deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

EXAMS EVALUATION
using Computer Vision

Lúcia Sousa - 93086
Raquel Pinto - 92948

03/01/2023 VC - Computer Vision

Index

01

Project Introduction

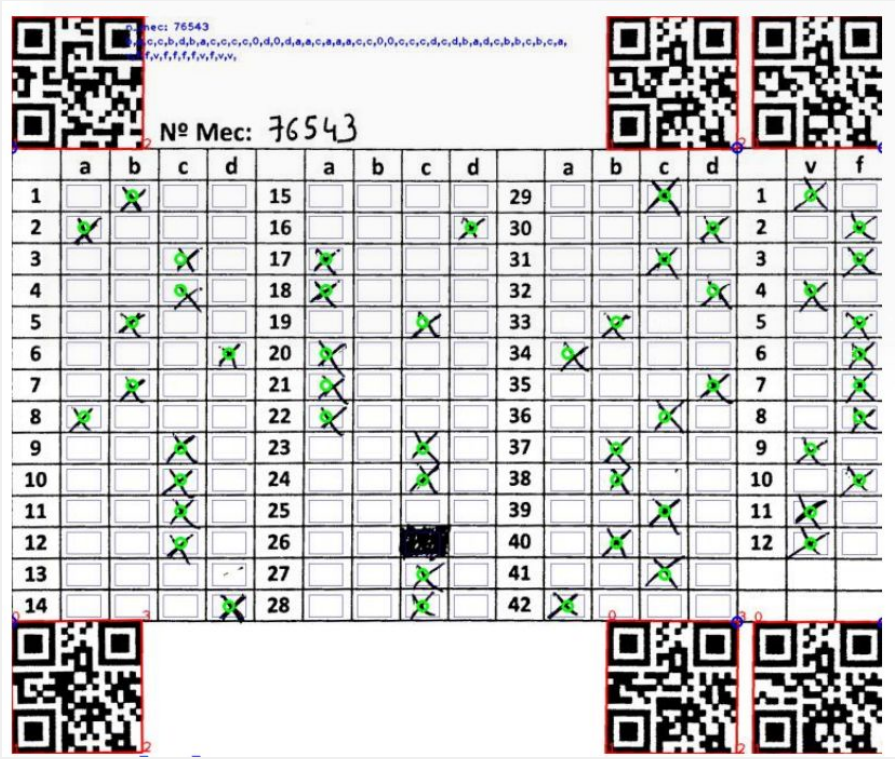
Description of the project

Exams Evaluation using Computer Vision

Read the QR Codes or Aruco to detect the borders of the exam.

Detecting the checkboxes and the crosses inside them.

At the end have the sequence of the student's answers.



02

Our Approach

The methods used to solve the project

2.1

First Approach

Using Aruco Markers, **take a picture** of the exam and proceed to the correction

First Approach

Detect Aruco

Using OpenCV, cv2.aruco.detectMarkers().

Take Picture

Press ‘s’ key to save image of the exam with the correct perspective.

Detect boxes

Detect vertical and horizontal lines.

Convert the exam into a matrix

Matrix with i lines and j columns corresponding to the i lines and j columns of the exam.

Detect an ‘x’ or filled checkbox

For each box, detect the percentage of the black pixels, fill the matrix with ‘1’ if boxes have 15% to 80% black pixels.

Get the sequence of the answers

Iterate over the matrix, and for each ‘1’ get the option (‘a’, ‘b’, ‘c’, ‘d’, ‘V’ or ‘F’), if it is not answered return ‘-’.

Convert grid into matrix

	8	39	72	105	138	171	204	238	271	305	338	371	405	439	472	505	538	571
11		a	b	c	d		a	b	c	d		a	b	c	d		V	F
43	1					15					29					1		
75	2					16					30					2		
107	3					17					31					3		
141	4					18					32					4		
173	5					19					33					5		
205	6					20					34					6		
237	7					21					35					7		
271	8					22					36					8		
303	9					23					37							
335	10					24					38							
367	11					25					39							
401	12					26					40							
433	13					27												
467	14					28												

- Map the coordinates of the boxes to the matrix.
- Remove the line of options and the columns with the numbers of the questions.
- Check if there is an ‘x’ and put ‘1’ in the matrix.

2.2

Second Approach

Without Aruco Markers, **take a picture** of the exam and proceed to the correction

Second Approach

Detect Grid

Detect the rectangle with bigger area, using contourArea function.

Take Picture

Press ‘s’ key to save image of the exam with the correct perspective.

Detect boxes

Detect vertical and horizontal lines.

Convert the exam into a matrix

Matrix with i lines and j columns corresponding to the i lines and j columns of the exam.

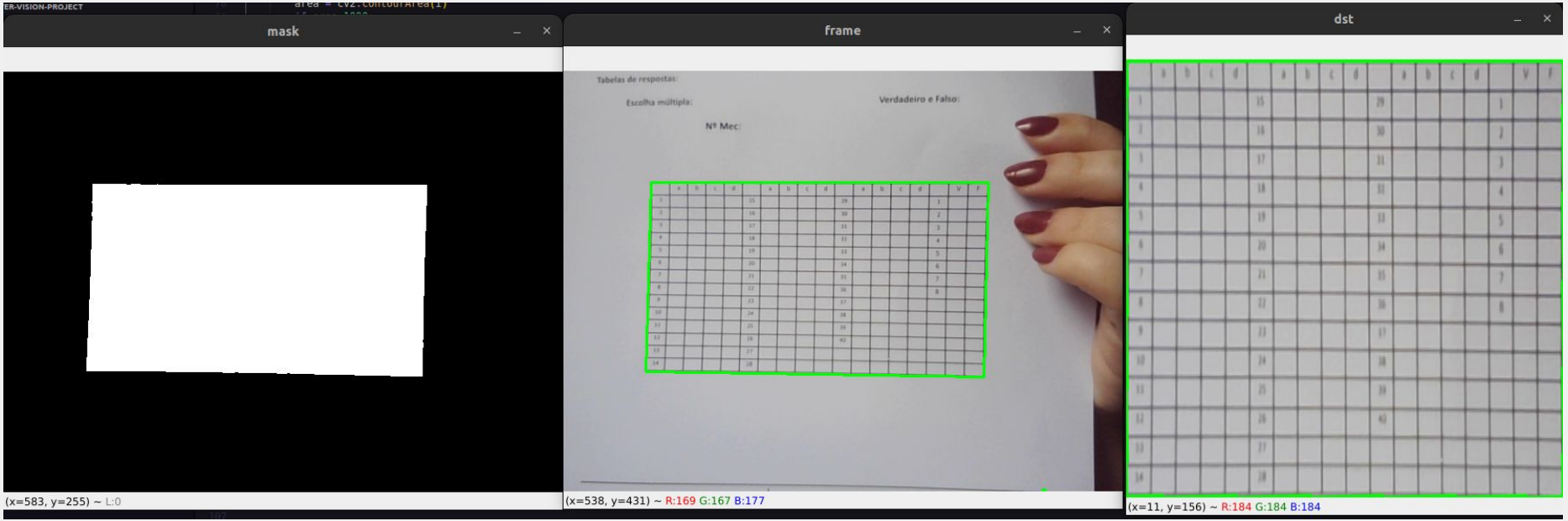
Detect an ‘x’ or filled checkbox

For each box, detect the percentage of the black pixels, fill the matrix with ‘1’ if boxes have 15% to 80% black pixels.

Get the sequence of the answers

Iterate over the matrix, and for each ‘1’ get the option (‘a’, ‘b’, ‘c’, ‘d’, ‘V’ or ‘F’), if it is not answered return ‘-’.

Detect Grid without Aruco Markers



2.3

Third Approach

Using Aruco Markers, proceed to the **live** correction of the exam

Third Approach

Detect Aruco

Using OpenCV, cv2.aruco.detectMarkers().

Signal for Feedback

Circle that it is green when the image is good, detect almost or all the boxes. Blue when the image is not so good for correction.

Detect an 'x' or filled checkbox

For each box, detect the percentage of the black pixels, fill the matrix with '1' if boxes have 15% to 80% black pixels.

Detect boxes

Detect vertical and horizontal lines, correct perspective.

Convert the exam into a matrix

Matrix with i lines and j columns corresponding to the i lines and j columns of the exam.

Get the sequence of the answers

Iterate over the matrix, and for each '1' get the option ('a', 'b', 'c', 'd', 'V' or 'F'), if it is not answered return '-'. Save the last frame.

2.4

Fourth Approach

Without Aruco Markers, proceed to the **live** correction of the exam

Fourth Approach

Detect Grid

Detect the rectangle with bigger area, using contourArea function.

Signal for Feedback

Circle that it is green when the image is good, detect almost or all the boxes. Blue when the image is not so good for correction.

Detect an 'x' or filled checkbox

For each box, detect the percentage of the black pixels, fill the matrix with '1' if boxes have 15% to 80% black pixels.

Detect boxes

Detect vertical and horizontal lines, correct perspective.

Convert the exam into a matrix

Matrix with i lines and j columns corresponding to the i lines and j columns of the exam.

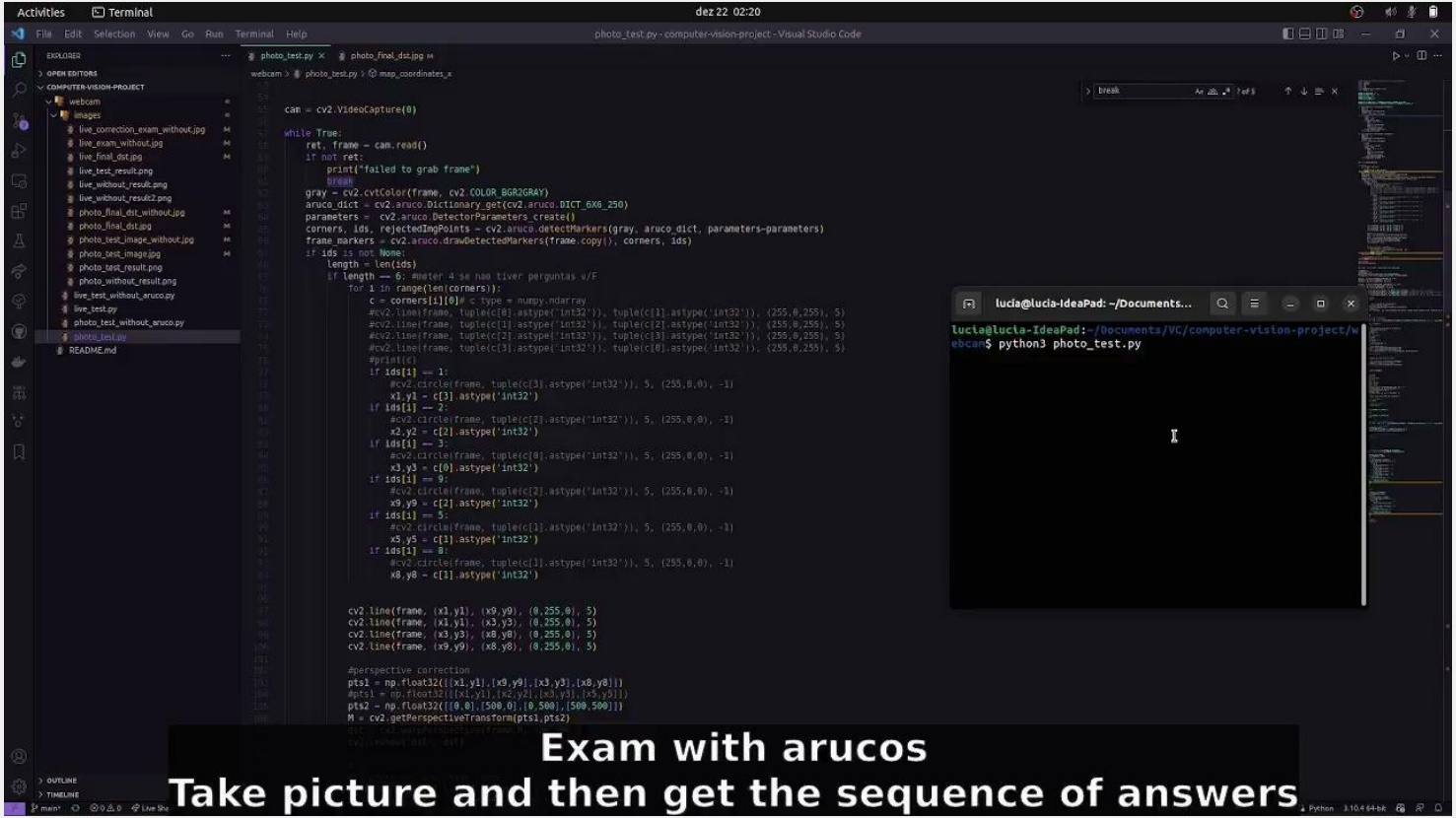
Get the sequence of the answers

Iterate over the matrix, and for each '1' get the option ('a', 'b', 'c', 'd', 'V' or 'F'), if it is not answered return '-'. Save the last frame.

03

Demonstration

Demonstration of the project


Activities
Terminal
dez 22 02:20
File Edit Selection View Go Run Terminal Help
photo_test.py - computer-vision-project - Visual Studio Code
EXPLORER
OPEN EDITORS
COMPUTER-VISION-PROJECT
webcam
images
live_correction_exam_without.jpg
live_exam_without.jpg
live_final_dst.jpg
live_test_result.png
live_without_result.png
live_without_result2.png
photo_final_dst_without.jpg
photo_final_dst.jpg
photo_test_image_without.jpg
photo_test_image.jpg
photo_test_result.png
photo_without_result.png
live_test_without_aruco.py
live_test.py
photo_test_without_aruco.py
photo_test.py
README.md
OUTLINE
TIMELINE
photo_test.py
photo_final_dst.jpg
webcam > photo_test.py > map_coordinates_x
cam = cv2.VideoCapture(0)
while True:
ret, frame = cam.read()
if not ret:
print("failed to grab frame")
break
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
aruco_dict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_6X6_250)
parameters = cv2.aruco.DetectorParameters_create()
corners, ids, rejectedImgPoints = cv2.aruco.detectMarkers(gray, aruco_dict, parameters=parameters)
frame_markers = cv2.aruco.drawDetectedMarkers(frame.copy(), corners, ids)
if ids is not None:
length = len(ids)
if length == 6: #water 4 se nao tiver perguntas v/F
for i in range(len(corners)):
c = corners[i][0]# c type = numpy.ndarray
#cv2.line(frame, tuple(c[0].astype('int32')), tuple(c[1].astype('int32')), (255,0,255), 5)
#cv2.line(frame, tuple(c[1].astype('int32')), tuple(c[2].astype('int32')), (255,0,255), 5)
#cv2.line(frame, tuple(c[2].astype('int32')), tuple(c[3].astype('int32')), (255,0,255), 5)
#cv2.line(frame, tuple(c[3].astype('int32')), tuple(c[0].astype('int32')), (255,0,255), 5)
#print(c)
if ids[i] == 1:
#cv2.circle(frame, tuple(c[3].astype('int32')), 5, (255,0,0), -1)
x1,y1 = c[3].astype('int32')
if ids[i] == 2:
#cv2.circle(frame, tuple(c[2].astype('int32')), 5, (255,0,0), -1)
x2,y2 = c[2].astype('int32')
if ids[i] == 3:
#cv2.circle(frame, tuple(c[0].astype('int32')), 5, (255,0,0), -1)
x3,y3 = c[0].astype('int32')
if ids[i] == 9:
#cv2.circle(frame, tuple(c[2].astype('int32')), 5, (255,0,0), -1)
x9,y9 = c[2].astype('int32')
if ids[i] == 5:
#cv2.circle(frame, tuple(c[1].astype('int32')), 5, (255,0,0), -1)
x5,y5 = c[1].astype('int32')
if ids[i] == 8:
#cv2.circle(frame, tuple(c[1].astype('int32')), 5, (255,0,0), -1)
x8,y8 = c[1].astype('int32')
cv2.line(frame, (x1,y1), (x9,y9), (0,255,0), 5)
cv2.line(frame, (x1,y1), (x3,y3), (0,255,0), 5)
cv2.line(frame, (x3,y3), (x8,y8), (0,255,0), 5)
cv2.line(frame, (x9,y9), (x8,y8), (0,255,0), 5)
#perspective correction
pts1 = np.float32([[x1,y1],[x9,y9],[x3,y3],[x8,y8]])
#pts1 = np.float32([[x1,y1],[x2,y2],[x3,y3],[x5,y5]])
pts2 = np.float32([[0,0],[500,0],[0,500],[500,500]])
M = cv2.getPerspectiveTransform(pts1,pts2)
break
lucia@lucia-IdeaPad: ~/Documents...
lucia@lucia-IdeaPad:~/Documents/VC/computer-vision-project/w
ebcam$ python3 photo_test.py
Exam with arucos
Take picture and then get the sequence of answers

04

Results

Results obtained

Results

First Approach		Got the sequence right 9/10
Second Approach		Got the sequence right 8/10
Third Approach		Got the sequence right 10/10
Fourth Approach		Got the sequence right 9/10

Thanks!