Algumas estruturas de dados utilizadas pelo OpenCV para representar imagens são: Mat, UMat, Ptr, InputArray, OutputArray entre outras. Mat é a estrutura fundamental para representar imagens no OpenCV, ela é uma matriz multidimensional que armazena o valor dos pixels da imagem. UMat é parecido com a estrutura Mat, porém é uma matriz unificada que processa imagens com um código específico do OpenCL, usando GPU caso exista no sistema, mas que alterna automaticamente para CPU caso contrário. A estrutura do Ptr também é similar à do Mat, mas difere no fato de que o Ptr gerencia automaticamente a memória do objeto associado a ele, enquanto o Mat precisa ser gerenciado manualmente.

InputArray e OutputArray são funções mais específicas que são usadas para fornecer uma interface uniforme para diferentes tipos de dados, incluindo imagens. O grande diferencial dessas estruturas é que elas tratam as imagens como read-only e write-only, para InputArray e OutputArray respectivamente.

Para definir qual dessas estruturas de dados vão ser usadas é essencial entender primeiro qual a sua necessidade específica. Se você quiser ter um desempenho melhor no processamento de imagens é recomendado utilizar a estrutura UMat, se quiser evitar erros em relação ao gerenciamento de memória pode-se usar o Ptr, se você quiser representar uma entrada ou saída pode-se usar o InputArray e OutputArray. Por fim, a estrutura Mat por ser a mais geral pode ser usada em outros diversos casos.

Além disso, duas principais estruturas de dados utilizadas pelo OpenCV para representar vídeos são: VideoCapture e VideoWriter. A primeira é usada para gravar vídeos ou para ler um arquivo de vídeo ou sequência de imagens, enquanto a segunda é usada para salvar o vídeo em um diretório.

É de extrema importância compreender bem as estruturas de dados ao trabalhar com algoritmos de visão computacional pois afeta diretamente o desempenho do código. Escolher a estrutura de dados mais adequada pode ajudar a reduzir erros, otimizar os códigos, minimizar o uso de recursos, conseguindo assim ter um código mais robusto e eficiente.