Article **Journal Computer Technology**
December 2023, Vol.7 No.19 1-9

Image recognition software geometry with Python and OpenCV

Software de reconocimiento de imágenes geométricas con Python y OpenCV

MAR-HERNÁNDEZ, Pedro Guillermo†, IBARRA-ANGULO, Pedro Luis, GRIJALVA-ACUÑA, Juan Carlos and ABRIL-GARCÍA, José Humberto*

Instituto Tecnologico de Hermosillo, Mexico.
Universidad de Sonora, Information Technologies Engineering, Mexico.

ID 1st Author: *Pedro Guillermo, Mar-Hernández* / **ORC ID:** 0009-0004-7080-5506, **Researcher ID Thomson:** IUM-2613-2023, **arXiv ID:** PedroMar, **CVU CONACYT ID:** 636335

ID 1st Co-author: *Pedro Luis, Ibarra-Angulo* / **ORC ID:** 0009-0009-8087-8844, **Researcher ID Thomson:** ITU-2671-2023, **arXiv ID**: plibarra1982, **CVU CONACYT ID:** 379254

ID 2nd Co-author: *Juan Carlos, Grijalva-Acuña* / **ORC ID** 0000-0001-9721-7879, **Researcher ID Thomson**: HGU-0872-2022, **arXiv ID:** JuanGrijalva1976, **CVU CONACYT ID:** 585450

ID 3rd Co-author: *José Humberto, Abril-García* / **ORC ID:** 0000-0003-3494-6817, **Researcher ID Thomson:** F-4252-2018, **arXiv ID**: jhabril, **CVU CONACYT ID**: 204935

DOI: 10.35429/JCT.2023.19.7.1.9 Received: July 10, 2023; Accepted December 30, 2023

Abstract

The project aims to develop a software application using Python and various libraries for image processing and manipulation. The software used consists of Windows 10 Pro Education, Python 3.8.2, OpenCV 4.5.3, NumPy 1.22, Imutils 0.5.4, Pillow 9.4.0, Tkinter, and Visual Studio Code 1.75.1. The methodology is divided into four sprints. In the first sprint, the necessary dependencies, including OpenCV, NumPy, Imutils, Pillow, and Tkinter, were installed. The second sprint involved developing a command-line interface and integrating a live webcam video feature. In the third sprint, color detection was implemented by converting the image to black and white and applying filters to identify specific colors. Finally, the fourth sprint focused on developing the selection of geometric figures and colors. When a figure and color were selected, the software would identify and outline the corresponding object. This project utilized Python and various libraries to create an application for image processing and manipulation. The methodology followed a sprint-based approach to gradually develop different features and functionalities of the software.

Python, OpenCV, Numpy

Resumen

El proyecto tiene como objetivo desarrollar una aplicación de software utilizando Python y varias bibliotecas para el procesamiento y manipulación de imágenes. El software utilizado consiste en Windows 10 Pro Education, Python 3.8.2, OpenCV 4.5.3, NumPy 1.22, Imutils 0.5.4, Pillow 9.4.0, Tkinter y Visual Studio Code 1.75.1. La metodología se divide en cuatro sprints. En el primer sprint, se instalaron las dependencias necesarias, incluyendo OpenCV, NumPy, Imutils, Pillow y Tkinter. El segundo sprint involucró el desarrollo de una interfaz de línea de comandos y la integración de una función de video de cámara web en vivo. En el tercer sprint, la detección de color se implementó convirtiendo la imagen a blanco y negro y aplicando filtros para identificar colores específicos. Finalmente, el cuarto sprint se centró en desarrollar la selección de figuras geométricas y colores. Cuando se seleccionaba una figura y un color, el software identificaba y delineaba el objeto correspondiente. Este proyecto utilizó Python y varias bibliotecas para crear una aplicación para el procesamiento y manipulación de imágenes. La metodología siguió un enfoque basado en sprints para desarrollar gradualmente diferentes características y funcionalidades del software.

Python, OpenCV, Numpy

Citation: MAR-HERNÁNDEZ, Pedro Guillermo, IBARRA-ANGULO, Pedro Luis, GRIJALVA-ACUÑA, Juan Carlos and ABRIL-GARCÍA, José Humberto. Image recognition software geometry with Python and OpenCV. Journal Computer Technology. 2023. 7-19:1-9.

* Correspondence to the author (E-mail: jose.abril@unison.mx).
† Researcher contributing as first author.

Introduction

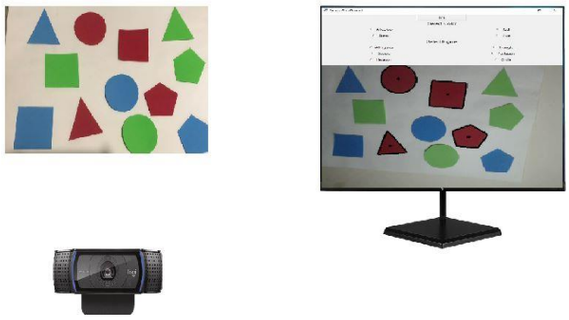

Figure 1 Diagram

Description of tools

Python was created in the late 1980s by Guido van Rossum at Stichting Mathematisch Centrum (CWI) in the Netherlands as a successor to the ABC programming language, capable of handling exceptions and interfacing with the Amoeba operating system. (Papercut Publications, 2022).

The language's name comes from its creator's fondness for British comedians Monty Python. Guido van Rossum is the main author of Python, and his continuing central role in deciding the direction of Python is recognized, referring to him as the Benevolent Dictator for Life (BDFL); however, on July 12, 2018, he declined from said position of honor without leaving a successor or successor and with a high-sounding statement. (Ortega, Faruque Sarker, & Washington, 2019)

OpenCV is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code. (Howse & Minichino, 2023)

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies. (Lentin, 2015)

VSCode is a free source code editor made by Microsoft for Windows, Linux and macOS. (Uzayr, 2023) Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality. VSCode source code comes from Microsoft's free and open source software VSCode project released under the permissive Expat License, but the compiled binaries are freeware for any use. (Speight, 2021)

GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager providing wiki, issue-tracking and continuous integration / continuous deployment pipeline features, using an open-source license, developed by GitLab Inc (Evertse, 2019). The software was created by Ukrainians Dmitriy Zaporozhets and Valery Sizov. (Murph, 2020)

NumPy is a Python library used for working with arrays. (McKinney, 2017)

It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely. NumPy stands for Numerical Python. (Rajender, 2023)

ISSN 2531-2197

Imutils (Rosebrock, 2015) (timgates42, 2022) has a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV.

The Python Imaging Library (Driscoll, 2021) or PIL, is an external library for the Python 2x programming language that allows images to be easily manipulated using high-level commands. (Pajankar, 2017)

The tkinter (Roseman, 2019) package ("Tk interface") is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems. (Moore, 2021)

Hardware and Software specificications

The hardware used in this project was:

- Intel(R) Core (TM) i5-10400F CPU @ 2.90GHz (12 CPUs), ~2.9GHz, 32 GB RAM, 64 bits.
- Logitech C920S Pro HD.

The software used in this project:

- Windows 10 Pro Education.
- Python 3.8.2.
- OpenCV 4.5.3.
- Numpy 1.22.
- Imutils 0.5.4.
- Pillow 9.4.0.
- Tkinter.
- Visual Studio Code version 1.75.1.

Methodology

Sprint 1

In the first sprint, the first thing that was done was to install the desired python version to start developing the software and install the necessary dependencies such as:

- OpenCV.
- Numpy.
- Imutils.
- Pillow.
- Tkinter.

Sprint 2

In the second sprint, a command line interface was developed to be able to program the desired actions from the buttons, a button was also programmed to be able to call the live webcam video.

Sprint 3

In the third sprint, color detection was developed, first we had to transform the image to black and white, followed by applying a filter with ranges of tones to identify the colors.

Sprint 4

In this fourth and final sprint, the selection of geometric figures and colors was developed, when selecting the button of a figure and the one of a color, only that object would be identified with a pointer and the outline of the selected object.

ISSN 2531-2197

Results

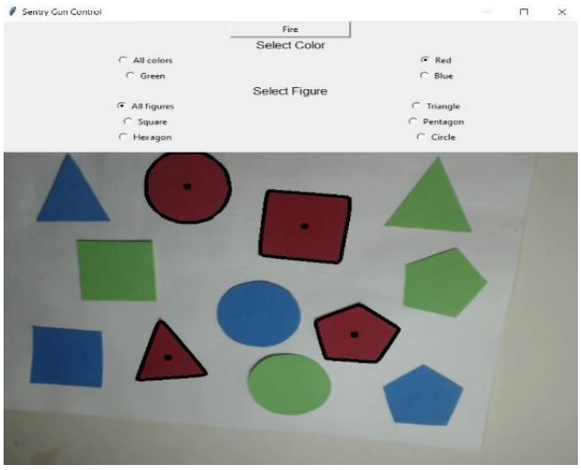

Figure 2

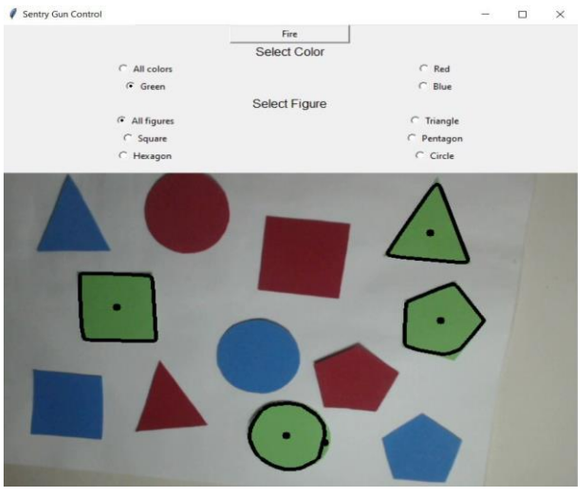

Figure 3

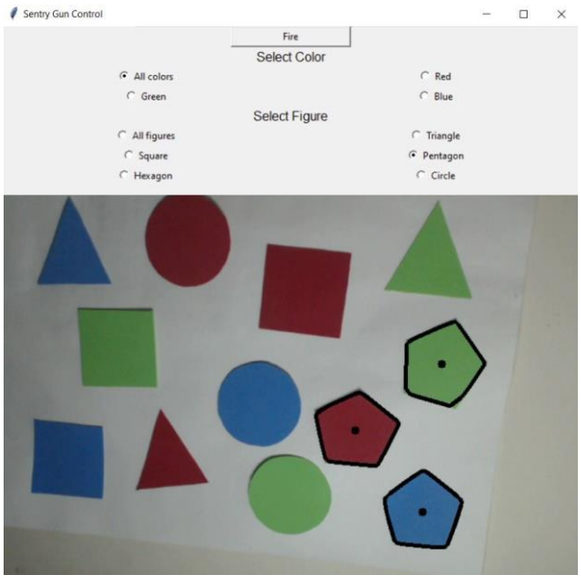

Figure 4

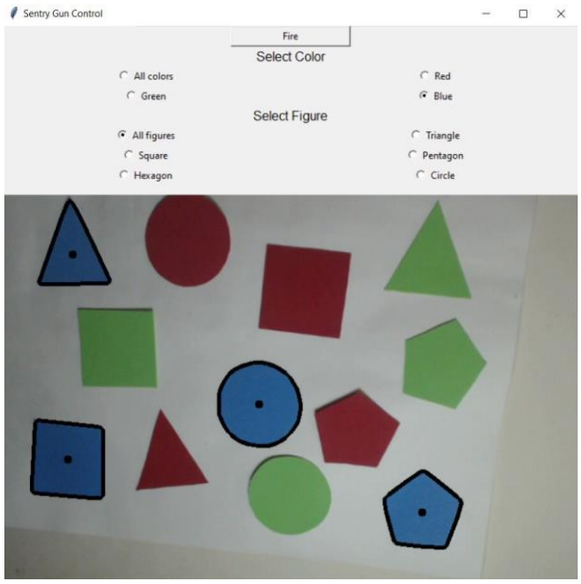

Figure 5

Appendix: Python and OpenCV

```
from tkinter import *
from PIL import Image
from PIL import ImageTk
import cv2
import imutils
import numpy as np

root = Tk()
root.title('Sentry Gun Control')
cap=None

lowRed1 = np.array([0, 100, 20],
np.uint8)
highRed1 = np.array([8, 255, 255],
np.uint8)
lowRed2 = np.array([175, 100, 20],
np.uint8)
highRed2 = np.array([180, 255, 255],
np.uint8)
lowGreen = np.array([36, 100, 20],
np.uint8)
highGreen = np.array([70, 255, 255],
np.uint8)
lowBlue =
np.array([100,100,20],np.uint8)
highBlue =
np.array([125,255,255],np.uint8)

def allSelected():
    global cap
    if cap is not None:
        ret, frame = cap.read()
        if ret == True:
            frame =
imutils.resize(frame, width=720)
        frameHSV=cv2.cvtColor(frame,cv2.
COLOR_BGR2HSV)
        if selColor.get()==0:
```

```
            maskRed1 =
cv2.inRange(frameHSV,lowRed1,highRed1)
            maskRed2 =
cv2.inRange(frameHSV,lowRed2,highRed2)
            maskRed =
cv2.add(maskRed1,maskRed2)
            maskBlue
=cv2.inRange(frameHSV,lowBlue,highBlue)
            maskBlueRed=cv2.add(maskRed,
maskBlue)
            maskGreen=cv2.inRange(frameH
SV,lowGreen,highGreen)
            mask=cv2.add(maskBlueRed,mas
kGreen)
        mask=cv2.medianBlur(mask,7)
        finalFrame=cv2.cvtColor(frameHSV
,cv2.COLOR_HSV2RGB)
        if figure.get()==0:
            drawAll(mask,finalFrame)
        if figure.get()==3:
            drawTriangles(mask,finalFram
e)
        if figure.get()==4:
            drawSquares(mask,finalFrame)
        if figure.get()==5:
            drawPentagons(mask,finalFram
e)
        if figure.get()==7:
            drawCircles(mask,finalFrame)
        im=Image.fromarray(finalFrame)
        img=
ImageTk.PhotoImage(image=im)

        lblVideoProcesado.configure(imag
e=img)
        lblVideoProcesado.image=img
        lblVideoProcesado.after(10,allSe
lected)

def redSelected():
    global cap
    if cap is not None:
        ret, frame = cap.read()
        if ret == True:
            frame =
imutils.resize(frame, width=720)
        frameHSV=cv2.cvtColor(frame,cv2.
COLOR_BGR2HSV)
        if selColor.get()==1:
            maskRed1 =
cv2.inRange(frameHSV,lowRed1,highRed1)
            maskRed2 =
cv2.inRange(frameHSV,lowRed2,highRed2)
            mask =
cv2.add(maskRed1,maskRed2)
        mask=cv2.medianBlur(mask,7)
        finalFrame=cv2.cvtColor(frameHSV
,cv2.COLOR_HSV2RGB)
        if figure.get()==0:
            drawAll(mask,finalFrame)
        if figure.get()==3:
            drawTriangles(mask,finalFram
e)
        if figure.get()==4:
            drawSquares(mask,finalFrame)
        if figure.get()==5:
            drawPentagons(mask,finalFram
e)
        if figure.get()==7:
            drawCircles(mask,finalFrame)
        im=Image.fromarray(finalFrame)
        img=
ImageTk.PhotoImage(image=im)

        lblVideoProcesado.configure(imag
e=img)
        lblVideoProcesado.image=img
        lblVideoProcesado.after(10,redSe
lected)

def blueSelected():
    global cap
    if cap is not None:
        ret, frame = cap.read()
        if ret == True:
            frame =
imutils.resize(frame, width=720)
        frameHSV=cv2.cvtColor(frame,cv2.
COLOR_BGR2HSV)
        if selColor.get()==3:
            mask =
cv2.inRange(frameHSV,lowBlue,highBlue)
        mask=cv2.medianBlur(mask,7)
        finalFrame=cv2.cvtColor(frameHSV
,cv2.COLOR_HSV2RGB)
        if figure.get()==0:
            drawAll(mask,finalFrame)
        if figure.get()==3:
            drawTriangles(mask,finalFram
e)
        if figure.get()==4:
            drawSquares(mask,finalFrame)
        if figure.get()==5:
            drawPentagons(mask,finalFram
e)
        if figure.get()==7:
            drawCircles(mask,finalFrame)
        im=Image.fromarray(finalFrame)
        img=
ImageTk.PhotoImage(image=im)

        lblVideoProcesado.configure(imag
e=img)
        lblVideoProcesado.image=img
        lblVideoProcesado.after(10,blueS
elected)

def greenSelected():
    global cap
    if cap is not None:
        ret, frame = cap.read()
        if ret == True:
            frame =
imutils.resize(frame, width=720)
        frameHSV=cv2.cvtColor(frame,cv2.
COLOR_BGR2HSV)
```

ISSN 2531-2197

```
        if selColor.get()==2:
            mask=cv2.inRange(frameHSV,lo
wGreen,highGreen)
        mask=cv2.medianBlur(mask,7)
        finalFrame=cv2.cvtColor(frameHSV
,cv2.COLOR_HSV2RGB)
        if figure.get()==0:
            drawAll(mask,finalFrame)
        if figure.get()==3:
            drawTriangles(mask,finalFram
e)
        if figure.get()==4:
            drawSquares(mask,finalFrame)
        if figure.get()==5:
            drawPentagons(mask,finalFram
e)
        if figure.get()==7:
            drawCircles(mask,finalFrame)

        im=Image.fromarray(finalFrame)
        img=
ImageTk.PhotoImage(image=im)

        lblVideoProcesado.configure(imag
e=img)
        lblVideoProcesado.image=img
        lblVideoProcesado.after(10,green
Selected)

def drawAll(mask,finalFrame):
    cont,_=cv2.findContours(mask,cv2.RET
R_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    for c in cont:
        epsilon =
0.10*cv2.arcLength(c,True)
        approx =
cv2.approxPolyDP(c,epsilon,True)
        x,y,w,h =
cv2.boundingRect(approx)
        area = cv2.contourArea(c)
        if figure.get()==0:
            area > 3000
            target=cv2.moments(c)
            if (target["m00"]==00):
target["m00"]=1
            x=int(target["m10"]/target["
m00"])
            y=int(target["m01"]/target["
m00"])
            cv2.circle(finalFrame,(x,y),
5,(0,0,0),-1)
            newContour=cv2.convexHull(c)
            cv2.drawContours(finalFrame,
[newContour],0,(0,0,0),3)

def drawTriangles(mask,finalFrame):
    cont,_=cv2.findContours(mask,cv2.RET
R_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    for c in cont:
        epsilon =
0.10*cv2.arcLength(c,True)
        approx =
cv2.approxPolyDP(c,epsilon,True)
        x,y,w,h =
cv2.boundingRect(approx)
        area = cv2.contourArea(c)
        if len(approx)==3:
            target=cv2.moments(c)
            if (target["m00"]==00):
target["m00"]=1
            x=int(target["m10"]/target["
m00"])
            y=int(target["m01"]/target["
m00"])
            cv2.circle(finalFrame,(x,y),
5,(0,0,0),-1)
            newContour=cv2.convexHull(c)
            cv2.drawContours(finalFrame,
[newContour],0,(0,0,0),3)

def drawSquares(mask,finalFrame):
    cont,_=cv2.findContours(mask,cv2.RET
R_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    for c in cont:
        epsilon =
0.030*cv2.arcLength(c,True)
        approx =
cv2.approxPolyDP(c,epsilon,True)
        x,y,w,h =
cv2.boundingRect(approx)
        area = cv2.contourArea(c)
        if len(approx)==4:
            target=cv2.moments(c)
            if (target["m00"]==00):
target["m00"]=1
            x=int(target["m10"]/target["
m00"])
            cv2.circle(finalFrame,(x,y),
5,(0,0,0),-1)
            newContour=cv2.convexHull(c)
            cv2.drawContours(finalFrame,
[newContour],0,(0,0,0),3)

def drawPentagons(mask,finalFrame):
    cont,_=cv2.findContours(mask,cv2.RET
R_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    for c in cont:
        epsilon =
0.035*cv2.arcLength(c,True)
        approx =
cv2.approxPolyDP(c,epsilon,True)
        x,y,w,h =
cv2.boundingRect(approx)
        area = cv2.contourArea(c)
        if len(approx)==5:
            target=cv2.moments(c)
            if (target["m00"]==00):
target["m00"]=1
            x=int(target["m10"]/target["
m00"])
            y=int(target["m01"]/target["
m00"])
            cv2.circle(finalFrame,(x,y),
5,(0,0,0),-1)
            newContour=cv2.convexHull(c)
```

ISSN 2531-2197

```
            cv2.drawContours(finalFrame,
[newContour],0,(0,0,0),3)
    cont,_=cv2.findContours(mask,cv2.RET
R_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    for c in cont:
        epsilon =
0.055*cv2.arcLength(c,True)
        approx =
cv2.approxPolyDP(c,epsilon,True)
        x,y,w,h =
cv2.boundingRect(approx)
        area = cv2.contourArea(c)
        if len(approx)==6:
            target=cv2.moments(c)
            if (target["m00"]==00):
target["m00"]=1
            x=int(target["m10"]/target["
m00"])
            y=int(target["m01"]/target["
m00"])
            cv2.circle(finalFrame,(x,y),
5,(0,0,0),-1)
            newContour=cv2.convexHull(c)
            cv2.drawContours(finalFrame,
[newContour],0,(0,0,0),3)

def drawCircles(mask,finalFrame):
    cont,_=cv2.findContours(mask,cv2.RET
R_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    for c in cont:
        epsilon =
0.030*cv2.arcLength(c,True)
        approx =
cv2.approxPolyDP(c,epsilon,True)
        x,y,w,h =
cv2.boundingRect(approx)
        area = cv2.contourArea(c)
        if len(approx)>7:
            target=cv2.moments(c)
            if (target["m00"]==00):
target["m00"]=1
            x=int(target["m10"]/target["
m00"])
            y=int(target["m01"]/target["
m00"])
            cv2.circle(finalFrame,(x,y),
5,(0,0,0),-1)
            newContour=cv2.convexHull(c)
            cv2.drawContours(finalFrame,
[newContour],0,(0,0,0),3)

def readVideo():
    global cap
    if cap is not None:
        ret, frame = cap.read()
        if ret == True:
            frame =
imutils.resize(frame, width=720)
            frameHSV =
cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
            frame=cv2.cvtColor(frameHSV,
cv2.COLOR_HSV2RGB)
            im = Image.fromarray(frame)
            img=
ImageTk.PhotoImage(image=im)
            lblVideo.configure(image=img
)
            lblVideo.image=img
            lblVideo.after(10,readVideo)

            root.update()

def showVideo():
    global cap
    cap =
cv2.VideoCapture(0,cv2.CAP_DSHOW)
    if selColor.get()==0:
        allSelected()
    if selColor.get()==1:
        redSelected()
    if selColor.get()==2:
        greenSelected()
    if selColor.get()==3:
        blueSelected()

selColor = IntVar()
lblColor = Label(root,text='Select
Color', font='bold')
lblColor.grid(column=0,row=1,columnspan=
2)
rbNoneColor = Radiobutton(root,text='All
colors',value=0,variable=selColor,comman
d=allSelected)
rbRed=Radiobutton(root,text='Red',value=
1,variable=selColor,command=redSelected)
rbGreen=Radiobutton(root,text='Green',va
lue=2,variable=selColor,command=greenSel
ected)
rbBlue=Radiobutton(root,text='Blue',valu
e=3,variable=selColor,command=blueSelect
ed)

rbNoneColor.grid(column=0,row=2)
rbRed.grid(column=1,row=2)
rbGreen.grid(column=0,row=3)
rbBlue.grid(column=1,row=3)

figure = IntVar()
lblColor = Label(root,text='Select
Figure', font='bold')
lblColor.grid(column=0,row=4,columnspan=
2)
rbNoneFigure =
Radiobutton(root,text='All
figures',value=0,variable=figure)
rbTriangle=Radiobutton(root,text='Triang
le', value=3,variable=figure)
rbSquare=Radiobutton(root,text='Square',
value=4,variable=figure)
rbPentagon=Radiobutton(root,text='Pentag
on',value=5,variable=figure)
rbCircle=Radiobutton(root,text='Circle',
value=7,variable=figure)

rbNoneFigure.grid(column=0,row=5)
rbTriangle.grid(column=1,row=5)
```

ISSN 2531-2197

```
rbSquare.grid(column=0,row=6)
rbPentagon.grid(column=1,row=6)
rbCircle.grid(column=0,row=7)

btnFire=Button(root,text='Fire',width=20
, command=showVideo)
btnFire.grid(column=0,row=0,columnspan=2
)

lblVideo=Label(root)
lblVideo.grid(column=0,row=8,columnspan=
2,pady=10)
lblVideoProcesado=Label(root)
lblVideoProcesado.grid(column=0,row=8,co
lumnspan=2,pady=10)

root.mainloop()
```

Conclusions

This description of tools provides an overview of the key tools and libraries used in a project. Python, serves as the primary programming language for the project. OpenCV, an open-source computer vision and machine learning library, is utilized for various image processing tasks and has a large and active user community. VSCode, a free source code editor, offers a range of features and flexibility for software development, including support for Python programming and Git integration. GitLab, a web-based DevOps tool, provides a comprehensive solution for version control, issue tracking, and continuous integration/deployment pipelines. NumPy, a powerful Python library, facilitates efficient array manipulation and mathematical operations, particularly in the domain of data analysis.

Imutils, an extension of OpenCV, simplifies common image processing tasks with convenient functions. The Python Imaging Library (PIL) offers high-level commands for easy image manipulation. Tkinter, the standard Python interface to the Tcl/Tk GUI toolkit, enables the development of user-friendly graphical interfaces.

The methodology employed a sprint-based approach with four sprints. The initial sprints focused on setting up the development environment, installing dependencies, and developing a command-line interface with webcam integration. Subsequent sprints involved implementing color detection and selecting geometric figures and colors.

The combination of these tools, libraries, and methodologies provided a comprehensive framework for developing an image processing software application using Python.

References

Driscoll, M. (2021). *Pillow: Image Processing with Python.* Independently published.

Evertse, J. (2019). *Mastering GitLab 12.* Packt.

Howse, J., & Minichino, J. (2023). *Learning OpenCV 5 Computer Vision with Python: Tackle computer vision and machine learning with the newest tools, techniques and algorithms.* Packt Publishing.

Lentin, J. (2015). *Learning Robotics Using Python.* Packt.

McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, Numpy, and IPython.* Oreilly & Associates Inc.

Moore, A. D. (2021). *Python GUI Programming with Tkinter: Design and build functional and user-friendly GUI applications.* Packt Publishing.

Murph, D. (2020). *The GitLab Remote Playbook: Covering Fundamentals to Long-Term Strategy.* GitLab Inc.

Ortega, J. M., Faruque Sarker, D. O., & Washington, S. (2019). *Learning Python Networking: A complete guide to build and deploy strong networking capabilities using Python 3.7 and Ansible.* Packt Publishing.

Pajankar, A. (2017). *Raspberry Pi Image Processing Programming: Develop Real-Life Examples with Python, Pillow, and Scipy.* Apress.

Papercut Publications. (2022). *Python Coding & Programming.* United Kingdom: Papercut Publications.

Rajender, K. (2023). *Mastering Data Analysis with Python: A Comprehensive Guide to NumPy, Pandas, and Matplotlib.* Jamba Academy.

ISSN 2531-2197

Rosebrock, A. (2015). *I just open sourced my personal imutils package: A series of OpenCV convenience functions.* Obtenido de https://pyimagesearch.com/2015/02/02/just-open-sourced-personal-imutils-package-series-opencv-convenience-functions/

Roseman, M. (2019). *Modern Tkinter for Busy Python Developers: Quickly learn to create great looking user interfaces for Windows, Mac and Linux using Python's standard GUI toolkit.* Late Afternoon Press.

Speight, A. (2021). *Visual Studio Code for Python Programmers.* Wiley.

timgates42. (2022). *GitHub - PyImageSearch/imutils: A collection of useful functions to simplify common image processing tasks with OpenCV and Python.* Obtenido de https://github.com/PyImageSearch/imutils

Uzayr, S. B. (Ed.). (2023). *Mastering Visual Studio Code: A Beginner's Guide.* CRC Press.

ISSN 2531-2197