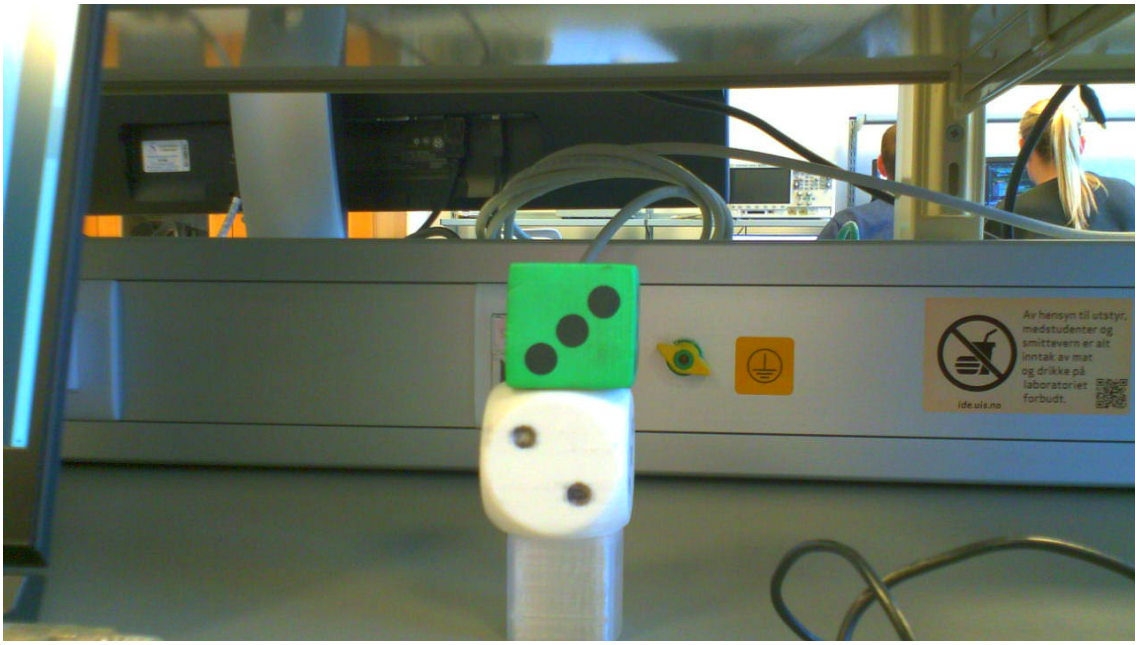
Image Acquisition Assignment 2

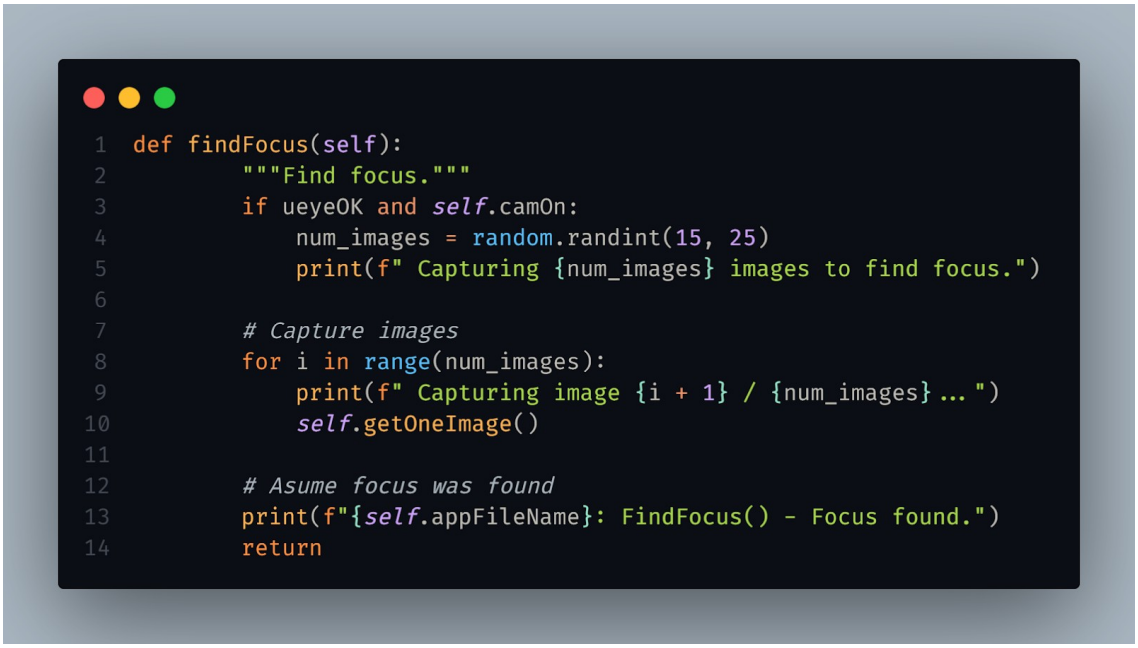
Théo de Morais // Jesús Sánchez

2.1 Capture an image using IDS software:



2.2 Use IDS Camera and Python

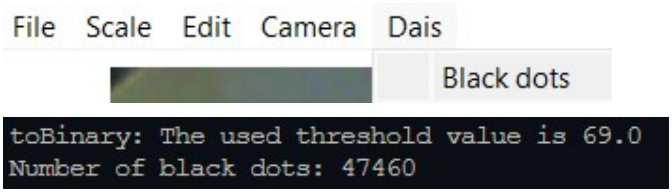
b. Create a function to find focus:



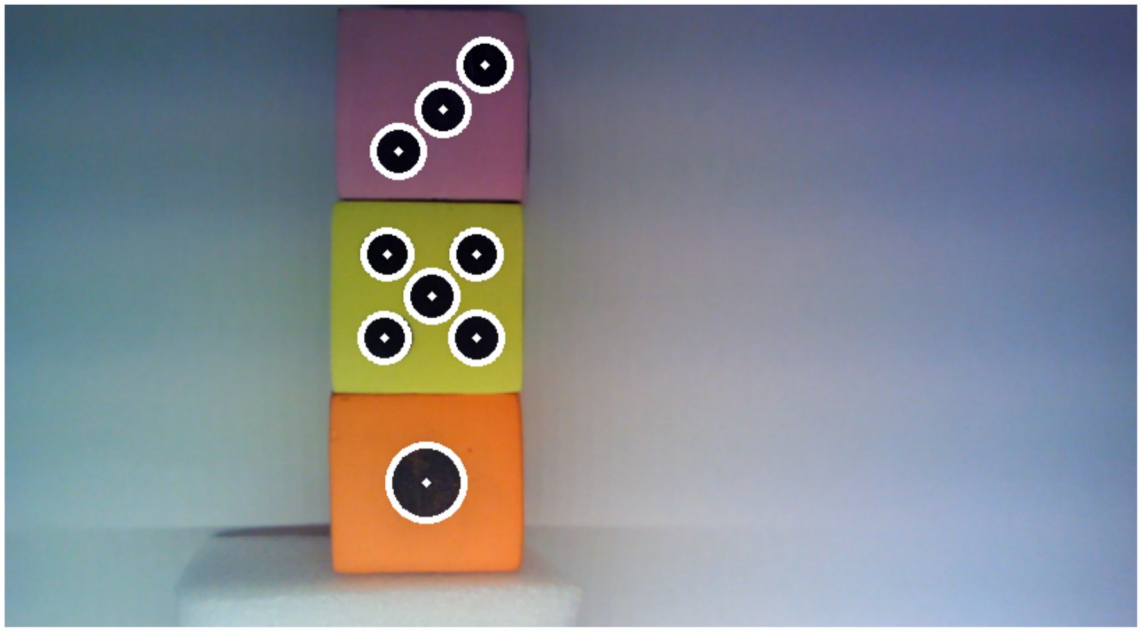
```python
def findFocus(self):
        """Find focus."""
        if ueyeOK and self.camOn:
            num_images = random.randint(15, 25)
            print(f" Capturing {num_images} images to find focus.")

        # Capture images
        for i in range(num_images):
            print(f" Capturing image {i + 1} / {num_images} ... ")
            self.getOneImage()

        # Asume focus was found
        print(f"{self.appFileName}: FindFocus() - Focus found.")
        return
```

c. Find black dots function

```python
def blackDots(self):
        """Detect black dots on a dice."""
        if ueyeOK and self.camOn:
            if self.npImage.size == 0:
                print(f"{self.appFileName}: blackDots() no image in buffer")
                return

        print(f"{self.appFileName}: BlackDots() - Detecting black dots on the dice.")


        self.toGray()
        self.toBinary()

        # Convert QPixmap to QImage
        qimage = self.pixmap.toImage()

        # Convert QImage to numpy array
        width = qimage.width()
        height = qimage.height()
        channels = 4 if qimage.hasAlphaChannel() else 3
        image_data = qimage.bits().asarray(height * width * channels)

        # Reshape numpy array to OpenCV format
        img = np.frombuffer(image_data, dtype=np.uint8).reshape((height, width, channels))

        # Convert from RGB(A) to BGR (OpenCV format)
        if channels == 4:
            img = cv2.cvtColor(img, cv2.COLOR_RGBA2BGR)
        else:
            img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)


        nDots = np.sum(img == 0)

        print(f"Number of black dots: {nDots}")
```



d. Find circles



```python
def find_circle(self):
            """Find circles in the current image using HoughCircles."""
            if self.npImage.size == 0:
                print(f"{self.appFileName}: find_circle() no image in buffer")
                return

            print(f"{self.appFileName}: find_circle() - Detecting circles in the image.")

            # Convert the image to grayscale
            gray = cv2.cvtColor(self.npImage, cv2.COLOR_BGR2GRAY)
            gray = cv2.medianBlur(gray, 5)

            # Detect circles using HoughCircles
            circles = cv2.HoughCircles(gray, cv2.HOUGH_GRADIENT, 1, 10, param1=30, param2=30, minRadius=2, maxRadius=50)

            if circles is not None:
                circles = np.uint16(np.around(circles))
                num_circles = circles.shape[1]
                print(f"Number of circles detected: {num_circles}")
                for i in circles[0, :]:
                    center = (i[0], i[1])
                    # Draw circle center
                    cv2.circle(self.npImage, center, 1, (0, 100, 100), 3)
                    # Draw circle outline
                    radius = i[2]
                    cv2.circle(self.npImage, center, radius, (255, 0, 255), 3)

                # Update the QPixmap with the detected circles
                self.image = np2qimage(self.npImage)
                self.pixmap = QPixmap.fromImage(self.image)
                if self.curItem:
                    self.scene.removeItem(self.curItem)
                self.curItem = QGraphicsPixmapItem(self.pixmap)
                self.scene.addItem(self.curItem)
                self.scene.setSceneRect(0, 0, self.pixmap.width(), self.pixmap.height())
                self.setWindowTitle(f"{self.appFileName} : Circles detected")
                (w, h) = (self.pixmap.width(), self.pixmap.height())
                self.status.setText(f"pixmap: (w,h) = ({w},{h})")
                self.scaleOne()
                self.view.setMouseTracking(True)
            else:
                print("No circles were found.")
            return

```

```
appImageViewer2V: find_circle() - Detecting circles in the image.
Number of circles detected: 9
```

e. Print some camera information:

```python
def printCameraInfo(self):
        """Print some information on camera."""
        if ueyeOK and self.camOn:
            print("printCameraInfo(): print (test) state and settings.")
            d = ueye.double()
            ui1 = ueye.uint()
            retVal = ueye.is_SetFrameRate(self.cam.handle(), 2.0, d)
            if retVal == ueye.IS_SUCCESS:
                print( f"  frame rate set to                    {float(d):8.3f} fps" )
            retVal = ueye.is_Exposure(self.cam.handle(),
                                      ueye.IS_EXPOSURE_CMD_GET_EXPOSURE_DEFAULT, d, 8)
            if retVal == ueye.IS_SUCCESS:
                print( f"  default setting for the exposure time  {float(d):8.3f} ms" )
            retVal = ueye.is_Exposure(self.cam.handle(),
                                      ueye.IS_EXPOSURE_CMD_GET_EXPOSURE_RANGE_MIN, d, 8)
            if retVal == ueye.IS_SUCCESS:
                print( f"  minimum exposure time                {float(d):8.3f} ms" )
            retVal = ueye.is_Exposure(self.cam.handle(),
                                      ueye.IS_EXPOSURE_CMD_GET_EXPOSURE_RANGE_MAX, d, 8)
            if retVal == ueye.IS_SUCCESS:
                print( f"  maximum exposure time                {float(d):8.3f} ms" )
            #
            print( f"  sys.getsizeof(d) returns   {sys.getsizeof(d)}  (??)" )
            print( f"  sys.getsizeof(ui1) returns {sys.getsizeof(ui1)}  (??)" )
            retVal = ueye.is_Focus(self.cam.handle(), ueye.FDT_CMD_GET_CAPABILITIES, ui1, 4)
            if ((retVal == ueye.IS_SUCCESS) and (ui1 & ueye.FOC_CAP_AUTOFOCUS_SUPPORTED)):
                print( "  autofocus supported" )
            if retVal == ueye.IS_SUCCESS:
                print( f"  is_Focus() is success          ui1 = {ui1}" )
            else:
                print( f"  is_Focus() is NOT success   retVal = {retVal}" )
            fZR = ueye.IS_RECT()
            retVal = ueye.is_Focus(self.cam.handle(), ueye.FOC_CMD_SET_ENABLE_AUTOFOCUS, ui1, 0)
            if retVal == ueye.IS_SUCCESS:
                print( f"  is_Focus( ENABLE ) is success      " )
            retVal = ueye.is_Focus(self.cam.handle(), ueye.FOC_CMD_GET_AUTOFOCUS_STATUS, ui1, 4)
            if retVal == ueye.IS_SUCCESS:
                print( f"  is_Focus( STATUS ) is success  ui1 = {ui1}" )
            retVal = ueye.is_Exposure(self.cam.handle(), ueye.IS_EXPOSURE_CMD_GET_EXPOSURE, d, 8)
            if retVal == ueye.IS_SUCCESS:
                print( f"  currently set exposure time          {float(d):8.3f} ms" )
            d =  ueye.double(5.0)
            retVal = ueye.is_Exposure(self.cam.handle(), ueye.IS_EXPOSURE_CMD_SET_EXPOSURE, d, 8)
            if retVal == ueye.IS_SUCCESS:
                print( f"  tried to changed exposure time to    {float(d):8.3f} ms" )
            retVal = ueye.is_Exposure(self.cam.handle(), ueye.IS_EXPOSURE_CMD_GET_EXPOSURE, d, 8)
            if retVal == ueye.IS_SUCCESS:
                print( f"  currently set exposure time          {float(d):8.3f} ms" )
            #
        return
```

```
printCameraInfo(): print (test) state and settings.
  frame rate set to                      2.000 fps
  default setting for the exposure time    23.000 ms
  minimum exposure time                     0.067 ms
  maximum exposure time                   499.855 ms
  sys.getsizeof(d) returns   128  (??)
  sys.getsizeof(ui1) returns 128  (??)
  autofocus supported
  is_Focus() is success          ui1 = 35
  is_Focus( ENABLE ) is success
  is_Focus( STATUS ) is success  ui1 = 0
  currently set exposure time             66.724 ms
  tried to changed exposure time to        5.000 ms
  currently set exposure time             66.724 ms
```