à cor de fundo predominante do frame.

D. Salt-and-Pepper Noise

O vídeo "Salt-and-Pepper Noise" mostra uma imagem com ruído, distúrbios que se apresentam como pixels brancos e pretos que ocorrem pela imagem toda. Neste caso, é típico aplicar-se um filtro de mediana ou uma filtragem morfológica. Para esta situação, a manipulação do vídeo foi através da aplicação do efeito "Median blur" do grupo "Smoothing images" (kernel size 3) (figura 11). Posteriormente, foi retirada a transformação de "Median blur" e foi aplicada uma operação de "Opening" (tratando-se de uma erosão e depois de uma dilatação) do grupo de "Morphologic Transformations" (em cruz com elemento estruturante 3x3), para comparar o resultado (figura 12). Ambas as operações removem com sucesso a maior parte do ruído nos frames.

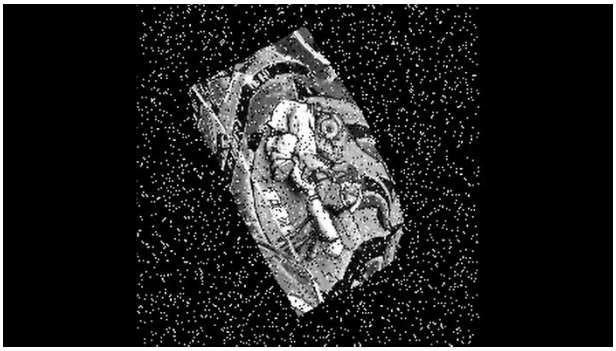

Figura 10 – Imagem do vídeo original "Salt Pepper Noise".

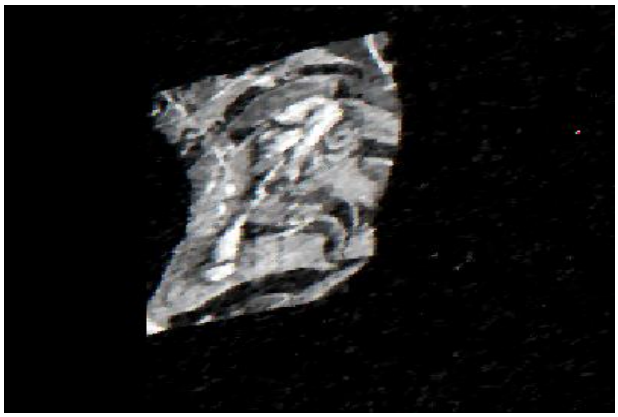

Figura 11 – Imagem do vídeo manipulado tendo como base o vídeo "Salt Pepper Noise", após aplicados o efeito "Median Blur".

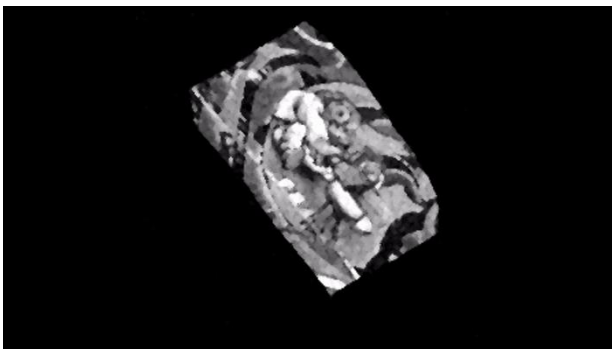

Figura 12 – Imagem do vídeo manipulado tendo como base o vídeo "Salt Pepper Noise", após aplicados o efeito "Opening".

E. Tiger Movement

Neste vídeo podemos ver um leão sobre um fundo verde, que se mexe na direção da câmara. Na figura 14 podemos ver o resultado do efeito "Background subtraction using MOG".

Figura 13 – Imagem do vídeo original "Tiger Movement".

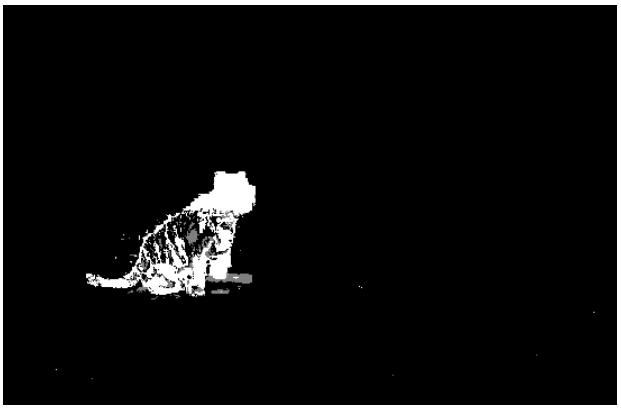

Figura 14 – Imagem do vídeo manipulado tendo como base o vídeo "Tiger Movement", após aplicado "Backgorund subtraction using MOG".

Na figura 15 foi aplicado o detetor de arestas Canny, com os thresholds sendo 150 e 200.

Figura 15 – Imagem do vídeo manipulado tendo como base o vídeo "Tiger Movement", após aplicado "Canny Detector".

Na figura 16 pode ser vista a deteção de cantos no frame, utilizando o detetor de Shi-Tomasi.

Figura 16 – Imagem do vídeo manipulado tendo como base o vídeo "Tiger Movement", após aplicado "Shi-Tomasi corner detection".

F. Overexposed video

Neste vídeo podemos ver um lago, mas o vídeo original está sobre-exposto, pois apresenta demasiado brilho nalguns locais nos frames. Para corrigir isso, foi aplicado o efeito "Equalize histogram" (figura 18)

Figura 16 – Imagem do vídeo original "Overexposed video".

Figura 17 – Imagem do vídeo manipulado tendo como base o vídeo "Overexposed video", após aplicado "Equalize Histogram".

VI. OBSERVAÇÕES E LIMITAÇÕES TÉCNICAS

Nesta secção iremos expor algumas observações sobre a implementação e operacionalidade da aplicação, bem como limitações técnicas.

Algumas aplicações demoram algum tempo a ser aplicadas ao vídeo transformado. Nalguns casos, as transformações aplicadas demoram mais tempo do que o tempo disponível para cada frame. Assim sendo, o vídeo transformado ficará atrasado temporalmente em relação ao vídeo original. Foram efetuadas algumas alterações para evitar que isto aconteça, tal como subtrair o tempo de aplicação de transformações ao atraso aplicado na emissão de frames no reprodutor de vídeo, mas é uma consequência natural de algumas transformações e não pode ser evitado. Ainda foi testado o vídeo transformado "saltar" alguns frames, de modo a estar síncrono com o vídeo original, mas foi decidido que seria prioritária a qualidade do vídeo transformado em relação à sincronia dos dois vídeos.

No caso dos efeitos para filtragem de cores num frame, seria interessante permitir a filtragem por mais do que uma cor (selecionar *Red Image* e *Green Image* e efetuar a soma de ambas as cores). No entanto, para não quebrar a lógica efetuada nas filas de transformações, e para não abrir exceções na arquitetura implementada, tal não é possível.

Por questões de compatibilidade, a aplicação apenas lê vídeos em formato ".avi". Os vídeos utilizados neste relatório podem ser encontrados no link [4].

A aplicação foi desenvolvida em *Visual Studio* utilizando *Qt 5.10.0*. Para executar a aplicação no *Visual Studio*, sendo necessário instalar e configurar o *Qt 5.10, Visual Studio* e *Qt for Windows*.

A integração entre o *Qt* e o *OpenCV* não foi fácil e demorou mais tempo do que o esperado, levando a que o esforço na construção da interface gráfica fosse superior ao esforço de programação e análise dos efeitos. Para além disso, também a construção da fila de efeitos foi demorada, e não está contruída de forma otimizada. Em vez das diversas classes para cada grupo de argumentos diferentes, poderia ter sido construída apenas uma classe com argumentos variáveis e com uma função que recebesse todos os argumentos variáveis. No entanto, por falta de tempo para corrigir alguns dos erros produzidos por essa aproximação, tal não foi possível.

VI. CONTRIBUIÇÃO DE CADA MEMBRO DO GRUPO E IDENTIFICAÇÃO DO TRABALHO EFETUADO

A contribuição de cada membro do grupo, em percentagem, foi a seguinte:

- Diogo Daniel Soares Ferreira – 60%
- Luís Davide Jesus Leira – 40%

VII. CÓDIGO EXTERNO UTILIZADO

A construção desta aplicação teve por base conhecimento adquirido nas aulas, tendo sido utilizado código fornecido através dos guiões práticos das aulas da unidade curricular em que este trabalho se insere.

O código base para a classe *Player* foi retirado do *website* [1] e modificado para as necessidades do projeto.

A função *Mat BrightnessAndContrastAuto(Mat src, float clipHistPercent)*, presente no ficheiro *Aux_functions.cpp*, foi retirada do *website* [2].

Algumas aplicações de funções no ficheiro *Aux_functions.cpp* foram retiradas da documentação da biblioteca *OpenCV* [3] e modificadas de acordo com as necessidades do projeto.

O restante código do projeto pertence aos autores.

VIII. CONCLUSÕES

A realização deste trabalho permitiu integrar diversos conhecimentos aplicados nas aulas, como as operações de *smoothing*, transformações morfológicas (erosão, dilatação, abertura, fecho) ou operadores de deteção de arestas (detetor de *canny*, operador de *sobel*), e também permitiu a integração de efeitos que não foram leccionados nas aulas, mas que são de maior interesse para o tratamento de vídeos, como subtração do *background*, deteção de cantos ou segmentação. Tal levou a que o projeto seja interessante a nível de diversidade e capacidade de manipulação de vídeos.

A aplicação apresenta uma interface gráfica interativa e de fácil utilização para um utilizador inexperiente, ao contrário de muitas aplicações avançadas de tratamento de vídeos, que são demasiado complicadas para um utilizador inexperiente. No entanto, isso também levou a que o tempo despendido na construção e implementação da interface e toda a lógica necessária para o seu bom funcionamento seja relativamente superior ao tempo despendido na programação de efeitos.

Consideramos que a aplicação cumpre o seu propósito de, com uma interface gráfica simples mas usável, demonstrar alguns efeitos que podem ser aplicados a vídeos filmados em tempo-real e carregados a partir de um ficheiro.

REFERENCES

[1] http://codingexodus.blogspot.pt/2012/12/working-with-video-using-opencv-and-qt.html

[2] http://answers.opencv.org/question/75510/how-to-make-auto-adjustmentsbrightness-and-contrast-for-image-android-opencv-image-correction/

[3] https://docs.opencv.org/master/d9/df8/tutorial_root.html

[4] https://mega.nz/#F!XpEDwCrC!pgPod4gqhETkrXCaBDAqdA