Pipelines com Python

Com scripts em Python, você pode aproveitar todo o poder do OpenCV para construir rapidamente seus próprios pipelines. O interpretador Python está integrado ao backend em C++ do Limelight, portanto, erros e falhas são tratados de maneira elegante.

Os pipelines regulares do SnapScript são programados diretamente na interface web do Limelight.

https://player.vimeo.com/video/872718295?h=59f7302d19

Limelight cuida do hardware, da interface com a câmera, da rede, da transmissão de dados e do pré-processamento básico de imagens. Tudo o que você precisa fazer é escrever uma única função em Python chamada runPipeline().

- Uma das características mais importantes que oferecemos é o retículo (crosshair) de um clique. O retículo, o retículo duplo, tx, ty, ta, ts, tvert e todas as outras leituras padrão do Limelight NetworkTables se ajustarão automaticamente ao contorno que você retornar da função Python runPipeline().
- Escreva suas próprias visualizações em tempo real, limiares (thresholding), filtros e passe por completo pelo nosso backend se desejar.
  - O scripting Python do Limelight tem acesso às bibliotecas completas de OpenCV e numpy.
  - Além do acesso à imagem, a função runPipeline() também tem acesso à matriz numérica "llrobot" da NetworkTables. Envie quaisquer dados de seus robôs para seus scripts Python para visualização ou aplicações avançadas (pode-se enviar dados de IMU, dados de pose, velocidade do robô, etc., para uso em scripts Python).
  - A função runPipeline também produz uma matriz numérica que é colocada diretamente na matriz numérica "llpython" da NetworkTables. Isso significa que você pode ignorar completamente o retículo e outras funcionalidades do Limelight e enviar seus próprios dados personalizados de volta para seus robôs.
  - Os scripts Python são isolados dentro do nosso ambiente C++, então você não precisa se preocupar com falhas. Mudanças nos scripts são aplicadas instantaneamente, e quaisquer mensagens de erro são impressas diretamente na interface web.

```
import cv2
import numpy as np

# runPipeline() is called every frame by Limelight's backend.
```

```
    def runPipeline(image, llrobot):
        # convert the input image to the HSV color space
        img_hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        # convert the hsv to a binary image by removing any pixels
        # that do not fall within the following HSV Min/Max values
        img_threshold = cv2.inRange(img_hsv, (60, 70, 70), (85, 255, 255))

        # find contours in the new binary image
        contours, _ = cv2.findContours(img_threshold,
        cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        largestContour = np.array([[]])

        # initialize an empty array of values to send back to the robot
        llpython = [0,0,0,0,0,0,0,0]

        # if contours have been detected, draw them
        if len(contours) > 0:
            cv2.drawContours(image, contours, -1, 255, 2)
            # record the largest contour
            largestContour = max(contours, key=cv2.contourArea)

            # get the unrotated bounding box that surrounds the contour
            x,y,w,h = cv2.boundingRect(largestContour)

            # draw the unrotated bounding box
            cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,255),2)

            # record some custom data to send back to the robot
            llpython = [1,x,y,w,h,9,8,7]

        #return the largest contour for the LL crosshair, the modified image, and custom robot
data
        return largestContour, image, llpython
```

Revisão #1
Criado 2024-01-22 13:40:45 UTC por Enzo
Atualizado: 2024-01-22 13:46:05 UTC por Enzo