Zona De Visão Computacional

Gonçalves, L.J.M

Resumo

A visão computacional é uma área da inteligência artificial que busca desenvolver sistemas capazes de interpretar e compreender imagens e vídeos, simulando a visão humana. O conceito de "zona de visão" em visão computacional refere-se às regiões de interesse em uma imagem ou sequência de imagens, onde as análises e extrações de informações são concentradas. Isso pode incluir identificação de objetos, reconhecimento facial, rastreamento de movimento, e outras tarefas que exigem interpretação visual detalhada.

O artigo aborda técnicas de aprendizado e análise de dados utilizando as zonas de visão computacional, como a segmentação de imagens, captura de dados via zona de visão computacional. E traz um claro contraste sobre como em nosso dia-a-dia a todo instante estamos sendo imersos no ambiente digital por meio de dispositivos com a capacidade de visão computacional.

1. INTRODUÇÃO

A capacidade de resolução de problemas e auxílio a diversos setores da sociedade permitem vislumbrar a capacidade analítica de um equipamento aprimorado com visão computacional. Existem diversos setores que utilizam de forma inovadora a sua capacidade. Hoje o setor tecnológico não apenas visa incrementar seu uso, mas aprimorar a cada instante uma tecnologia que nos traz benefícios.

O estudo da visão computacional é estimado em sua fase inicial no usi de reconhecimento de imagens bidimensionais no início da década de 1960. Tendo seus trabalhos com base em reconhecimento bidimensional, com Larry Roberts, desde então temos mais de metade de um século de evolução no setor. Os campos de atuação são diversos sendo usados em veículos autônomos que circulam em nossas ruas até a tratores que aram campos totalmente autônomos, com a capacidade de incorporar algoritmos de predição, deep learning e rápida análise de dados. Somos capazes de visualizar não apenas padrões de plantações, mas

também corpos celestes distantes, o mapeamento terrestre e seu uso em telescópios que hoje exploram o espaço com uso da visão computacional.

A capacidade de manter uma câmera aberta é interessante, mas ensinar a máquina a entender o que ela está visualizando é ainda mais sublime. Observe o seguinte: para a máquina, uma câmera conectada é apenas um conjunto finito de frames contínuos. Máquinas fotográficas, câmeras de vídeo ou smartphones apenas visualizam ambiente ou capturam fotos. Ou vídeos que nada mais são que fotos sequenciais exibidas em torno de 24 a 60 frames por segundo. Desta maneira cria a ilusão de movimento contínuo que na visão humana é observável como movimento contínuo.

Para as máquinas, uma obra-prima como a Mona Lisa nada mais é que um conjunto de frames. No entanto, se acrescentarmos um script que identifique e processe o que está sendo capturado, a máquina não apenas pode nos fornecer dados detalhados sobre a obra de arte, mas também dados sobre sua moldura, estilo, autor, cores, tom de fundo, objetos em seu fundo, e até mesmo o reconhecimento dessa arte entre milhares de outras.

Sendo de forma autônoma ou por intermédio de comandos é inegável a utilidade do campo da visão computacional. O artigo descreve de forma prática e simples a forma para se criar um script inicial, que usa de visão computacional. Ao qual traz uma experiência imersiva e contínua nesta área de desenvolvimento. Delineando um princípio para diversos projetos futuros, que podem ser executados a partir do conteúdo deste artigo. Portanto é necessário entender como o reconhecimento funciona desde sua fase de aquisição. Até a análise de dados em tempo real utilizando de sistemas de análise de dados.

- **Aquisição:** A primeira etapa em visão computacional é a aquisição do objeto a ser analisado, que pode ocorrer por meio de uma imagem previamente existente ou pela captura de um vídeo em tempo real, como de uma webcam ou outro dispositivo de vídeo.

- **Pré-processamento:** Em seguida, ocorre o pré-processamento da imagem. Essa fase envolve a remoção de ruídos e a transformação da escala de cores. Além disso, a imagem pode ser normalizada, ajustando suas características para que atenda aos requisitos do sistema, garantindo uma melhor qualidade para as etapas subsequentes.
- **Segmentação:** A segmentação é o processo de dividir a imagem em regiões específicas, isolando bordas, contornos e objetos de interesse. Essa etapa é essencial para identificar características distintas da imagem que serão usadas nas fases seguintes.
- **Extração de Características:** Na extração de características, o script analisa e reconhece padrões na imagem, identificando informações relevantes como formas, texturas ou outros detalhes necessários para a análise. Esse processo transforma a imagem em um conjunto de dados processáveis.
- **Classificação:** A classificação é responsável por distinguir objetos ou elementos específicos da imagem. Com base nos dados extraídos, o algoritmo pode identificar e diferenciar pessoas, rostos, objetos ou quaisquer outros itens que sejam o foco da análise.
- **Pós-processamento:** O pós-processamento envolve o refinamento dos resultados obtidos, aprimorando a precisão do algoritmo. Esta etapa melhora a qualidade da imagem e a eficiência da obtenção de dados, garantindo que os resultados sejam mais precisos e úteis.
- **Análise:** Finalmente, na fase de análise, os dados extraídos são avaliados. Esses dados podem ser armazenados, manipulados ou usados para criar novas abordagens de tratamento, conforme os objetivos do projeto.

Em seguida iremos explorar de um modo simples para a aprender de forma consistente ainda mais sobre este incrível universo, desde a concepção da ideia do projeto até a parte de elaboração de seu script e seu teste em um ambiente. Usando realidade aumentada e desenvolvimento de visão computacional. Em seguida será destacada a forma de como se iniciar um novo projeto, qual sua função e como o executar.

Antes de se iniciar um novo projeto deve-se defini-lo em três questões as quais são: Qual o objetivo? Como seremos capazes de realizar o projeto? E qual será seu nível de abrangência? Com tais questões resolvidas, o desempenho do projeto deve alcançar e manter um padrão de qualidade e desempenho.

Em resposta às questões anteriores começaremos definindo o nosso projeto. O projeto que definiremos será um script de modelo de Visão Computacional, seu objetivo principal é a detecção de bordas em rodovias. Com isso como foco agora devemos aprender como o projeto deve ser realizado. De qual forma ocorre o processamento das imagens, o que são frames? O que são pixels e como acontece a captura de dados? E a sua manipulação. Como iremos definir se seremos capazes de realizar o projeto. E na parte inicial definiremos como tratar a aquisição de realidade aumentada do nosso projeto começando pelo processamento.

2. PROCESSAMENTO DE IMAGENS

O processamento de imagem possui uma variedade de formas de realizar a leitura e a manipulação de imagens. Dois dos principais conceitos utilizados são a escala de cores, e o processamento de pixels.

O pixel(picture element) é a menor unidade digital de uma imagem, cada pixel representa a cor em um ponto específico da imagem. A resolução de uma imagem é definida pelo número de pixels na horizontal e na vertical. Por exemplo, uma imagem com resolução de 1920x1080 pixels tem 1920 pixels em cada linha horizontal e 1080 pixels em cada coluna vertical.

Esses conceitos são fundamentais no processamento de imagem e podem ser combinados para criar soluções complexas, como reconhecimento facial, rastreamento de objetos, e análise de vídeo em tempo real.

Sobre o tratamento da escala de cores, os pixels geralmente possuem um método em escala de cores que se dividem em três métodos em escala sendo os mais comumente usados.

- **Escala de Cinza (Grayscale):** Uma imagem em escala de cinza é uma imagem que contém apenas tons de cinza, variando do preto ao branco. Cada pixel é representado por um valor de intensidade que varia entre 0 (preto) e 255 (branco). Para conversão de escala em grascale em OpenCv use o script abaixo.

cv2.cvtColor#gray_image=cv2.cvtColor(color_image,cv2.COLOR_BGR2GRAY)

- **Escala RGB (Red, Green, Blue):** Uma imagem colorida em escala RGBé composta por três canais:Vermelho (Red), Verde (Green) e Azul (Blue). Cada canal contém valores que variam de 0 a 255, representando a intensidade da cor naquele pixel. A combinação dos valores dos três canais determina a cor final do pixel. No OpenCV, as imagens são lidas no formato BGR (Blue, Green, Red) por padrão.

blue = image[:, :, 0] # Canal Azul green = image[:, :, 1] # Canal Verde red = image[:, :, 2] # Canal Vermelho

- **Escala HSV (Hue, Saturation, Value):** A escala HSV é uma representação alternativa onde. Essa escala é útil no processamento de imagem, especialmente para segmentação baseada em cor, porque separa as informações de cor da intensidade da luz.

Hue (Matiz): Representa a cor (ângulo entre 0 e 360 graus).

Saturation (Saturação): Representa a vivacidade da cor (0 a 255).

Value (Valor): Representa o brilho da cor (0 a 255).

Essa escala é útil no processamento de imagem, especialmente para segmentação baseada em cor, porque separa as informações de cor da intensidade da luz. No OpenCV, você pode converter uma imagem de BGR para HSV usando: *hsv_image = cv2.cvtColor(color_image, cv2.COLOR_BGR2HSV).*

3. PACOTES E LINGUAGENS UTILIZADAS

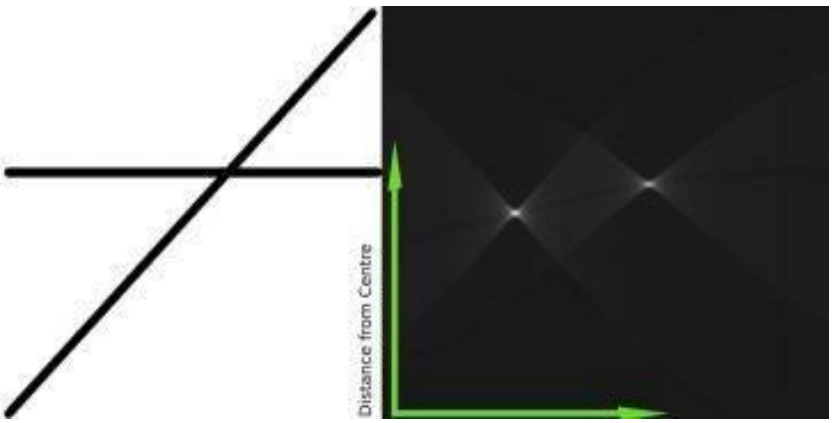
Para criar um projeto que faça algo semelhante, seja por meio de uma câmera de vídeo ou um vasto banco de dados cheio de imagens, é essencial seguir alguns passos. De forma clara deve-se primeiramente obter uma IDE(Integrated Development Environment), seja Visual Studio, Pycharm, jupyter ou spyder. A linguagem de programação recomendada será Python, e a biblioteca utilizada será a OpenCv(Open Source Computer Vision Library).

O pacote numpy(Numerical Python), utilizado por sua robusta biblioteca de computação numérica. Ela fornece suporte para arrays multidimensionais e matrizes, junto com uma coleção de funções matemáticas de alto nível para operar com esses arrays.

Para este script em específico a realização de cálculos e realizá-lo em nível 1D, 2D e 3D, usaremos o numpy do qual já falamos sobre. E para obter os dados para conseguirmos calcular de forma devida usaremos a Transformada de Hough.

A Partir da Transformada de Hough que é uma técnica usada em processamento de imagens para detectar formas geométricas em uma imagem, sendo particularmente eficaz para a detecção de linhas, círculos e outras formas parametrizáveis. Ela é amplamente utilizada em visão computacional para reconhecer padrões de bordas em imagens.

A Transformada de Hough converte o problema de encontrar formas na imagem (por exemplo, uma linha) em um problema de busca no espaço de parâmetros da forma. Em vez de procurar diretamente as linhas na imagem, a transformada mapeia cada ponto da imagem para o espaço de parâmetros (espaço de Hough) e, então, procura por interseções que indicam a presença de uma linha.



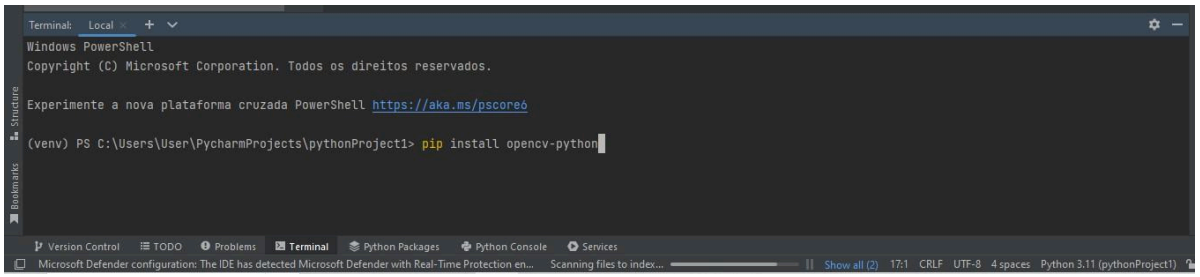
Fonte: **Exemplo do funcionamento da transformada de Hough**

Agora que sabemos como criar nosso projeto, somos capazes de realizá-lo por meio da transformada de Hough. Iremos iniciar o projeto desde seus primeiros passos até a conclusão final do projeto.

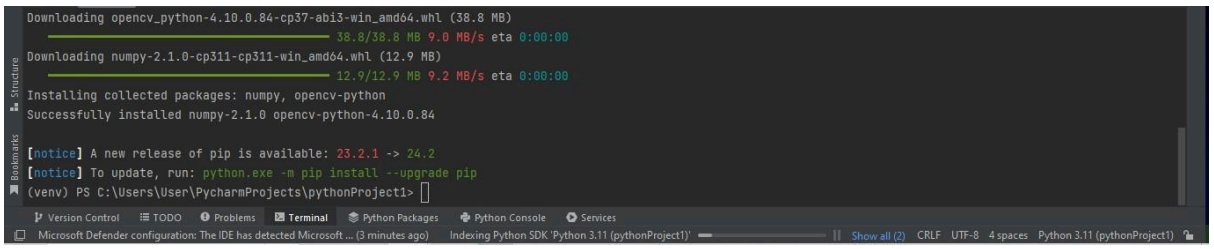
4. CRIAÇÃO DO PROJETO: PASSO A PASSO

O primeiro passo no desenvolvimento é instalar a biblioteca OpenCV na sua IDE, para que possa ser utilizada no script de visão computacional que você criará. O comando a seguir é necessário para instalação da biblioteca, e você deve digitar no terminal de sua IDE ou prompt de comando a seguinte sintaxe.

#pip install opencv-python



Após a instalação verifique o pacote baixado e se foi instalado corretamente, observe a imagem abaixo e verifique os pacotes e dados.



Após a instalação da biblioteca será possível utilizar técnicas de processamento visual e análise de dados. Com a biblioteca disponível agora seremos capazes de desenvolver o script do projeto e neste momento o criaremos um para detecção de bordas e visualização em tempo real.

A importação de bibliotecas é criada a partir do cabeçalho do código, importando diversas funções necessárias para o projeto. A partir dos recursos já disponíveis podemos começar a desenvolver nosso script,começando por importar as referidas bibliotecas disponibilizando-as para uso no código. Vislumbre a imagem abaixo observando a sintaxe correta para o desenvolvimento do código.

#import cv2

#import numpy as np

Após a importação é necessário definir os parâmetros em que o script tenha qual a imagem que deve reconhecer os seus parâmetros e definir a imagem ou ambiente.

Reconhecer objetos em tempo real ou em imagens traz uma demonstração de como podemos utilizar sistemas de computação visual de forma a interagir com nosso dia-a-dia. Desta forma devemos definir o caminho para o arquivo que deve ser reconhecido os traços do projeto. Segue o exemplo abaixo:

```
# Carregar a imagem
image = cv2.imread('C:/Users/User/Downloads/rodovia3d.jpeg')

```

Com a imagem carregada deve-se agora definir o método de reconhecimento, neste caso usaremos o método da Transformada de Hough para o reconhecimento de imagens. Abaixo veremos não apenas sua sintaxe mas também seu funcionamento.

gray_image = cv3.cvtColor(image, cv3.COLOR_BGR2GRAY)

```

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

print("Aqui ocorre a detecção de pixel que são transformados em escala de cinza")
```

Após a definição do algoritmo de busca iremos utilizar o dimensionamento da imagem adicionando a capacidade de detecção de bordas. Isso para que nosso algoritmo reconheça a auto estrada e mantenha seu curso conforme o deslocamento.

```
# Detecção de bordas
edges = cv2.Canny(gray_image, 50, 150)

print("Detecção de bordas após o processamento de escala em cinza.")

```

A aplicação da transformada de hough aplicada em linhas nas bordas da rodovia cria de forma consistente a detecção de bordas detectadas. E assim que definidas vamos ser capazes de criar não apenas a detecção mas também poderemos conseguir manipular os dados para decisão precisa com dados obtidos com a aquisição.

```

# Detecção de linhas usando Transformada de Hough
lines = cv2.HoughLinesP(edges, 1, np.pi/180, 100, minLineLength=100, maxLineGap=50)

print("Define as linhas de detecção de borda da rodovia.")
```

Após desenharmos as bordas na imagem e estarmos definindo as bordas da auto estrada. Com este projeto se pode contemplar quão benéfico é desenvolver

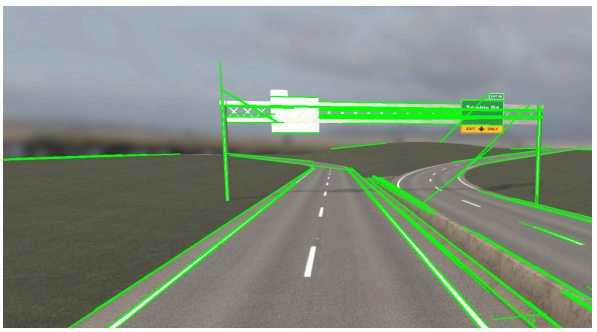
```
    # Desenhar as linhas detectadas na imagem original
    for line in lines:
        x1, y1, x2, y2 = line[0]
        cv2.line(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
```

novas tecnologias, de forma a ser usada não apenas em usos extremos mas ao dia-a-dia. A visão computacional é um campo amplo que deve ser explorado com dedicação e inovações.

A exibição da imagem resultante, deve ser chamada em uma nova janela e assim de forma clara conseguimos visualizar o resultado. Com o título de 'Rodovia Detectada' também podemos fechar a nova janela pressionando a tecla "0", de forma que mantemos a janela aberta e também podemos analisar em seguida os dados da imagem.

```
# Exibir a imagem resultante
cv2.imshow('Rodovia Detectada', image)
cv2.waitKey(0)
print("Código de parada")
cv2.destroyAllWindows()
```

Após o código de criação conseguimos visualizar o resultado adaptado com nosso algoritmo, de forma que criamos o sistema de detecção de bordas em uma rodovia. Abaixo conseguimos visualizar o antes e depois visualizando as rodovias detectadas junto com as bordas.



5. CONSIDERAÇÕES FINAIS

A criação do script revela o imenso potencial da visão computacional, uma área que permite não apenas a análise de dados visuais, mas também a estruturação de comandos a partir dessa análise, impulsionando o desenvolvimento de softwares e scripts para diversas áreas. A visão computacional tem aplicações amplas, abrangendo campos como medicina, arte, educação, veículos autônomos, reconhecimento facial e de movimentos, entre outros.

O exemplo de reconhecimento de bordas em rodovias ilustra de forma prática a versatilidade da visão computacional. Essa técnica, essencial para veículos autônomos, pode ser adaptada para outros contextos, como monitoramento de tráfego e segurança. A capacidade de detectar e analisar bordas, padrões e outros detalhes visuais em tempo real demonstra como a visão computacional transforma a maneira como máquinas interagem com o ambiente visual, possibilitando a criação de sistemas autônomos e inteligentes.

Além do reconhecimento facial, a automação de veículos não tripulados é outro exemplo de como a visão computacional pode ser aplicada de maneira eficaz. Esse campo permite a criação de algoritmos para diversas finalidades, desde o rastreamento de objetos até a imersão em ambientes digitais, contribuindo para a evolução de tecnologias como realidade mista, educação avançada, e segurança industrial.

Assim, a visão computacional se destaca como uma ferramenta poderosa, promovendo inovação, imersão e soluções para vários setores. Seja na medicina, com a análise detalhada de imagens médicas, ou na indústria, com a automação de processos complexos, esse campo nos oferece um leque infinito de possibilidades para criar um futuro mais eficiente, seguro e tecnologicamente avançado.

Referências

Bradski, G. and Kaehler, A. **Learning OpenCV: Computer Vision with the OpenCVLibrary**. O'Reilly Media, 2008.

Duda, R.O. and Hart, P.E. **Use of the Hough Transformation to Detect Lines and Curves in Pictures**. Communications of the ACM, 1972.

Forsyth, D. and Ponce, J. **Computer Vision: A Modern Approach**. 2nd ed. Pearson, 2011.

Gonzalez, R.C. and Woods, R.E. **Digital Image Processing**. 4th ed. Pearson, 2018.

Goodfellow, I., Bengio, Y., and Courville, A. **Deep Learning**. MIT Press, 2016.

LeCun, Y., Bengio, Y., and Hinton, G. **Deep Learning**. Nature, 521(7553), pp. 436-444, 2015.

Marr, D. **Vision: A Computational Investigation into the Human Representation and Processing of Visual Information**. W.H. Freeman, 1982.

OpenCV: Open Source Computer Vision Library. OpenCV Documentation, 2023.

Russell, S. and Norvig, P. **Artificial Intelligence: A Modern Approach**. 4th ed.Pearson, 2020.

Szeliski, R. **Computer Vision: Algorithms and Applications**. Springer, 2010.