DESENVOLVIMENTO DE APLICATIVO DE PROCESSAMENTO DE IMAGENS EM TEMPO REAL PARA DISPOSITIVOS MÓVEIS

Ana Karolina Ferraro[1], Tiago Alexandre Dócusse[1]
[1]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo – *Campus* Barretos.
ana.ferraro@aluno.ifsp.edu.br
Palavras-chave: Processamento de imagens, Aplicativos para dispositivos móveis, Acessibilidade.

Introdução

O rastreamento ocular é uma tecnologia de visão computacional que permite identificar a direção do olhar e os pontos de fixação de um usuário em uma tela. Essa técnica tem sido amplamente estudada em contextos de interação humano-computador e em tecnologias assistivas, com destaque para aplicações voltadas à acessibilidade e à comunicação alternativa (Reed, 2018). Estudos recentes demonstram que é possível realizar rastreamento ocular em dispositivos móveis utilizando apenas câmeras convencionais, sem a necessidade de sensores infravermelhos ou *hardware* especializado (Arakawa et al., 2022; Chhimpa et al., 2024).

Embora existam tecnologias avançadas de rastreamento ocular, ainda existem barreiras de acesso, como altos custos para aquisição de câmeras externas e complexidade de implementação. O desenvolvimento de algoritmos gratuitos e de fácil utilização para dispositivos móveis tem o potencial de ampliar a acessibilidade, permitindo que mais pessoas, incluindo as com deficiência, se beneficiem dessas soluções (Liu et al., 2025).

Dentro desse contexto, o presente trabalho, em desenvolvimento no Instituto Federal de São Paulo, *campus* Barretos, apresenta, em uma etapa inicial, a implementação de operações básicas de processamento de imagens em dispositivos móveis, através do processamento, em tempo real, da imagem capturada pela câmera destes dispositivos.

O processamento de imagens é uma área da computação responsável pelo processamento de imagens digitais, capturadas por meio de sensores (normalmente câmeras) e representadas computacionalmente como uma matriz de números inteiros, em que cada um representa um ponto da imagem capturada, denominados *pixels.*

O desenvolvimento do aplicativo apresentado neste trabalho é uma etapa importante de um projeto maior, pois servirá como base para o restante do mesmo, tanto em termos de código desenvolvido como do aprendizado obtido durante seu desenvolvimento.

Objetivos

O objetivo deste trabalho foi criar um aplicativo para dispositivos móveis, capaz de processar imagens capturadas por câmeras desses dispositivos em tempo real, de forma a servir como base para o restante do projeto ao qual este pertence.

Material e Métodos

O aplicativo apresentado neste trabalho foi desenvolvido utilizando o *framework* Expo 53 (650 Industries, 2025a) do ambiente de programação para dispositivos móveis React Native (Meta Platforms, 2025). O React Native é uma biblioteca de código aberto que permite desenvolver aplicações móveis utilizando a linguagem de programação Javascript, possibilitando a criação de aplicativos para dispositivos com os sistemas operacionais Android ou iOS. Embora o aplicativo seja compatível com iOS, os testes foram realizados apenas em sistemas Android, devido à ausência de um ambiente Macintosh para execução e depuração em dispositivos da Apple por parte dos autores.

O aplicativo foi projetado para capturar imagens da câmera frontal ou traseira do dispositivo e aplicar operações básicas de processamento de imagens localmente no

aparelho, em tempo real, armazenando o resultado do processamento quando selecionado pelo usuário do aplicativo. Como a câmera desses dispositivos captura imagens em cores, e as informações de cores não são importantes para este projeto, foi utilizado o modelo de cores YUV, em que a imagem é separada nos seus componentes de luminosidade (Y), correção para o azul (U) e correção para o vermelho (V). Como não foram utilizados os valores de cores, as componentes U e V são descartadas, e foram utilizados apenas os valores da componente Y, obtendo uma imagem monocromática, ou em tons de cinza. Dessa forma, não é necessário fazer processamento sobre a imagem, para converter de um modelo de cores para outro, como no caso do modelo de cores *Red Blue Green* (RGB) ou *Hue Saturation Value* (HSV) (Gonzalez e Woods, 2010).

Seja I a imagem monocromática capturada pelo aplicativo através da câmera do dispositivo móvel, são aplicadas as operações de binarização (B) e limiarização (L) nela, dadas pelas seguintes equações:

$$B(x,y) = \begin{cases} 0, \text{ se } I(x,y) \leq \text{limiar} \\ 255, \text{se } I(x,y) > limiar \end{cases}$$

$$L(x,y) = \begin{cases} 0, \text{ se } I(x,y) \leq \text{limiar} \\ I(x,y), \text{se } I(x,y) > limiar \end{cases}$$

, em que (x, y) representa a localização do *pixel* na imagem correspondente na linha *x* e coluna *y*, 0 representa a cor preta, 255 representa a cor branca ao utilizar 8 *bits* para representar as cores, e limiar é um valor intermediário utilizado na operação correspondente.

Para poder capturar a imagem e realizar seu processamento, foi utilizada a biblioteca React Native Vision Camera (Rousavy, 2025), que permite o acesso ao fluxo de vídeo da câmera do dispositivo e a execução de rotinas em tempo real por meio de um recurso chamado *Frame Processor*. Esse recurso é executado em um *thread* separado, independente da interface gráfica, o que possibilita a captura das imagens da câmera sem comprometer o desempenho do *thread* principal do aplicativo. Ele realiza a leitura de cada quadro capturado, e envia essas informações à *thread* principal do aplicativo, que executa em Javascript, onde as imagens são armazenadas após o processamento. O armazenamento é feito utilizando imagens em formato *Portable Network Graphics* (PNG) no armazenamento local do dispositivo, utilizando a biblioteca PNGJS (GitHub, 2025) para transformar os dados em uma imagem armazenável no sistema de arquivos do dispositivo móvel, e a biblioteca Expo File System (650 Industries, 2025b) para acessar o sistema de arquivos do dispositivo móvel.

Resultados e Discussão

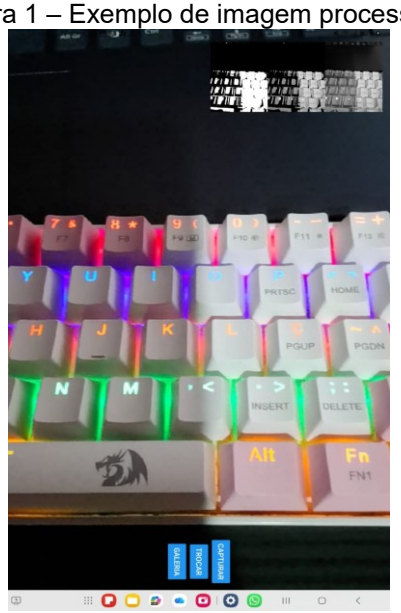
O processamento das imagens capturadas pelos autores foi realizado utilizando imagens com tamanho de 640 × 480 *pixels*. No entanto, processar 60 quadros por segundo de imagens nesse tamanho se mostrou muito custoso computacionalmente, ocasionando atrasos e lentidão no dispositivo de testes utilizado. Dessa forma, a imagem foi redimensionada para a resolução de 320 x 240 *pixels*, permitindo o seu processamento sem perceptíveis atrasos. No entanto, não foram calculados os tempos de processamento de cada imagem, o que deverá ser feito numa próxima etapa do projeto.

Na Figura 1 é exibido um exemplo da utilização do aplicativo. Nela é possível visualizar a imagem capturada pela câmera do dispositivo, exibida em tela cheia, e no canto superior direito as três imagens armazenadas ao pressionar o botão “Capturar” presente na tela do aplicativo. Essas imagens são, em ordem da esquerda para a direita, a imagem resultante da operação de binarização, a imagem resultante da operação de limiarização e a imagem transformada para tons de cinza. Para ambas as operações de binarização e limiarização, foi utilizado o valor 127 como valor de limiar, por ser o valor intermediário entre o 0 (preto) e o 255 (branco).

Na Figura 2 é exibida a galeria do aplicativo, onde as imagens processadas e armazenadas são exibidas, para possibilitar uma melhor visualização. Nessa figura, são exibidas de cima para baixo a imagem resultante da operação de binarização, a imagem

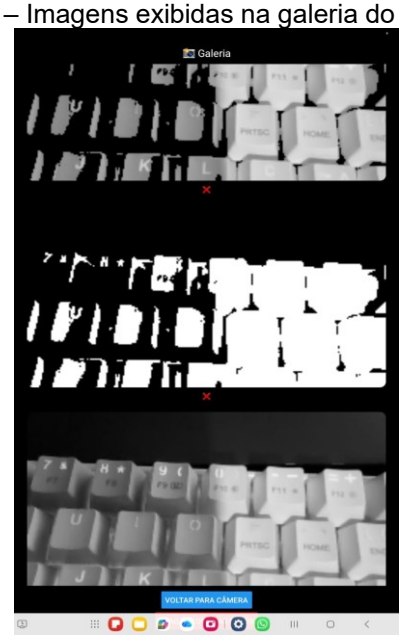
resultante da operação de limiarização, e a imagem resultante da transformação para imagem em tons de cinza. Em ambas as operações, foi utilizado o valor de 127 como limiar.

Figura 1 – Exemplo de imagem processada



Fonte: criação própria

Figura 2 – Imagens exibidas na galeria do aplicativo



Fonte: criação própria

O aplicativo desenvolvido foi capaz de realizar o processamento de imagens com sucesso, atendendo ao objetivo deste trabalho. As imagens processadas foram corretamente capturadas e armazenadas no dispositivo móvel, demonstrando a viabilidade das operações implementadas nesta etapa do projeto. No entanto, neste momento não é possível afirmar se a diminuição na resolução da imagem irá afetar o objetivo final do projeto, que é o de realizar o rastreamento ocular na imagem capturada pela câmera. Essa restrição será trabalhada em um futuro desenvolvimento do aplicativo, com a implementação do *Frame Processor* em código nativo, de acordo com o sistema operacional do dispositivo móvel utilizado, ao invés de em Javascript, como feito neste trabalho. Esta implementação garante a execução do código sem a necessidade de interpretação e comunicação adicional entre *threads* de linguagens diferentes, o que permitirá o processamento de imagens com resoluções maiores, sem que o aplicativo apresente atrasos perceptíveis ao usuário.

Conclusões

Os resultados obtidos até o momento mostram que foi possível criar um aplicativo que realize a implementação de operações básicas de processamento de imagens utilizando a câmera de dispositivos móveis. O aplicativo foi implementado com sucesso e servirá como base para o restante do projeto, sendo necessária a implementação dessas operações em código nativo para um melhor desempenho do mesmo.

Como continuidade do projeto, pretende-se realizar a validação de algoritmos de rastreamento ocular, implementados com o *Frame Processor* após a realização de um pré-processamento das imagens da câmera utilizando operações de processamento de imagens, entre elas as discutidas neste artigo, buscando realizar o rastreamento ocular em tempo real, sem a necessidade de câmeras adicionais ou *softwares* pagos, contribuindo para o avanço de tecnologias assistivas acessíveis, portáteis e de baixo custo, alinhadas ao compromisso social com a inovação e a inclusão digital.

Agradecimentos

Os autores gostariam de agradecer ao Programa de Apoio à Ciência e Tecnologia (PACTec) do Instituto Federal de São Paulo, pelo apoio financeiro recebido.

Referências Bibliográficas

650 Industries, Inc. **Expo**, 2025. Disponível em: <https://expo.dev>. Acesso em: 27 out. 2025a.

650 Industries, Inc. **FileSystem - Expo Documentation**, 2025. Disponível em: <https://docs.expo.dev/versions/v53.0.0/sdk/filesystem/>. Acesso em: 27 out. 2025b.

Arakawa, R.; et al.*: Gaze Tracking on Smartphones with RGB and Depth Data*. In: **International Conference on Multimodal Interaction (ICMI)**, 2022, Bengaluru, Índia. ACM, 2022. p. 329–336. DOI: 10.1145/3536221.3556568. Disponível em: https://dl.acm.org/doi/10.1145/3536221.3556568. Acesso em: 27 out. 2025.

Chhimpa, G. R.; et al. *Revolutionizing Gaze-Based Human–Computer Interaction Using Iris Tracking*. **IEEE Access**, v. 12, p. 168256 - 168269, 2024. Disponível em: https://ieeexplore.ieee.org. Acesso em: 27 out. 2025.

GitHub, Inc. **GitHub - pngjs/pngjs: Simple PNG encoder/decoder**, 2025. Disponível em: <https://github.com/pngjs/pngjs>. Acesso em: 27 out. 2025.

Gonzalez, R.C.; Woods, R. E. **Processamento digital de imagens**. 3. ed. São Paulo: Pearson Prentice Hall, 2010. 624 p.

Liu, C.; et al. *Enhancing Smartphone Eye Tracking with Cursor-Based Interactive Implicit Calibration*. In: **CHI Conference on Human Factors in Computing Systems**, 2025. ACM, 2025. DOI: 10.1145/3706598.3713936. Disponível em: https://dl.acm.org/doi/10.1145/3706598.3713936. Acesso em: 27 out. 2025.

Meta Platforms, Inc. **React Native - Learn Once, write anywhere**, 2025. Disponível em: <https://reactnative.dev>. Acesso em: 27 out. 2025.

Reed, V. A. **An introdocution to children with language disorders**. 5. ed. Pearson, 2018.

Rousavy, M. **VisionCamera Documentation | VisionCamera**, 2025. Disponível em: <https://react-native-vision-camera.com>. Acesso em: 27 out. 2025.