XVI CONGRESSO DE PRODUÇÃO CIENTÍFICA E ACADÊMICA

VISÃO COMPUTACIONAL PARA RECUPERAÇÃO DE PARTITURAS NO AMBIENTE MOSAICODE

Frederico Ribeiro Resende, graduando em Ciência da Computação
Flávio Luiz Schiavoni, Departamento de Ciência da Computação

RESUMO

A identificação de objetos e pessoas por meio de computadores é algo que impressiona e indaga ao quão longe podem chegar as aplicações que têm esse papel. A Visão Computacional é a área da computação responsável por desempenhar este papel, interpretando a visão humana frente ao computador. Ela é cada vez mais presente no cotidiano, entretanto, ainda são escassos os meios para criação de aplicações que a utilizem, principalmente para as pessoas que não possuem conhecimento específico para tal. Neste meio é que surge o Mosaicode, uma ferramenta para simplificar a criação de aplicações específicas, incluindo as que utilizam Visão Computacional e Processamento Digital de Imagens. Este projeto resultou na construção de uma extensão para o Mosaicode utilizando a biblioteca OpenCV.

Palavras-chave: Mosaicode. VPL. Arte Digital. Visão Computacional. Processamento de Imagens.

INTRODUÇÃO

A Visão Computacional é uma área da computação que trata de construir uma análise computacional sobre imagens e dados multidimensionais de forma semelhante à visão humana [1]. A detecção de eventos (reconhecimento de movimentos, objetos, etc), o

controle e manipulação de processos industriais e os veículos autônomos são exemplos práticos da utilidade da Visão Computacional no cotidiano das pessoas.

Muitas vezes podemos assemelhar Visão Computacional e Processamento Digital de Imagens, pois ambas possuem um habitat bastante similar e trabalham de forma correlata, onde a primeira é derivada da segunda. Para tal, o Processamento Digital de Imagens trata-se da ciência do melhoramento digital de imagens, para futuras análises humanas e também para pré-processamento de imagens para aplicações de Aprendizado de Máquina e Visão Computacional, dentre outras [2].

Já a Visão Computacional atua performando a visão humana sobre um conjunto de dados multidimensionais, de forma a analisá-los computacionalmente, extraindo informações que são humanamente reconhecíveis e válidas. Neste contexto, o Processamento Digital de Imagens serve como base para que a Visão Computacional possa trabalhar da melhor forma, realizando todo o pré-processamento necessário [3].

A Visão Computacional é a ciência que possibilita a identificação de objetos, textos, pessoas e outros afins em fotos, vídeos e outros meios de representação de dados possíveis. Este reconhecimento se dá através de bases de dados que possuem um padrão em comum, como por exemplo, para identificação de faces em fotos; neste caso, teríamos uma base com as mais variadas faces de forma que a nossa aplicação possa identificar na imagem, padrões semelhantes ao passado à ela. Mais precisamente, isto é trabalho de outra área da computação, conhecida como Aprendizado de Máquina, a ciência que dá aos computadores a possibilidade de aprenderem automaticamente, sem serem programados para tal funcionalidade [4]. Na Figura 1[1], temos um exemplo de aplicação de Visão Computacional no cotidiano.

1 Fonte da imagem:https://aitrends.com/.

Figura 1: Visão Computacional utilizada para detecção de carros, pedestres e ciclistas.

Com estes princípios, podemos concluir que a conjunção entre Processamento Digital de Imagens, Visão Computacional e Aprendizado de Máquina auxiliariam a desempenhar a proposta inicial do projeto, com ênfase na recuperação de partituras antigas, ou que não estejam um tanto quanto aptas à leitura à olho nu. Entretanto, estas ações deveriam se dar dentro do ambiente Mosaicode[2].

O Mosaicode é um ambiente de programação visual que foi bifurcado na base do projeto Harpia, desenvolvido pela S2i - Industrial Intelligent Systems, da UFSC, onde a ideia remetida é que usuários que não conheçam ferramentas de desenvolvimento de linguagens de domínio específico consigam criar suas aplicações através do simples ato de arrastar blocos para gerar código e criar aplicações bem definidas. Desde então, o Mosaicode têm tido como âmbito geral auxiliar artistas, performistas e usuários que não possuam conhecimento específico sobre ferramentas e bibliotecas a criarem suas

2 Website do projeto:http://mosaicode.github.io

próprias aplicações de maneira facilitada, utilizando apenas os princípios da ciência em questão. O ambiente também auxilia no aprendizado de novos estudantes das áreas disponíveis no Mosaicode, pois a ideia de criar aplicações e gerar código permite ao usuário que consiga acompanhar internamente a aplicação que está criando. Dentre as extensões desenvolvidas ou em desenvolvimento encontram-se as de Visão Computacional, Computação Gráfica e Computação Musical.

Existem várias ferramentas e bibliotecas disponíveis para implementação de algoritmos que utilizem princípios das ciências citadas acima. Dentre elas, utilizamos o OpenCV para tal, uma biblioteca open-source idealizada e desenvolvida pela Intel em 1999, que conta com mais de 2500 algoritmos de Análise e Processamento de imagens e Visão Computacional [5].

A proposta deste projeto foi atualizar e inserir plugins para o Mosaicode voltados para Processamento Digital de Imagens e Visão Computacional utilizando a biblioteca OpenCV.

DESENVOLVIMENTO

Este projeto foi iniciado primeiramente com o objetivo de estudar e compreender o domínio de Processamento de Imagens e Visão Computacional, englobando funções e algoritmos desde os mais básicos e fundamentais até a outros mais complexos e sofisticados. Devemos definir primeiramente que se tratando destas ciências, tudo que realizamos são operações em imagens (funções, algoritmos), uma por após a outra, de forma que a sequência em que as operações são realizadas é extremamente importante para o resultado da aplicação final. Considerando isto, a troca de posições na sequência de operações em uma determinada imagem traz a forte possibilidade de que o produto final seja totalmente diferente do anterior.

Em paralelo com o estudo das ciências definidas, também foi-se necessário o rudimento do entendimento do OpenCV, em sua versão 3, para a linguagem C++. Desta

forma, foi possível sincronizar as técnicas estudadas com as práticas realizadas na biblioteca, acelerando todo o processo de aprendizagem.

O funcionamento das operações realizadas sobre as imagens são dadas de forma sequencial, assim como se desejarmos tratar uma imagem antes de aplicar técnicas de Visão Computacional à ela. Neste caso, a entrada pode receber primeiramente todo o pré-processamento necessário através de funções e algoritmos de Processamento Digital de Imagens, mas isto não é impreterível; este estágio ocorre normalmente quando a imagem de entrada não está apta para ser repassada à fase posterior, em termos de qualidade pictórica [6]. Depois desta etapa, caso ela seja realmente necessária, aplica-se técnicas de Visão Computacional e Aprendizado de Máquina, a fim de extrair elementos e informações relevantes da imagem, como rastreamento de pessoas e objetos. Por fim, temos como saída da aplicação a imagem com as devidas informações retiradas ou somente os dados os necessários que foram extraídos da mesma, como é observável na Figura 2.

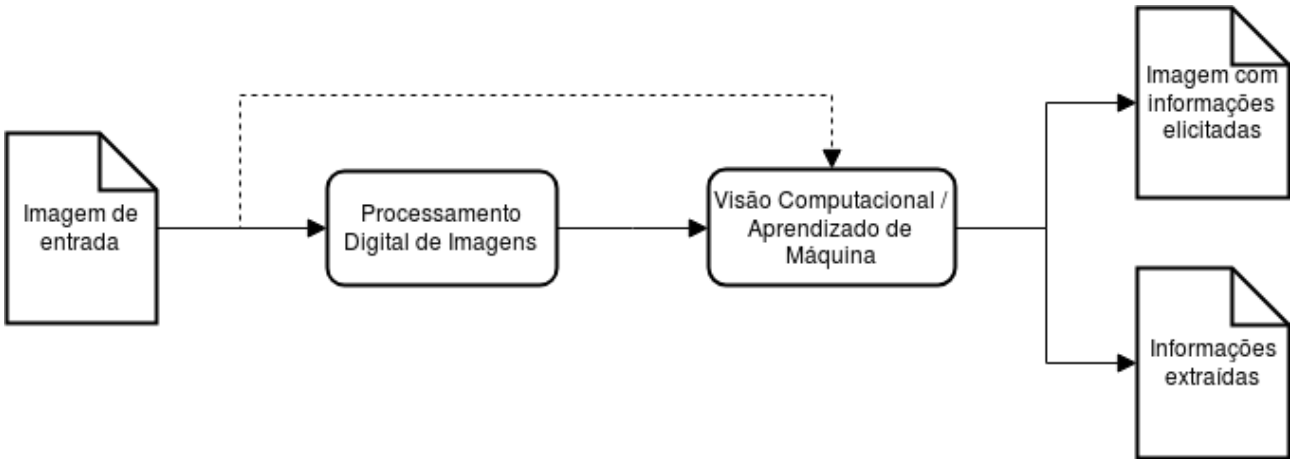

Figura 2: Diagrama sequencial para extração dos dados com Visão Computacional no ambiente Mosaicode.

Foram criados exemplos abordando os diferentes campos do domínio desde os princípios de Processamento e Análise de Imagens até os algoritmos mais básicos em

Visão Computacional como detecção de faces, pessoas e objetos. A criação dos exemplos era justamente voltada para que os pilares das áreas em questão fossem desenvolvidas para o futuro do projeto. Após este passo, tornou-se possível iniciar os primeiros contatos com a ferramenta Mosaicode a fim de retomar o desenvolvimento dos plugins OpenCV.

O desenvolvimento da extensão de Visão Computacional para a ferramenta Mosaicode seria reaproveitada do projeto Harpia, já que o mesmo havia construído anteriormente plugins utilizando a bibliteca OpenCV. Entretanto, o OpenCV assim como toda biblioteca aclamada em seu domínio, evoluiu e sofreu mudanças drásticas, principalmente se considerarmos a alteração da versão 2 para a versão 3, onde passou a utilizar o princípio de orientação à objetos. Logo, em primeira instância, os plugins foram todos remodelados e atualizados da linguagem C para C++, em função da nova versão da biblioteca, deixando para trás a programação estruturada e adquirindo uma forma mais simplificada e eficiente em mesmo modo, como podemos observar na Figura 3.

```
# ------------------C/OpenCv code---------------------------------------
self.codes[1] = \
    'IplImage * block$id$_img_i0 = NULL;\n' + \
    'IplImage * block$id$_img_t0 = NULL;\n' + \
    'IplImage * block$id$_img_t1 = NULL;\n' + \
    'IplImage * block$id$_img_t2 = NULL;\n' + \
    'IplImage * block$id$_img_o0 = NULL;\n' + \
    'IplImage * block$id$_img_o1 = NULL;\n' + \
    'IplImage * block$id$_img_o2 = NULL;\n'

self.codes[2] = \
    '\nif(block$id$_img_i0){\n' + \
    'block$id$_img_o0 = cvCloneImage(block$id$_img_i0);\n' + \
    'block$id$_img_o1 = cvCloneImage(block$id$_img_i0);\n' + \
    'block$id$_img_o2 = cvCloneImage(block$id$_img_i0);\n' + \
    'block$id$_img_t0 = cvCreateImage' + \
    '(cvGetSize(block$id$_img_i0), block$id$_img_i0->depth, 1);\n' + \
    'block$id$_img_t1 = cvCreateImage' + \
    '(cvGetSize(block$id$_img_i0), block$id$_img_i0->depth, 1);\n' +\
    'block$id$_img_t2 = cvCreateImage' + \
    '(cvGetSize(block$id$_img_i0), block$id$_img_i0->depth, 1);\n' + \
    'cvSplit(block$id$_img_i0 ,block$id$_img_t2 ,' + \
    'block$id$_img_t1 ,block$id$_img_t0 , NULL);\n' + \
    'cvMerge(block$id$_img_t0 ,block$id$_img_t0, block$id$_img_t0,' + \
    'NULL, block$id$_img_o0);\n' + \
    'cvMerge(block$id$_img_t1 ,block$id$_img_t1, ' + \
    'block$id$_img_t1, NULL, block$id$_img_o1);\n' + \
    'cvMerge(block$id$_img_t2 ,block$id$_img_t2, ' + \
    'block$id$_img_t2, NULL, block$id$_img_o2);\n}\n'

self.codes[3] = \
    'cvReleaseImage(&block$id$_img_t0);\n' + \
    'cvReleaseImage(&block$id$_img_t1);\n' + \
    'cvReleaseImage(&block$id$_img_t2);\n' + \
    'cvReleaseImage(&block$id$_img_o0);\n' + \
    'cvReleaseImage(&block$id$_img_o1);\n' + \
    'cvReleaseImage(&block$id$_img_o2);\n' + \
    'cvReleaseImage(&block$id$_img_i0);\n'
```

```
# ------------------C/OpenCv code---------------------------
self.codes["declaration"] = \
    'Mat $port[input_image]$;\n' + \
    'Mat block$id$_img_t0[3];\n' + \
    'Mat $port[output_image1]$;\n' + \
    'Mat $port[output_image2]$;\n' + \
    'Mat $port[output_image3]$;\n'

self.codes["execution"] = \
    '\nif(!$port[input_image]$.empty()){\n' + \
    'split($port[input_image]$, block$id$_img_t0);\n' + \
    '$port[output_image1]$ = block$id$_img_t0[0];\n' + \
    '$port[output_image2]$ = block$id$_img_t0[1];\n' + \
    '$port[output_image3]$ = block$id$_img_t0[2];\n}\n'

self.codes["deallocation"] = \
    'block$id$_img_t0[0].release();\n' + \
    'block$id$_img_t0[1].release();\n' + \
    'block$id$_img_t0[2].release();\n' + \
    '$port[output_image1]$.release();\n' + \
    '$port[output_image2]$.release();\n' + \
    '$port[output_image3]$.release();\n' + \
    '$port[input_image]$.release();\n'
```

Figura 3: Evolução do código do plugin de decomposição RGB; do lado esquerdo código na versão 2 da biblioteca e no lado direito na versão 3.

Neste ponto, existiam exatos 43 plugins com técnicas de Processamento de Imagens funcionando para a ferramenta. Dentre eles, existem plugins para operações lógicas, morfológicas e aritméticas de imagens, formas básicas e experimentais, entre outras. Com isto, já era possível realizar praticamente todos os tipos básicos de operações para processamento e análise, e até mesmo para a modificação de imagens e inserção de formas geométricas mais simples. À vista disso, foi-se determinado então que plugins contendo técnicas e algoritmos mais básicos de Visão Computacional fossem inseridos na extensão. Entre eles, podemos citar plugins para detecção de faces, vértices mais nítidos, objetos e outros. Após isto, temos mais de 55 plugins totais na extensão, envolvendo desde técnicas simples de Visão Computacional e Processamento de Imagens até algoritmos experimentais sobre o domínio. Estes plugins estão divididos em 12 categorias, como mostra a Figura 4.

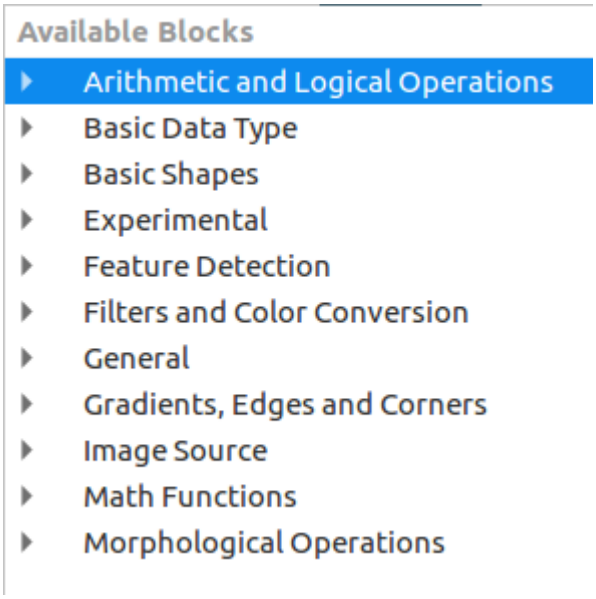

Figura 4: Plugins da extensão de OpenCV para o Mosaicode.

Voltando ao ponto em que abordamos sobre a sequencialidade das operações sobre as imagens, é de se ressaltar a facilidade em que o Mosaicode trás para aqueles que não possuem um objetivo bem definido no momento de criar uma aplicação, fornecendo a possibilidade de alterar as aplicações simplesmente por trocar a ordem em que os blocos estão dispostos nos diagramas. Logo, o Mosaicode torna-se também uma ferramenta de experimentação e recriação facilitada, como podemos visualizar na Figura 5.

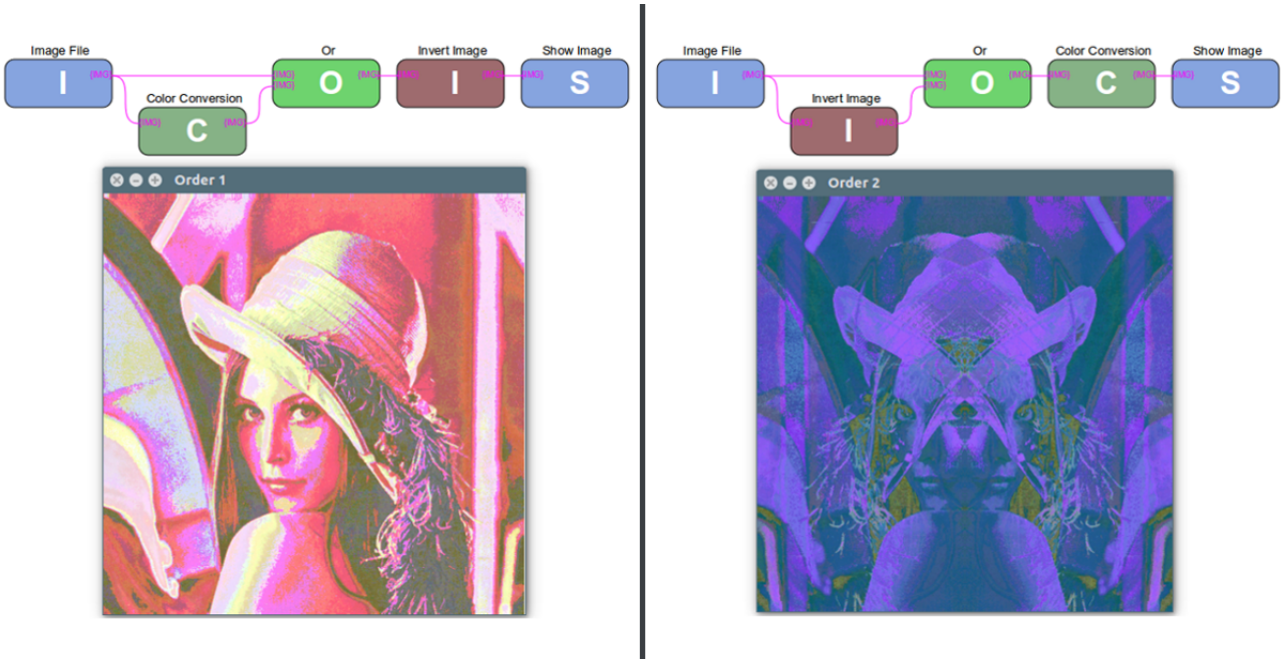

Figura 5: Exemplo de um diagrama disposto em sequências distintas no Mosaicode e os resultados de sua execução.

Para passos futuros, deve ser continuado a projeção e criação de mais plugins envolvendo técnicas mais avançadas e inovadoras de Visão Computacional, incluindo aqueles que possam ser úteis à peças e apresentações artísticas, isto em interação com síntese de imagens, som e outras áreas que possam gerar recursos inovadores.

RESULTADOS

Inicialmente, o projeto era focado exclusivamente no desenvolvimento da solução do problema de recuperação de partituras, entretanto, este objetivo foi reavaliado e obteve uma menor exclusividade momentânea, já que a necessidade de realizar a manutenção e criar novos plugins para a extensão do OpenCV foi mais urgente. Devemos considerar também que o problema da recuperação de partituras deverá ser trabalhado futuramente através da extensão e não desenvolvendo aplicações externas sobre ela, comprovando a necessidade de trabalhar primeiramente na extensão.

Com a atualização dos plugins para a nova versão da biblioteca OpenCV, a extensão obteve sua primeira versão totalmente funcional dentro do ambiente Mosaicode. Logo, ela foi encapsulada para distribuição pelo repositório do projeto na plataforma GitHub e também através do empacotamento via Pypi[3], um repositório de terceiros oficial para a linguagem Python. Na sequência, os novos plugins de Visão Computacional foram inseridos e prontamente disponibilizados na extensão.

Futuramente, um trabalho de testes na usabilidade da extensão na ferramenta poderá ser executado visando qualificar o funcionamento da extensão [7]. Além disto, considerando que a extensão OpenCV agora está totalmente funcional, podemos nos concentrar na idealização de algoritmos específicos de Visão Computacional para tratar a proposta inicial do projeto para recuperação de partituras com o Mosaicode. Desta forma, é necessário elicitar as principais funções e algoritmos a serem encapsulados e dispostos na ferramenta para que tal funcionalidade de recuperação seja possível. Possivelmente, a extensão contará com uma gama ampla de plugins de Visão Computacional, assim como já possui para Processamento Digital de Imagens.

CONCLUSÃO

Com o intuito principal de fornecer auxílio aos artistas e performistas que não possuem conhecimento específico suficiente para criarem suas peças, o Mosaicode tenta trazer consigo variadas extensões representando diferentes áreas como Síntese de Imagens, Som e Visão Computacional. Mais especificamente sobre a Visão Computacional, podemos afirmar que a atualização dos blocos fez com que a extensão finalmente estivesse à pronto uso, totalmente funcional.

3 Website do projeto: https://pypi.org/,

Desta forma, artistas que utilizarem a ferramenta poderão utilizar os plugins já existentes para detecção de pessoas e objetos da forma que possam controlar e gerar formas e sons a partir disto, considerando que futuramente novos plugins serão construídos e disponibilizados. Extraindo outra vantagem da ferramenta, podemos elicitar a didática da mesma, onde alunos que não possuem conhecimento específico de Visão Computacional e Processamento Digital de Imagens podem aprender elaborando diagramas experimentais dentro da ferramenta e gerando o código da aplicação construída, aumentando o conhecimento construído pelo aluno.

A utilização da biblioteca OpenCV em sua mais recente versão facilitou a contrução dos novos plugins justamente por oferecer uma grande gama de algoritmos a serem utilizados de forma eficiente, e reduzindo em grande escala a quantidade de linhas de código dada sua última versão. Desta forma, facilita-se o aprendizado dos usuários que desejam utilizar a ferramenta e a extensão justamente para geração de código.

Além de dar continuidade à construção de plugins para Visão Computacional, é relevante o interesse na mesclagem do OpenCV com outras bibliotecas e APIs de linguagem específica, como o OpenGL [8] para síntese de imagens ou o MIDI [9], para o uso de um controlador físico. Ambas estas possibilidades poderão ser de grande proveito pela comunidade artística.

AGRADECIMENTOS

Os autores agradecem o auxílio dado pela Universidade Federal de São João del-Rei por meio de sua bolsa institucional de Iniciação Científica.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Maurício Marengoni and Stringhini Stringhini. Tutorial: Introdução à visão computacional usando opencv. *Revista de Informática Teórica e Aplicada*, 16(1):125–160, 2009.

[2] Ogê Marques Filho and Hugo Vieira Neto. *Processamento digital de imagens.* Brasport, 1999.

[3] Antonio Escaño Scuri. Fundamentos da imagem digital. *Pontifícia Universidade Católica do Rio de Janeiro,* 1999.

[4] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging,* 16(4):049901, 2007.

[5] Cleiton Goulart, André Persechino, Marcelo Albuquerque, and Márcio Albuquerque. Introdução à biblioteca de processamento de imagens opencv. *NOTAS TÉCNICAS*, 8(2), 2018.

[6] José Eustáquio Rangel de Queiroz and Herman Martins Gomes. Introdução ao processamento digital de imagens. *RITA*, 13(2):11–42, 2006.

[7] Flávio Luiz Schiavoni and Luan Luiz Gonçalves. Teste de usabilidade do sistema mosaicode. *In Anais [do] IV Workshop de Iniciação Científica em Sistemas de Informação (WICSI),* pages 5–8, Lavras - MG - Brazil, 2017.

[8] Francis S Hill and Stephen M Kelley. *Computer graphics: using OpenGL*, volume 2. Prentice Hall Upper Saddle River, NJ, 2001.

[9] Craig Anderton. The midi protocol. *In Audio Engineering Society Conference: 5th International Conference: Music and Digital Technology.* Audio Engineering Society, 1987.