JSS MAHAVIDYAPEETHA
JSS SCIENCE AND TECHNOLOGY UNIVERSITY
SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING
JSS Technical Institutions Campus, Mysuru – 570006

"CAMOUFLAGED OBJECT DETECTION"

Mini project report submitted in partial fulfillment of curriculum prescribed for the Forensics and Cyber Law (CS665) course for the award of the degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

by

Rahul Singh.
(01JST18CS098)

Sanjay CB
(01JST18CS119)

Under the Guidance of

Dr. Trisiladevi C. Nagavi
Assistant Professor,
Dept.of CS &E, SJCE, JSS STU Mysore

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
JUNE 2021

JSS MAHAVIDYAPEETHA
JSS SCIENCE AND TECHNOLOGY UNIVERSITY

SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING
JSS Technical Institutions Campus, Mysuru – 570006

CERTIFICATE

This is to certify that the work entitled **"CAMOUFLAGED OBJECT DETECTION"** is a bonafied work carried out **by Rahul Singh and Sanjay CB** in partial fulfillment of the award of the degree of **Bachelor of Engineering in Computer Science and Engineering of SJCE, JSS Science and Technology, Mysuru during the year 2021**. It is certified that all corrections / suggestions indicated during CIE have been incorporated in the report. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work prescribed for the **Forensics and Cyber Law (CS665)** course.

Course in Charge and Guide

Dr. Trisiladevi C. Nagavi
Assistant Professor,
Dept.of CS &E,
SJCE, JSS STU Mysore

Place: Mysore **Date: 01/06/2021**

DECLARATION

I hereby declare that the project report entitled "**CAMOUFLAGED OBJECT DETECTION"** submitted by us to JSSSTU Mysore in partial fulfillment of the requirement for the award of the degree of B. TECH in COMPUTER SCIENCE DEPARTMENT is a record of project work carried out by us under the guidance of Dr. TRISILADEVI C. NAGAVI. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Names:	Signature of candidates:
Rahul Singh	singhrahul
Sanjay CB	sanjaycb

Date: 01st June 2021

ABSTRACT

We present a comprehensive study on a new task named camouflaged object detection (COD), which aims to identify objects that are “seamlessly” embedded in their surroundings. The high intrinsic similarities between the target object and the background make COD far more challenging than the traditional object detection task. To address this issue, we elaborately collect a novel dataset, calledCOD10K, which comprises 10,000 images covering camouflaged objects in various natural scenes, over 78 object categories. All the images are densely annotated with category, bounding-box, object-/instance-level, and matting level labels. This dataset could serve as a catalyst for progressing Many vision tasks, e.g., localization, segmentation, and alpha-matting, etc. In addition, we develop a simple but effective framework for COD, termed Search Identification Network (SINet).

ACKNOWLEDEMENT

We would like to express my special thanks of gratitude to my teacher Dr. Trisaladevi Nagavi, who gave us the opportunity to do this wonderful project of Forensics and Cyber Law on “CAMOUFLAGED OBJECT DETECTION”, who also helped us in completing my project. We came to know about so many new things, we are really thankful to them.

Secondly we would also like to thanks our friends who helped us a lot in finishing this project within the limited time. Also it helped us a lot in increasing our skills and knowledge as we were working on totally something.

So it was great working this project and it has surely pushed us and made us to a level up.

PROJECT TEAM DETAILS

1. 

RAHUL SINGH

USN: 01JST18CS098

Branch: COMPUTER SCIENCE ENGG

2. 

SANJAY CB

USN: 01JST18CS119

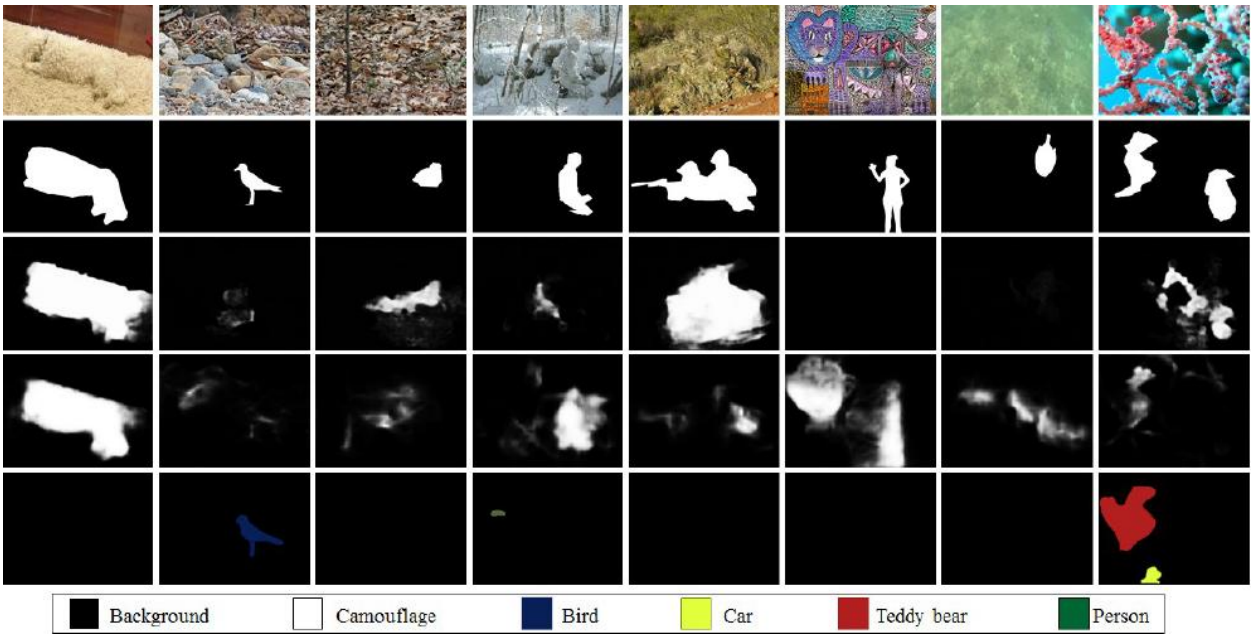
Branch: COMPUTER SCIENCE ENGG

TABLE OF CONTENTS

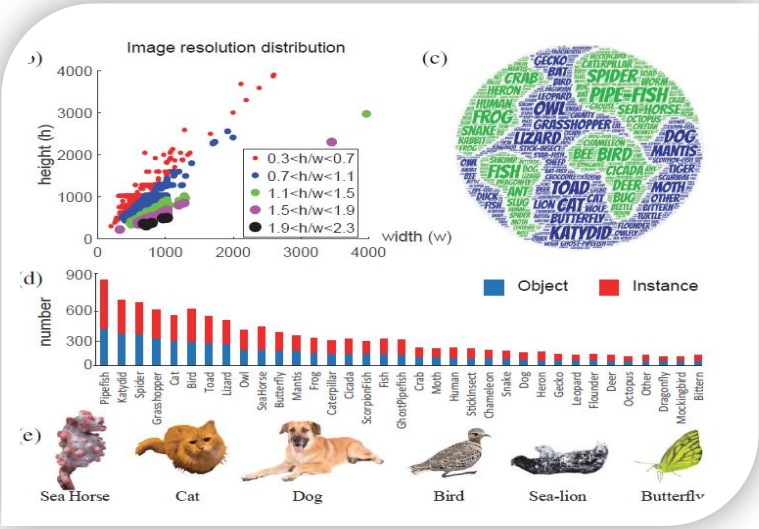
Chapter 1:	1: Introduction 1.1 Introduction to the problem domain 2. Aim/statement of the problem 1.3 Objectives of the project work 1.4 Applications
Chapter 2:	Tools and technology used
Chapter 3:	System design and implementation
Chapter 4:	System testing and results analysis
Chapter 5:	Conclusion and future work
References	

1.1Introduction to the problem domain

The term “camouflage” was originally used to describe the behaviour of an animal trying to hide itself from its surroundings to hunt or avoid being hunted, namely naturally camouflaged objects. Human adopted this and began to apply it widely on the battlefield. For example, soldiers and war equipment are applied the camouflage effect by dressing or coloring their appearance to blend them with their surroundings, namely artificially camouflaged objects. Autonomously detecting/segmenting camouflaged objects is thus a difficult task where discriminative features do not play an important role since we have to ignore objects that capture our attention. While detecting camouflaged objects is technically challenging on the one hand, it is beneficial in various practical scenarios, on the other, to include surveillance systems and search-and rescue missions.



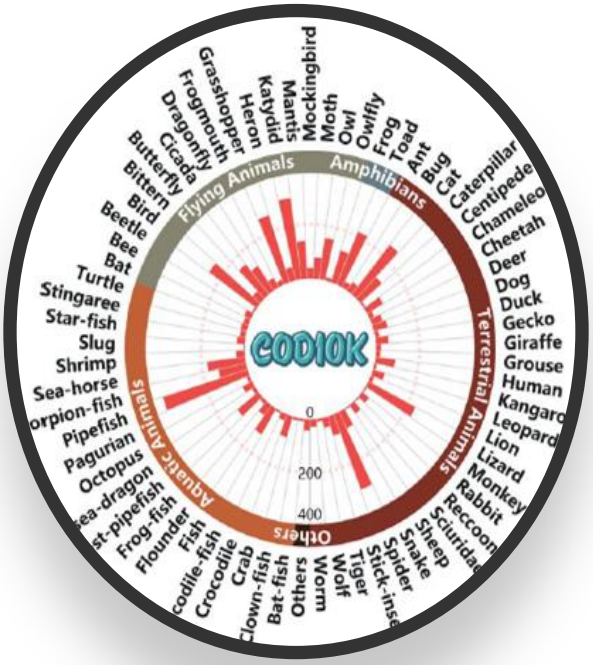
This figure shows several examples that camouflaged objects are failed to be detected by state-of-the-art object segmentation networks. Moreover, there are plenty of creatures in nature that have evolved over time to confuse themselves with their surroundings and even harder to be detected. Therefore, the study of detecting these less obvious objects which we call it camouflaged object in this project, is necessary in the field which targets detecting all objects in all scenes. Note that camouflage is a very subjective concept here. An object, such as a person which is regarded as a common class in MS-COCO, can be considered as a camouflaged object when he is hiding himself as a sniper. In other words, we treat all objects that are too similar to their surroundings because of their color, textureor both as camouflaged object and its complement set as non-camouflaged object. Moreover, what camouflage has in common is that its features are very little different from its surroundings. Therefore, it is reasonable to treat all objects of different classes but similar to the background as one class.



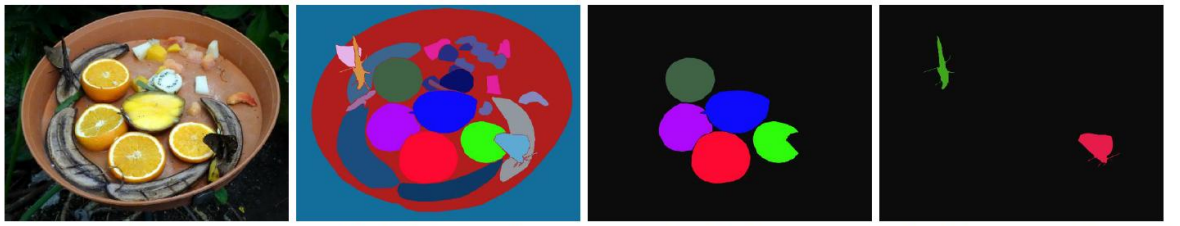
(b)Image resolution distribution. (c) Word cloud distribution. (d) Object/Instance number of several categories. (e) Examples of sub-classes.

Types of Camouflage

Camouflaged images can be roughly split into two types: those containing natural camouflage and those with artificial camouflage. Natural camouflage is used by animals (*e.g.*, insects, cephalopods) as a survival skill to avoid recognition by a predator. In contrast, artificial camouflage is usually occurs in products (so-called defects) during the manufacturing process, or is used in gaming/art to hide information.



Taxonomic system and its histogram distribution.



(a) Image (b) Generic object (c) Salient object (d) Camouflaged object

1.2 AIM

Can you find the concealed object(s) in each image of the above figure?

Well this is called as *background matching camouflage.*

Where an animal attempts to adapt their body's coloring to match "perfectly" with the surroundings in order to avoid recognition. Sensory ecologists have found that this camouflage strategy works by deceiving the visual perceptual system of the observer. Thus, addressing *camouflaged object detection* (COD) requires a significant amount of visual perception knowledge. As shown in the above figure, the high intrinsic similarities between the target object and the background make COD far more challenging than the traditional salient object detection or generic object detection. In addition to its scientific value, COD is also beneficial for applications in the fields of computer vision (for search and- rescue work, or rare species discovery), medical image segmentation (*e.g.*, polyp segmentation, lung infection segmentation), agriculture (*e.g.*, locust

detection to prevent invasion), and art (*e.g.*, for photo-realistic blending, or recreational art.

Currently, camouflaged object detection is not well-studied due to the lack of a sufficiently large dataset. To enable a comprehensive study on this topic, we provide two contributions. First, we carefully assembled the novel COD10K dataset exclusively designed for COD. It differs from current datasets in the following aspects:

• It contains 10K images covering 78 camouflaged object categories, such as *aquatic*, *flying*, *amphibians*, and *terrestrial*, *etc*.

• All the camouflaged images are *hierarchically annotated* with category, bounding-box, object-level, and

instance-level labels, facilitating many vision tasks, such as localization, object proposal, semantic edge detection, task transfer learning,*etc*.

• Each camouflaged image is assigned with *challenging attributes* found in the real-world and *matting level* labelling (requiring _60 minutes per image). These high-quality annotations could help with providing deeper insight into the performance of algorithms.

1.3 **Objectives of the project work**

The objective of our project is to find the camouflaged object. We all know that making the machine learn about the camouflaged objects and then training it for those many objects requires a high computing system and also a large dataset. For example there is dataset of ten thousand different animals which is used for finding the camouflaged animals in the background. So just a dataset of 10K animals the size of the dataset is 50-60GB which is a huge file. Now we can just think if all the objects in the world are chosen in the dataset then the size of the file will be very very large and task will be tedious. So this is the reason that we have restricted ourselves to Camouflaged Human detection. And in this project we will be doing that i.e. finding and alerting if the human is getting camouflaged with background.

1.4 Applications

Camouflage detection systems (CDS) have various possible

applications. Here, we envision two potential uses.

Medical Image Segmentation. If a medical image segmentation method was equipped with a CDS trained for

specific objects, such as polyp, it could be used to automatically segment polyps in nature to find & protect

rare species, or even in disaster areas for search and rescue.

Polyp detection/segmentation results

Search and rescue system working in a disaster area

Search Engines. Figureshows an example of search results from Google. From the results, we notice that the search engine cannot detect the concealed butterfly, and thus only provides images with similar backgrounds. Interestingly, when the search engine is equipped with a CDS (here, we just simply change the keyword), the engine can identify the camouflaged object and then feedback several butterfly images.

Chapter 2:Tools and technology used

We have used Windows operating system and python programming language to implement this project. We have also made use of Visual Studio Code to implement the camouflaged Object Detection project. **Visual Studio Code** (famously known as **VS Code**) is a free open source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS. Although the editor is relatively lightweight, it includes some powerful features that have made VS Code one of the most popular development environment tools in recent times.

Chapter 3 System design and implementation

Training : In the training part we are going to train a model for detecting the humans and create the model named SVM.py .To train model we use the image data set from INRAI image dataset.

Testing : The system is tested using the training model that is SVM.npy . Input is given in the form of video that is the video is divided into frames of about 250 images and if the background colour matches with the foreground colour then we say that camouflage is detected and a beep sound is set.

So in this way it helps in detecting the camouflaged object.

DATA FLOW DIAGRAM

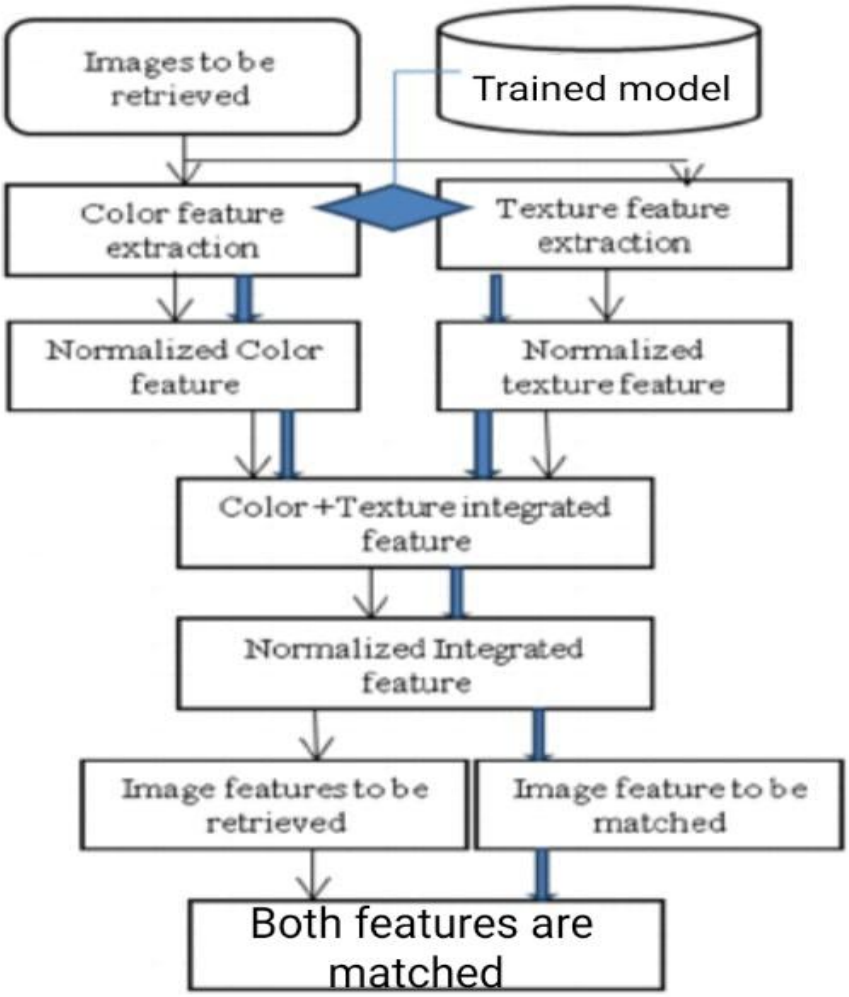


Fig. 9. Algorithm Scheme

Chapter 4: System testing and results analysis

CODE:: Training Part

```
from skimage.feature import hog
import joblib
from skimage import feature
import numpy as np
import os
from PIL import Image
from numpy import *
import warnings
warnings.filterwarnings("ignore")
from skimage import data

PERSON_WIDTH = 50
PERSON_HEIGHT = 100
leftop = [16, 16]
rightbottom = [16 + PERSON_WIDTH, 16 + PERSON_HEIGHT]

radius = 3
n_points = 8 * radius
METHOD = 'uniform'

def describe(image, radius, n_points,METHOD,eps=1e-7):

        lbp = feature.local_binary_pattern(image, n_points,
```

```
                radius, method="uniform")
            (hist, _) = np.histogram(lbp.ravel(),
                bins=np.arange(0, n_points + 3),
                range=(0, n_points + 2))


            hist = hist.astype("float")
            hist /= (hist.sum() + eps)


            return hist



pos_img_dir  = r"INRIAPerson/train_64x128_H96/pos/"

neg_img_dir = r"INRIAPerson/Train/neg/"

pos_img_files = os.listdir(pos_img_dir)
neg_img_files = os.listdir(neg_img_dir)

X = []
y = []
print('start loading ' + str(len(pos_img_files)) + ' positive files')
for pos_img_file in pos_img_files:
    pos_filepath = pos_img_dir + pos_img_file
    pos_img = data.imread(pos_filepath, as_grey=True)
```

```
    pos_roi = pos_img[leftop[1]:rightbottom[1], leftop[0]:rightbottom[0]]
    fd_hog = hog(pos_roi, orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(2, 2), visualise=False)
    fd_lbp = describe(pos_roi, radius, n_points,METHOD)
    fd = np.hstack([fd_lbp,fd_hog])
    X.append(fd)
    y.append(1)

print('start loading ' + str(len(neg_img_files)) + ' negative files')
for neg_img_file in neg_img_files:
    neg_filepath = neg_img_dir + neg_img_file
    neg_img = Image.open(neg_filepath)
    neg_img = neg_img.convert('L')
    neg_roi = neg_img.resize((50,100))
    fd_hog = hog(neg_roi, orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(2, 2), visualise=False)
    fd_lbp = describe(neg_roi, radius, n_points,METHOD)
    fd = np.hstack([fd_lbp,fd_hog])
    X.append(fd)
    y.append(0)

X = np.array(X)
y = np.array(y)
print(X.shape)
print(y.shape)
from sklearn import svm
print('start learning SVM.')
```

```
lin_clf = svm.LinearSVC()
lin_clf.fit(X, y)
print('finish learning SVM.')
print(lin_clf.fit(X,y))
print(lin_clf.score(X,y))
joblib.dump(lin_clf, 'modell_name.npy')
```

CODE:: Testing Part

```
from skimage.transform import pyramid_gaussian
from skimage.feature import hog
import joblib
import cv2
import os
import numpy as np
from skimage import feature
from imutils.object_detection import non_max_suppression
import warnings
warnings.filterwarnings("ignore")
import winsound
frequency=1500
duration=1000
```

```
def describe(image, radius, n_points,METHOD,eps=1e-7):

        lbp = feature.local_binary_pattern(image, n_points,
            radius, method="uniform")
        (hist, _) = np.histogram(lbp.ravel(),
            bins=np.arange(0, n_points + 3),
            range=(0, n_points + 2))

        # normalize the histogram
        hist = hist.astype("float")
        hist /= (hist.sum() + eps)



        return hist

def rgb2gray(im):
    gray = im[:, :, 0]*0.2989+im[:, :, 1]*0.5870+im[:, :, 2]*0.1140
    return gray

def sliding_window(image, window_size, step_size):

    for y in range(0, image.shape[0], step_size[1]):
        for x in range(0, image.shape[1], step_size[0]):
            yield (x, y, image[y:y + window_size[1], x:x + window_size[0]])

if _name_ == "_main_":
```

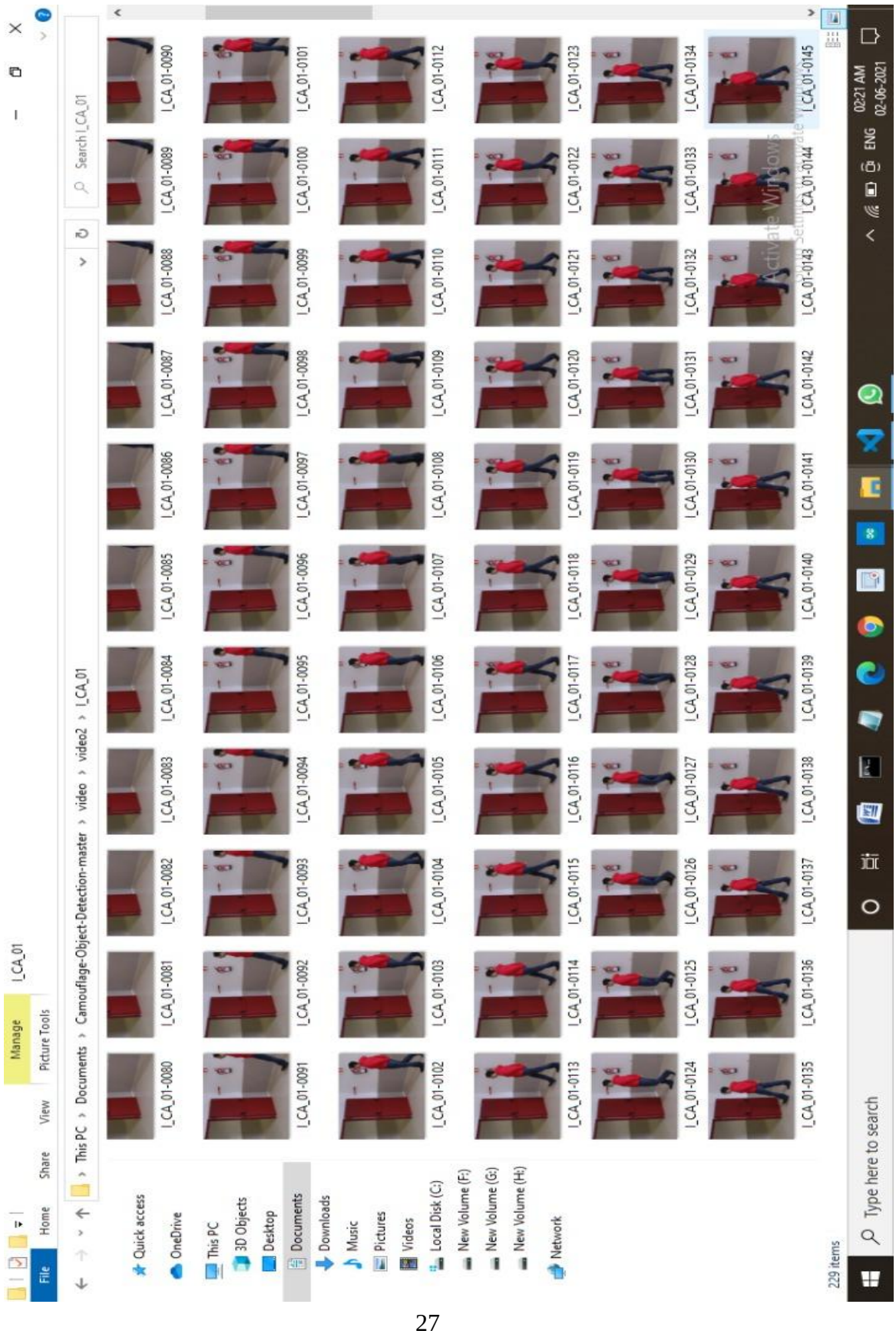
```
test_images_path = 'Video/video2/I_CA_01'
gt_images_path = 'Video/video2/I_CA_01-GT'
test_dir_list = os.listdir(test_images_path)
gt_dir_list = os.listdir(gt_images_path)
test_length = len(test_dir_list)
gt_length = len(gt_dir_list)
print("Total Images = ",test_length)

for i in range(0,test_length):
    downscale=1.25
    visualize = True

    im= cv2.imread(test_images_path+'/' + test_dir_list[i])
    im= cv2.resize(im,(352,288))
    img = im.copy()
    print("Processing Image ",i+1," of ",test_length)
    min_wdw_sz = (100, 200)
    step_size = (30, 30)
    visualize_det = visualize
    radius = 3
    n_points = 8 * radius
    METHOD = 'uniform'
    model_path = "svm_model.npy"


    # Load the classifier
    clf = joblib.load(model_path)
```

```
        detections = []

        scale = 0

        for im_scaled in pyramid_gaussian(im, downscale=downscale):

            cd = []
            for (x, y, im_window) in sliding_window(im_scaled, min_wdw_sz,
step_size):
                if im_window.shape[0] != min_wdw_sz[1] or im_window.shape[1]
!= min_wdw_sz[0]:
                    continue
                # Calculate the HOG features
                im_window = cv2.resize(im_window,(64,128))
                im_window = rgb2gray(im_window)
                fd_hog = hog(im_window,orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(2, 2), visualize=False)
                fd_lbp = describe(im_window, radius, n_points,METHOD)
                fd = np.hstack([fd_hog,fd_lbp])
                fd = fd.reshape(1,-1)
                pred = clf.predict(fd)
                if pred == 1:

                    detections.append((x, y,clf.decision_function(fd),
                    int(min_wdw_sz[0](downscale*scale)),
                    int(min_wdw_sz[1](downscale*scale))))
```

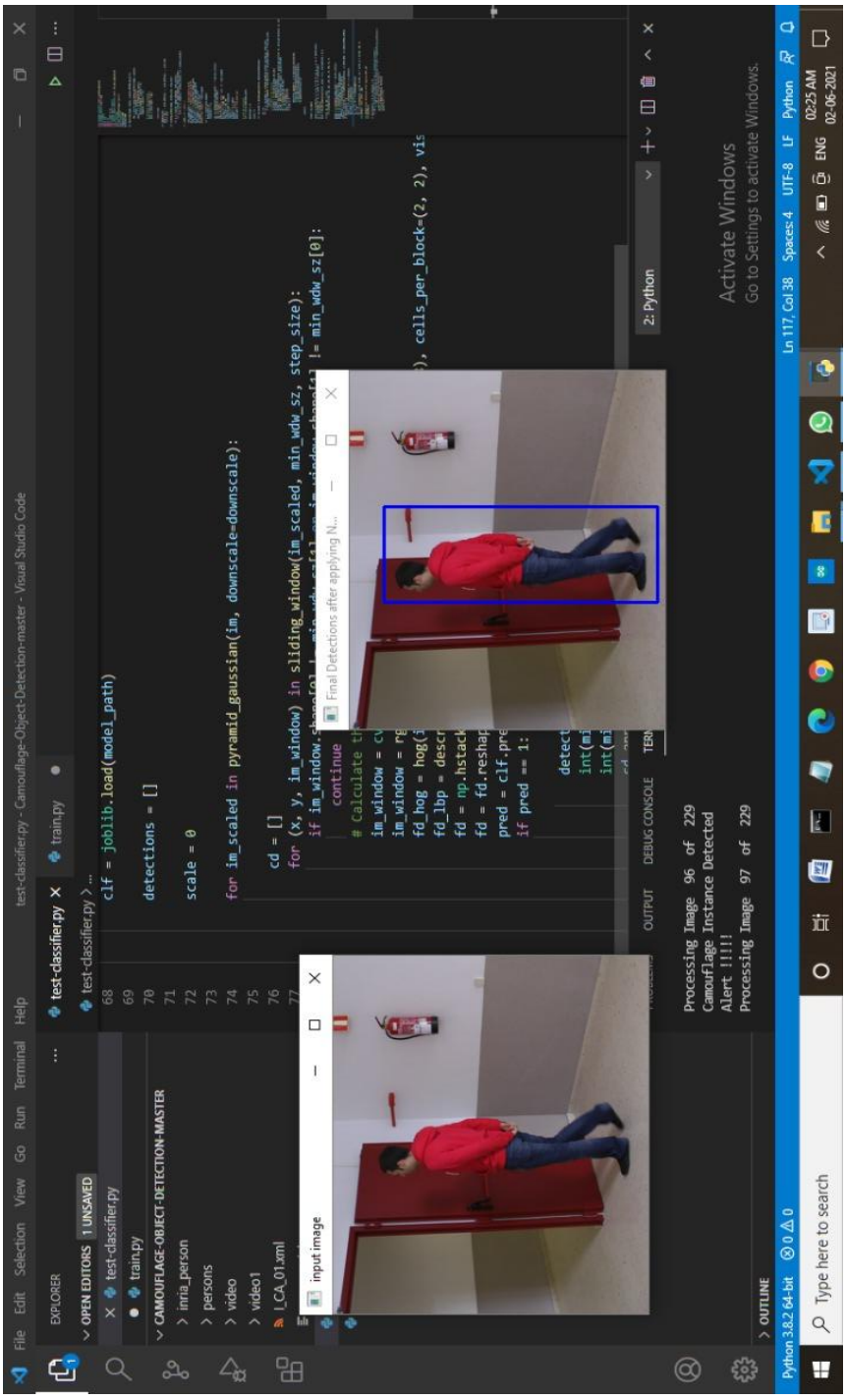
```
                cd.append(detections[-1])

            scale+=1


        clone = im.copy()

        rects = np.array([[x, y, x + w, y + h] for (x, y, _, w, h) in detections])
        sc = [score[0] for (x, y, score, w, h) in detections]
        sc = np.array(sc)
        pick = non_max_suppression(rects, probs = None, overlapThresh =
0.1)

        for(xA, yA, xB, yB) in pick:
            cv2.rectangle(clone, (xA, yA), (xB, yB), (0, 255, 0), 2)



        gtImg = cv2.imread(gt_images_path+'/' + gt_dir_list[i])
        gtImg = cv2.resize(gtImg,(352,288))
        gtImg = cv2.cvtColor(gtImg,cv2.COLOR_BGR2GRAY)
        _,gtThresh = cv2.threshold(gtImg,50,255,0)
        contours, hierarchy = cv2.findContours(gtThresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        if len(contours) > 0:
            areas = [cv2.contourArea(c) for c in contours]
            max_index = np.argmax(areas)
            cnt = contours[max_index]
```

```
            xc, yc, wc, hc = cv2.boundingRect(cnt)
            inc = 30
            im = cv2.rectangle(im, (xc-10, yc-10), (xc + wc+10, yc + hc+10),
255, 2)
        cv2.imshow(" input image", img)

        cv2.imshow("Final Detections after applying NMS",im)
        cv2.waitKey(1)

        if len(contours) > 0 and len(pick) > 0:
            if (xA in range(xc-inc,xc+wc//2) and yA in range(yc-inc,yc+inc)) \
                    or (xB in range(xc+wc-wc//2,xc+wc+inc) and yB in
range(yc+hc-inc,yc+hc+inc)) :
                print("Camouflage Instance Detected ")
                winsound.Beep(frequency, duration)
                print("Alert !!!!!")
```

INPUT:


I_CA_01
File
Home
Share
View
Manage
Picture Tools
This PC › Documents › Camouflage-Object-Detection-master › video › video2 › I_CA_01
Search I_CA_01
Quick access
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
New Volume (F:)
New Volume (G:)
New Volume (H:)
Network
I_CA_01-0080
I_CA_01-0081
I_CA_01-0082
I_CA_01-0083
I_CA_01-0084
I_CA_01-0085
I_CA_01-0086
I_CA_01-0087
I_CA_01-0088
I_CA_01-0089
I_CA_01-0090
I_CA_01-0091
I_CA_01-0092
I_CA_01-0093
I_CA_01-0094
I_CA_01-0095
I_CA_01-0096
I_CA_01-0097
I_CA_01-0098
I_CA_01-0099
I_CA_01-0100
I_CA_01-0101
I_CA_01-0102
I_CA_01-0103
I_CA_01-0104
I_CA_01-0105
I_CA_01-0106
I_CA_01-0107
I_CA_01-0108
I_CA_01-0109
I_CA_01-0110
I_CA_01-0111
I_CA_01-0112
I_CA_01-0113
I_CA_01-0114
I_CA_01-0115
I_CA_01-0116
I_CA_01-0117
I_CA_01-0118
I_CA_01-0119
I_CA_01-0120
I_CA_01-0121
I_CA_01-0122
I_CA_01-0123
I_CA_01-0124
I_CA_01-0125
I_CA_01-0126
I_CA_01-0127
I_CA_01-0128
I_CA_01-0129
I_CA_01-0130
I_CA_01-0131
I_CA_01-0132
I_CA_01-0133
I_CA_01-0134
I_CA_01-0135
I_CA_01-0136
I_CA_01-0137
I_CA_01-0138
I_CA_01-0139
I_CA_01-0140
I_CA_01-0141
I_CA_01-0142
I_CA_01-0143
I_CA_01-0144
I_CA_01-0145
229 items
Activate Windows
Type here to search
ENG
02:21 AM
02-06-2021

OUTPUT:


test-classifier.py - Camouflage-Object-Detection-master - Visual Studio Code
input image
Final Detections after applying N...
Processing Image 96 of 229
Camouflage Instance Detected
Alert !!!!!
Processing Image 97 of 229
Activate Windows
Go to Settings to activate Windows.

Chapter 5: Conclusion and future work

By implementing the algorithms like SINet. We can detect the object in more realistic way and we can almost detect almost all animals and in all condition but the real problem in implementing the project requires high eneded system which is having a high computation power.

Chapter 6: References

Optical camouflage using retro-reflective projection technology: https://ieeexplore.ieee.org/document/1240754

Camouflage image generation system for security: https://ieeexplore.ieee.org/document/7824827

M. Galun, E. Sharon, R. Basri, and A. Brandt. Texture segmentation by multiscale aggregation of filter responses and shape elements. In Proceedings of International Conference on Computer Vision, pages 716–723, Oct 2003

Camouflage: Memory Traffic Shaping to Mitigate Timing Attacks: https://ieeexplore.ieee.org/document/7920837

A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. In Proceedings of International Conference on Learning Representations, 2017

J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pages 3431– 3440, June 2015

Optical camouflage III: Auto-stereoscopic and multiple-view display system using retro-reflective projection technology: https://ieeexplore.ieee.org/document/6180880