Classificação de Mosquitos Aedes aegypti com Template Matching

Artur Wiederkehr (artur.wiederkehr@edu.pucrs.br)

Euzébio Henzel Antunes (euzebio.antunes@edu.pucrs.br)

Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Telefone (51) 3320-3558 – 90619-900 – Porto Alegre – RS – Brasil.

***Abstract**. This article describes the process of applying certain knowledge acquired in the Introduction to Computer Vision discipline to classify mosquitoes of the species Aedes aegypti from mosquitoes of other species using images. A set of own images was created together with an image processing algorithm to obtain a satisfactory result, all these elements will be presented below*

***Resumo.** Este artigo descreve o processo de aplicação de determinados conhecimentos adquiridos na disciplina de Introdução a Visão Computacional para a classificação de mosquitos da espécie Aedes aegypti de mosquitos de outras espécies através de imagens. Um conjunto de imagens próprio foi confeccionado junto de um algoritmo de processamento de imagens para a obtenção de um resultado satisfatório, todos esses elementos serão apresentados a seguir.*

1. Introdução

Através deste relatório será apresentado o procedimento adotado durante a realização do trabalho, ressaltando os detalhes mais importantes, que possibilitaram a obtenção de um resultado satisfatório na tarefa de classificar os mosquitos das imagens constantes no conjunto de imagens, em espécie *Aedes aegypti* ou imagens relativas a outras espécies.

Faz-se mister ressaltar que foram feitos dois programas com o desiderato de resolver o problema em análise. Primeiramente, será analisado o algoritmo que leva em conta a área, a borda, as cores das imagens e que foi falha a classificação referente ao escopo desejado. Uma segunda, usando um *template match* como forma de classificar os mosquitos constantes no conjunto em mosquitos da dengue ou outros mosquitos.

2 Desenvolvimento

A seguir, a explicação do *pipeline* da aplicação junto do desenvolvimento de cada etapa que o constitui, no que concerne as duas abordagens feitas:

2.1 Algoritmo que leva em conta a área, cor e bordas das imagens

Este código realiza a análise de imagens de mosquitos para a detecção de mosquitos da dengue utilizando a biblioteca OpenCV . Cabe referir que, no caso em tela, tem o algoritmo o escopo de identificar os mosquitos Aedes aegypti, através da detecção de bolinhas brancas, tendo em vista que eles têm tais marcas por toda extensão do corpo.

Em apertada síntese, no que tange ao funcionamento do código, a biblioteca OpenCV é usada para processar imagens, e epsilon é um parâmetro usado na detecção dos contornos , bem como threshold é um parâmetro usado para a detecção das bordas. A função detectar bolinhas brancas recebe uma imagem como entrada e tem como objetivo detectar uma bolinha branca na imagem, com bordas pretas. Converte a imagem para escala de cinza, aplica um desfoque gaussiano e utiliza o algoritmo Canny para detecção de bordas. Em seguida, procura por contornos na imagem, verificando se cada contorno atende aos critérios específicos de área e número de vértices. Se encontrar um contorno que corresponda a esses critérios, desenha um retângulo verde ao redor.

A função cv2.contourArea é usada para calcular a área de um contorno, que é uma região fechada na imagem. Os contornos são obtidos a partir da detecção de bordas na imagem (área que queremos calcular). A função retorna um valor que representa a área do contorno. Essa área é posteriormente usada como um critério para determinar se o contorno representa uma bolinha branca com borda preta. A função cv2.approxPolyDP é usada para realizar uma simplificação aproximada de um contorno poligonal. Ela tenta reduzir o número de vértices em um contorno, aproximando-o por um polígono com menos vértices. A função retorna o contorno simplificado, que é uma lista de pontos representando o novo contorno com menos vértices. É usada para simplificar o contorno detectado na imagem original. O valor de epsilon é multiplicado pelo comprimento do contorno original (cv2.arcLength(contorno, True)) para determinar o quão simplificado o contorno resultante será. A simplificação é usada para verificar o número de vértices no condicional `if 6 <= len(approx) <= 500`, onde se decide se o contorno representa uma bolinha branca com bordas pretas com base na quantidade de vértices.

A função cv2.arcLength é usada para calcular o comprimento de um contorno (curva) ou uma sequência de pontos em um contorno. A função retorna o comprimento do contorno. Quanto maior o comprimento do contorno, maior será a precisão necessária para a simplificação. É uma função fundamental para o processamento de contornos em imagens e são usadas no código para identificar bolinhas brancas com bordas pretas nas imagens de mosquitos.

É calculado o número de acertos com base no número de imagens de mosquitos detectadas em relação ao número total de imagens processadas. O resultado obtido foi uma taxa de acerto de 32,35%. De um total de 34 imagens, ele acertou 11 como sendo o mosquito transmissor da dengue (*Aedes aegypti*).

2.2 Algoritmo que analisa as imagens através de um template match

A seguir discorremos sobre o método *Template Match,* onde abordamos sobre a base de conhecimento, aquisição de imagens, pré-processamento, representação, descrição, reconhecimento e interpretação. A figura 1 ilustra o *pipeline* adotado para a confecção da solução.

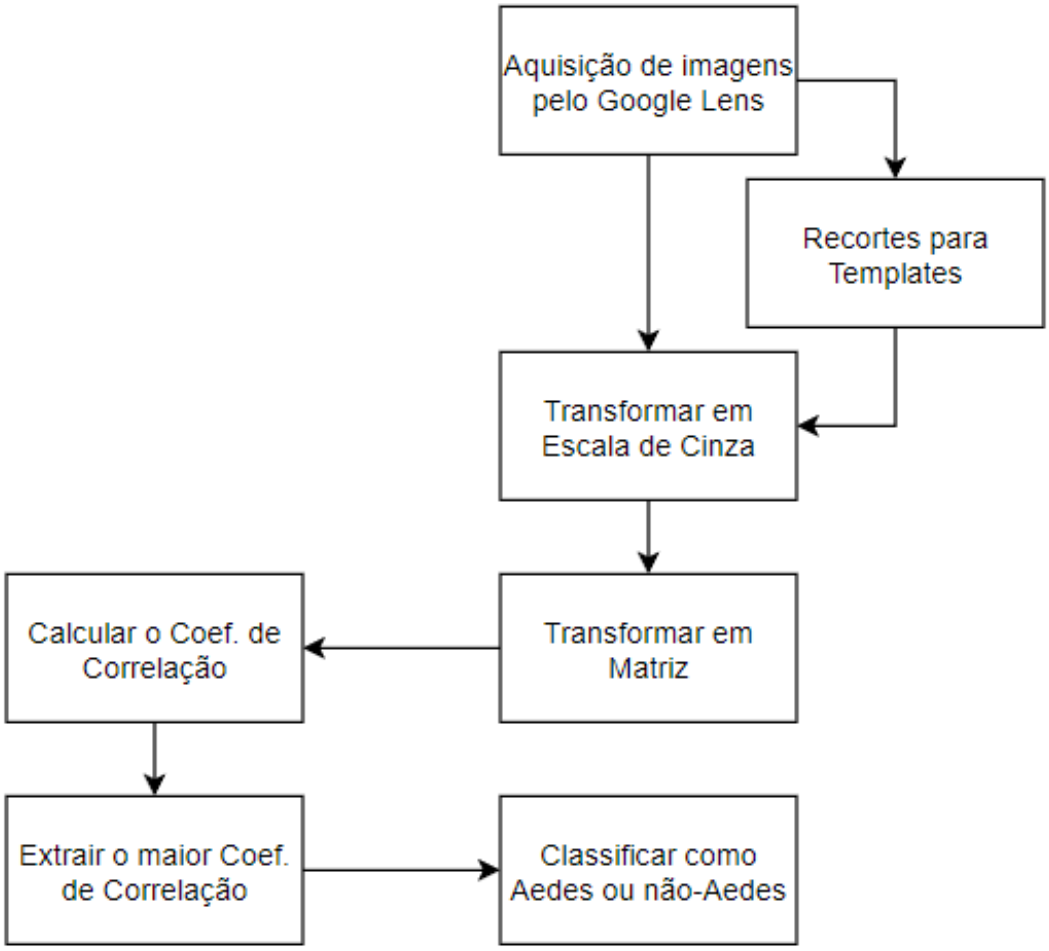


Figura 1 – Pipeline executado para a solução.

2.2.1 Base de Conhecimento

No problema abordado, as imagens devem ser acessadas pelo computador, processadas e os mosquitos presentes nelas devem ser classificados como sendo ou não da espécie *Aedes aegypti*. A biblioteca OpenCV oferece os recursos necessários para a resolução deste problema, o nosso maior trabalho é como utilizaremos estes recursos a fim de obtermos um bom resultado.

2.2.1.1 Aquisição de Imagens

As imagens dos mosquitos foram adquiridas através de buscas no Google, elas foram separadas em duas pastas para que possamos ter controle sobre qual classe cada imagem possui (se é ou não da espécie *Aedes aegypti*). A figura 2 mostra um exemplo de imagem de *Aedes aegypti* e de um mosquito de outra espécie:

Figura 2 – *Aedes aegypti* (à esquerda) e mosquito de outra espécie (à direita).

2.2.1.1.1 Pré-Processamento

Nesta etapa cada imagem foi convertida para nível de cinza com o propósito de reduzir a dimensão dos dados, reduzir ruídos e tornar o modelo menos sensível a variações de cor.

2.2.1.1.2 Representação e Descrição

Uma vez em preto e branco, a imagem é transformada em uma matriz em que cada número representa um valor de intensidade luminosa naquele ponto específico.

Foram selecionadas algumas imagens do conjunto para que fossem observadas características em comum dos mosquitos *Aedes aegypti*, então foram extraídas, destas imagens, pequenas partes em que se acreditava serem boas candidatas a repetirem seu padrão em outras imagens do mosquito *Aedes aegypti*, é válido ressaltar que este foi um processo manual. As pequenas frações de imagem foram utilizadas para o método *Template Matching* portanto passaram pela fase de pré-processamento também.

A tabela 1 apresenta as imagens usadas como *templates* no processo de classificação:

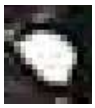
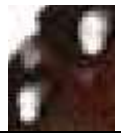
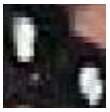
Template 1	Template 2	Template 3	Template 4	Template 5
				

Tabela 1 – Templates utilizados

2.2.1.1.3 Reconhecimento e Interpretação

Cada imagem foi processada pela função *matchTemplate* cujo um dos parâmetros é o "TM_CCOEFF" responsável por calcular o coeficiente de correlação entre a imagem de origem e o ponto de referência: quanto maior o valor resultante, maior a correspondência (mais parecida a imagem é naquela coordenada), outro parâmetro é o *template,* dos 5 mostrados da tabela acima prosseguimos apenas com o *Template 1*.

Agora possuímos uma matriz com valores que representam o quanto o *template* é similar com cada pixel, utiliza-se o método *minMaxLoc* para identificar o maior valor contido na matriz e suas respectivas coordenadas, este é o local em que o template possui a maior correspondência.

Ao ler os maiores valores que cada imagem possui, observa-se que este oscilam bastante, mas há uma predominância entre 300.000 e 400.000, então a comparação para classificação é um processo de tentativa e calibragem: é necessário achar um valor que separe mosquitos Aedes dos demais, para isso foram testados os valores 400.000 e 350.000, observou-se o quanto o modelo se tornou mais flexível, em seguida foram testados os valores múltiplos de 10.000 entre estes dois últimos e notou-se que 360.000 e 390.000 foram valores que tinham bons resultados: o primeiro maximizava os verdadeiros positivos ao custo de um pouco mais de falsos positivos, enquanto o segundo minimizava drasticamente os falsos positivos e gerava menos verdadeiros positivos, testando os múltiplos de milhar um pouco abaixo de cada um constatou-se que os melhores candidatos para as comparações no nosso modelo são os valores 358.000 e 388.000. Como possuímos as coordenadas em que ocorre a maior similaridade, desenha-se um retângulo no local para nos certificarmos que capturamos uma área de interesse corretamente. Observe a figura 3 em que dois *Aedes aegypti* foram classificados corretamente:

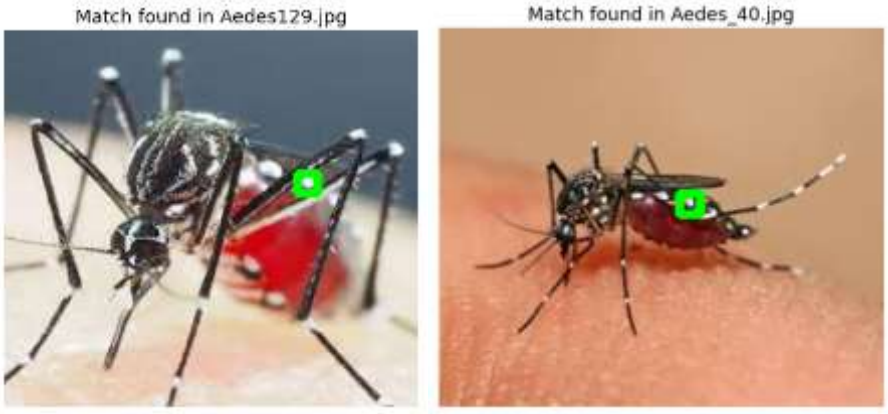


Figura 3 – *Aedes aegypti* classificados corretamente.

2.2.2 Algoritmo de Template Matching

```
template = abrir(template)
templateCinza = converter(template)
templateCinzaMatriz = matriz(templateCinza)
Para cada Imagem em Diretório:
imagemCinza = converter(Imagem)
imagemCinzaMatriz = matriz(imagemCinza)
resultado = matchTemplate(imagemCinzaMatriz, templateCinzaMatriz, TM_CCOEFF)
_, maxValor, _, maxLocal = minMaxLocal(resultado)
        Se maxValor > 358.000:
                Imagem = retangulo(maxLocal, bordaDebaixo)
                mostrar(Imagem)
```

3. Resultado

O algoritmo de classificação processou 34 imagens de Aedes e 44 de outras espécies, então obteve-se os seguintes resultados apresentados na tabela 2:

Valor de Comparação: 358.000			
		Valor Predito	
		Aedes	Outro
Valor Real	Aedes	32	2
	Outro	4	39

Valor de Comparação: 388.000			
		Valor Predito	
		Aedes	Outro
Valor Real	Aedes	28	6
	Outro	1	43

Tabela 2 – Matrizes de Confusão do algoritmo

O valor de comparação 358.000 obteve **92,208% de acurácia**, enquanto o valor 388.000 resultou em **91,026% de acurácia** na classificação de mosquitos *Aedes aegypti* de mosquitos de outras espécies. No problema abordado, não há grande impacto negativo em classificar incorretamente um mosquito não-Aedes como Aedes (Falso Positivo), entretanto pode ser prejudicial classificar um Aedes como não-Aedes (Falso Negativo), portanto optou-se pelo valor 358.000 como melhor valor para o problema. Utilizando o valor 358.000 o modelo atinge **94,117% de precisão.**

Os erros cometidos pelo modelo têm origem na comparação com o limite de correspondência, algumas imagens de mosquitos não-Aedes possuem partes em que sua correspondência é muito alta graças ao padrão que o *template* impõe: um círculo grosseiro preenchido com valor de intensidade luminosa alto (branco) cercado de pixels de valor

baixo de intensidade luminosa (preto) e nem todas as imagens de Aedes estão aptas a apresentarem este padrão, em seguida explicamos especificamente cada caso.

Os falsos negativos (Aedes classificado como outra espécie) ocorrem normalmente quando o mosquito na imagem está borrado ou pequeno em proporção à imagem, como demonstrado na figura 4, então o *template* não é considerado similar:

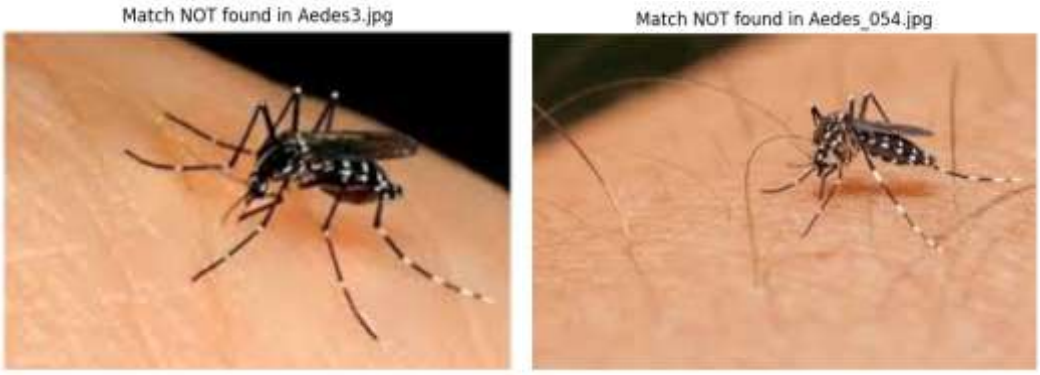


Figura 4 – Aedes classificados incorretamente

Os falsos positivos (outra espécie classificada como Aedes) ocorrem quando uma parte do corpo do mosquito se aproxima demais de outra parte e forma um ponto (relativamente circular) de tonalidade clara, veja a figura 5:

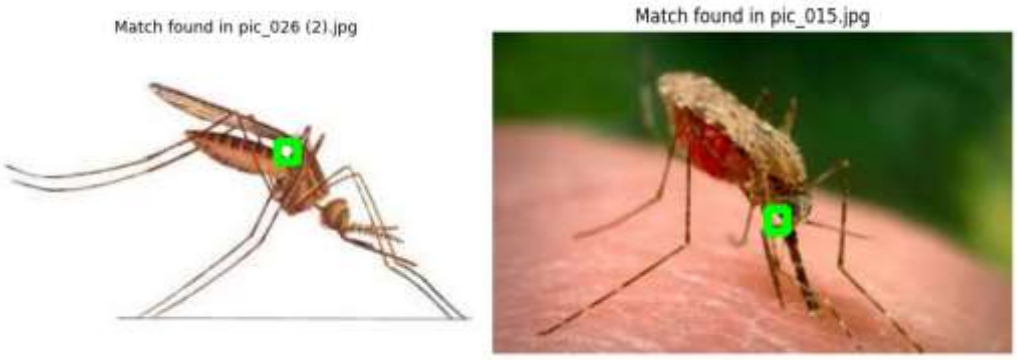


Figura 5 – Mosquitos não-Aedes classificados incorretamente pela posição do corpo.

Outro fator que causa falsos positivos é o Mosquito Culex, que será abordado dentro do próximo tópico.

4. Limitações

As imagens processadas pelo algoritmo não possuem alta resolução, a maior não chega a 300 x 200 pixels, entretanto o mosquito presente em cada imagem possui boa definição (ele e seus detalhes são nítidos). O fundo das imagens não interferiu no resultado, há imagens de fundo branco, fundo verde e com o mosquito em contato com a pele de uma pessoa.

Uma alteração no pré-processamento, no *template* ou no valor de limite de correspondência pode gerar um equívoco por parte do algoritmo, estes fatores dependem um do outro, ao alterar um os outros devem ser modificados também.

4.1 Nosso Maior Rival: o Mosquito Culex

Há uma espécie de mosquito que tem semelhanças com o *Aedes aegypti* chamada de Culex (*Culex pipiens*), como mostrado na figura 6.

Figura 6 – Mosquito Culex classificado incorretamente.

O modelo busca por uma área circular clara com contorno escuro, como este padrão aparece no mosquito Culex, a comparação terá alto grau de semelhança e resultará na classificação incorreta de mosquitos Culex como *Aedes aegypti*, como dito anteriormente, uma segunda tentativa de classificação poderia melhorar o modelo para lidar com esse problema.

5. Considerações Finais

Este artigo descreve duas abordagens diferentes para a classificação de mosquitos da espécie *Aedes aegypti* a partir de imagens, utilizando conhecimentos adquiridos na disciplina de Introdução à Visão Computacional. A abordagem que utilizou o método *template match* mostrou um resultado promissor alcançando acima 92% de acurácia, contudo com limitações a serem consideradas e melhoradas: a dificuldade em classificar corretamente um mosquito Aedes que possui baixa resolução, a classificação incorreta baseada na posição do mosquito e o padrão do mosquito Culex similar ao Aedes.

Acredita-se que uma segunda tentativa de classificação com outra abordagem para corrigir os equívocos melhoraria a acurácia do modelo ao preço de um pouco mais de processamento. Em resumo, o artigo demonstra a aplicação de conhecimentos de Visão Computacional, para a classificação de mosquitos *Aedes aegypti* a partir de imagens, apresentando duas abordagens, uma com resultado promissor. No entanto, cabe salientar a importância de continuar aprimorando o modelo, considerando as limitações e desafios específicos do problema.

6. Referências

1. Richard Szeliski. Computer Vision: Algorithms and Applications. Springer, 2011.

2. Adrian Kaehler, Gary Rost Bradski. Learning OpenCV 3. O'Reilly Media, 1 ed., 2017.