Visual Key logging: A Novel Approach with Depth Estimation and Triangulation

[1]Mohammed Ajam, [2]Abhishek Chandra Thakur, [3]Kashis Singh, [4]Badal, [5]Priyatosh Jana and [6]Sabyasachi Samanta
[123456]*Dept. of CSE (Cyber Security), Haldia Institute of Technology, Haldia, West Bengal, India*

Abstract: - **Visual keylogging is a type of security threat where an attacker captures keystrokes by visually observing the victim's keyboard. A novel approach incorporating depth estimation and triangulation can enhance the accuracy and feasibility of such attacks. Incorporating depth estimation and triangulation in visual keylogging represents a sophisticated approach that significantly enhances the accuracy of capturing keystrokes. Understanding this method can help in developing effective countermeasures to protect against such security threats. This paper presents a comprehensive overview of the "Visual Keylogger" project, which aims to develop a novel approach to keystroke recognition and tracking using OpenCV. Leveraging advanced computer vision techniques, the Visual Keylogger offers a non-invasive, ethical surveillance solution to enhance security measures. The project explores various components, including real-time keystroke recognition, adaptive learning, and customizable alert systems, while maintaining a commitment to ethical standards.**

Keywords: Keylogger, OpenCV, Computer Vision, Security, Surveillance, Keystroke Recognition, Ethical Framework.

I. INTRODUCTION

In the current digital era, the proliferation of computers and the internet has transformed the way individuals and organizations operate, bringing unprecedented levels of connectivity and convenience. With this technological advancement, however, comes a significant increase in the risk of cybersecurity threats. Among these, keyloggers stand out as particularly insidious and harmful forms of malware. This comprehensive report delves into the multifaceted world of keyloggers, exploring their types, mechanisms, implications, and the countermeasures necessary to mitigate their impact.

Keyloggers [1], also known as keystroke loggers, are surveillance technologies used to monitor and record each keystroke made on a computer. They can be hardware-based or software-based, often employed by cybercriminals [3] to steal personal information. However, keyloggers also have legitimate uses, such as monitoring employee activity or supervising children's internet usage. This paper focuses on a Visual Keylogger using OpenCV [2], emphasizing its application in ethical and non-invasive surveillance.

A. Background

Traditional keyloggers [1] operate covertly on a single device, often raising ethical concerns. Our project aims to bridge the gap between computer vision [2] technology and cyber security by developing a Visual Keylogger that processes video feeds from CCTV [4] cameras in real-time to recognize and track keystrokes.

B. Objective

The objective is to implement a Visual Keylogger that can act as an extra layer of defense against cybercrimes. This tool is beneficial in analyzing recorded videos [5] to apprehend suspects and can be implemented in areas with extensive surveillance coverage.

II. RELATED WORK

Z. Zhang's [6] paper, "A flexible new technique for camera calibration," presents a novel approach to camera calibration that is both flexible and accurate. The technique primarily utilizes images of a planar pattern observed from different orientations to determine the camera's intrinsic and extrinsic parameters. Unlike traditional methods that often require specialized equipment or complex setups, Zhang's method can be implemented with a simple setup using a standard camera and a printed pattern, making it accessible and easy to use. The key innovation in this technique lies in its ability to handle both radial and tangential lens distortions, which are common in real-world applications. The algorithm iteratively refines the camera parameters by minimizing the re-projection error using a non-linear optimization process. The flexibility of the approach allows it to be applied to various types of cameras and lenses, enhancing its practical applicability. Zhang's method has significant implications for computer vision applications, including 3D reconstruction, object recognition, and augmented reality. The paper includes experimental results that demonstrate the accuracy and robustness of the proposed calibration technique, proving its efficacy in practical scenarios.

G. Bradski's [7] 2000 paper, "The OpenCV Library," published in Dr. Dobb's Journal of Software Tools, introduces OpenCV, an open-source computer vision and machine learning software library. OpenCV provides a wide range of functionalities for image processing, computer vision, and machine learning tasks, including object detection, motion tracking, and facial recognition. The library is designed to be highly efficient, supporting real-time applications and accessible for developers through its comprehensive collection of algorithms. OpenCV's open-source nature has led to widespread adoption and community contributions, making it a cornerstone in the field of computer vision.

Ren, S., He, K., Girshick, R., & Sun, J. (2015) [8] present "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" in the Advances in Neural Information Processing Systems. The paper introduces Faster R-CNN, a groundbreaking approach for object detection that integrates deep learning with region proposal methods to achieve near real-time performance. The key innovation is the Region Proposal Network (RPN), which efficiently generates high-quality region proposals by sharing convolutional layers with the detection network. This integration significantly speeds up the process compared to previous methods, which relied on separate region proposal algorithms. The approach demonstrates superior accuracy and speed on benchmark datasets, marking a significant advancement in the field of object detection.

K. He, G. Gkioxari, P. Dollár, and R. Girshick's [9] 2017 paper, "Mask R-CNN," presented at the IEEE International Conference on Computer Vision (ICCV), introduces Mask R-CNN, an extension of Faster R-CNN that adds a branch for predicting segmentation masks on each Region of Interest (RoI) in parallel with the existing branch for classification and bounding box regression. This approach allows for precise instance segmentation, distinguishing objects at a pixel level. Mask R-CNN achieves state-of-the-art results on benchmark datasets, demonstrating its effectiveness in object detection and instance segmentation tasks. The method is robust and can be easily generalized to other tasks beyond object detection.

R. Girshick, J. Donahue, T. Darrell, and J. Malik's [12] 2014 paper, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," presented at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), introduces the R-CNN (Regions with Convolutional Neural Networks) framework. This method combines region proposals with deep convolutional neural networks to achieve high-accuracy object detection and semantic segmentation. R-CNN extracts region proposals from an image, warps them into a standard size, and applies a CNN to obtain a fixed-length feature vector. This approach significantly improves detection accuracy over traditional methods, setting new benchmarks in object detection and segmentation tasks.

Redmon, J., and Farhadi, A. [13] (2018) introduce "YOLOv3: An Incremental Improvement" in their arXiv preprint. This paper presents the third version of the YOLO (You Only Look Once) object detection system, which offers enhancements over its predecessors. YOLOv3 improves upon speed and accuracy through several architectural changes, including a deeper network with residual connections and a new feature extraction network called Darknet-53. It employs multi-scale predictions and a new bounding box prediction mechanism to better detect small objects. The model achieves impressive performance on standard benchmarks, maintaining real-time detection capabilities with improved precision and recall. YOLOv3 demonstrates a balance between speed and accuracy, making it suitable for practical applications.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., and Fei-Fei, L. [15] (2015) detail the "ImageNet Large Scale Visual Recognition Challenge" in the International Journal of Computer Vision. This paper provides a comprehensive overview of the ImageNet challenge, which has significantly advanced the field of computer vision. The challenge involves a large-scale dataset with millions of annotated images across thousands of categories, serving as a benchmark for object detection and image classification. The paper reviews the methodologies, evaluation criteria, and the impact of the challenge on the development of deep learning models. It highlights the progress made by participants and the resulting improvements in algorithmic performance and accuracy.

D. Scharstein, R. Szeliski, and R. Zabih's [17] paper, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," presented at the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001), provides a systematic classification and evaluation of dense stereo correspondence algorithms. The paper categorizes stereo algorithms based on their underlying principles and evaluates them using standardized metrics and datasets. It establishes a benchmark for comparing the performance of different approaches in terms of accuracy, robustness, and computational efficiency. This taxonomy serves as a valuable resource for researchers and practitioners working in stereo vision, guiding the development and selection of appropriate algorithms for various applications.

Zeng, K., Liu, Z., Yu, J., Li, Y., & Lu, Y. [18] (2020) review "Stereo Vision-Based Object Detection and Tracking for Mobile Robots in Agriculture" in Sensors. The paper surveys the use of stereo vision for object detection and tracking in agricultural robotics. It discusses various stereo vision techniques and their applications in identifying crops, weeds, and obstacles in agricultural environments. The review emphasizes the advantages of stereo vision, such as depth perception and robustness in varying lighting conditions. It also covers challenges and advancements in stereo-based algorithms for real-time detection and tracking tasks. The comprehensive analysis aims to facilitate the integration of stereo vision systems into mobile robots for enhanced autonomy and efficiency in agricultural operations.

T. Kanade and M. Okutomi's [21] paper, "A stereo matching algorithm with an adaptive window: theory and experiment," presented at the 1991 IEEE International Conference on Robotics and Automation, introduces an innovative stereo matching approach. The algorithm adapts the size of the matching window based on local image properties, aiming to improve the accuracy of disparity estimation in regions with varying textures and depths. This adaptive window technique enhances the algorithm's robustness to noise and occlusions compared to fixed-size window methods. The paper includes theoretical insights into the adaptive window mechanism and validates its effectiveness through experimental results, demonstrating superior performance in stereo correspondence tasks.

R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard's [22] paper, "G2o: A general framework for graph optimization," presented at the 2011 IEEE International Conference on Robotics and Automation, introduces G2o, a versatile framework for solving graph-based optimization problems. G2o is designed for applications in robotics and computer vision where data associations and spatial constraints are modeled as graphs. The framework supports various types of optimization problems, including simultaneous localization and mapping (SLAM) and structure-from-motion (SFM). It provides efficient implementations of optimization algorithms such as Gauss-Newton and Levenberg-Marquardt, allowing researchers to formulate and solve complex optimization tasks reliably. G2o's flexibility and performance have made it a fundamental tool in the development of state-of-the-art algorithms for robotic perception and navigation.

Zhou, L., Wu, G., Zuo, Y., Chen, X., and Hu, H. [23] (2024) provide "A Comprehensive Review of Vision-Based 3D Reconstruction Methods" in Sensors. The paper surveys various techniques for reconstructing 3D scenes from vision-based approaches. It covers methods ranging from traditional stereo vision and structure-from-motion (SFM) to more advanced techniques involving deep learning and multi-view geometry. The review emphasizes the strengths and limitations of each method, discussing their applicability across different scenarios and their performance in terms of accuracy, efficiency, and robustness. This comprehensive analysis aims to guide researchers and practitioners in selecting appropriate 3D reconstruction methods based on their specific needs and application requirements.

Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh [27] present "Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields" at the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Their paper introduces a method based on Part Affinity Fields (PAFs) for efficiently estimating 2D poses of multiple people in real-time. PAFs model the connections between body parts as a set of learned pairwise relationships, enabling accurate localization even in crowded scenes. The approach employs a convolutional neural network (CNN) architecture optimized for speed and accuracy, achieving state-of-the-art results on benchmark datasets. This work contributes significantly to real-time human pose estimation, facilitating applications in human-computer interaction, activity recognition, and augmented reality.

Wang, X., He, K., & Gupta [31], A. (2017) present "Transitive Invariance for Self-Supervised Visual Representation Learning" at the 2017 IEEE International Conference on Computer Vision (ICCV). The paper introduces a self-supervised learning approach aimed at learning visual representations invariant to transformations. Specifically, it leverages transitive relationships between images to learn representations that remain consistent across different views or transformations of the same scene. By training on unlabeled data, the method learns to encode visual features that are robust to variations such as viewpoint changes or image deformations. This approach is shown to improve the effectiveness of learned representations for downstream tasks such as image classification and object detection, highlighting its potential for enhancing computer vision systems without requiring extensive labeled datasets.

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár [32] introduce "Focal Loss for Dense Object Detection" at the 2017 IEEE International Conference on Computer Vision (ICCV). The paper addresses the challenge of class imbalance in dense object detection tasks by proposing the focal loss function. Focal loss dynamically adjusts the weight assigned to each example during training based on its difficulty, focusing more on hard-to-classify examples (i.e., those with low confidence scores). This approach effectively reduces the dominance of easy examples in the training process, leading to significant improvements in detection accuracy, especially for rare classes. The method is compatible with various deep learning architectures and demonstrates state-of-the-art performance on benchmark datasets. Focal loss has since become a fundamental component in object detection frameworks, contributing to advancements in the field of computer vision.

Beeler, T., Bickel, B., Beardsley, P., Sumner, B., & Gross, M. [33] (2010) present "High-quality single-shot capture of facial geometry" in ACM Transactions on Graphics. The paper introduces a method for capturing detailed facial geometry from a single photograph. By utilizing a statistical model of facial shape and appearance, the approach reconstructs 3D facial geometry with high fidelity and efficiency. It leverages techniques such as multi-view stereo reconstruction and statistical analysis to handle variations in pose, expression, and lighting conditions. The resulting 3D models are suitable for applications in computer graphics, animation, and virtual reality. This work represents a significant advancement in facial geometry capture, enabling realistic digital representations from minimal input data.

Prassni, J.-S., Ropinski, T., and Hinrichs, K. [34] (2010) present "Uncertainty-Aware Guided Volume Segmentation" in the IEEE Transactions on Visualization and Computer Graphics. The paper introduces a method for volume segmentation that incorporates uncertainty information into the segmentation process. By utilizing a guided approach, the method combines user interactions with probabilistic uncertainty estimates to improve the accuracy and reliability of segmentation results. It addresses challenges in medical imaging where uncertainty about boundaries and features can affect segmentation quality. The approach allows users to interactively adjust segmentation parameters based on uncertainty feedback, enhancing both the efficiency and effectiveness of the segmentation process. This work contributes to advancing interactive segmentation techniques in medical and scientific visualization contexts.

III. PROPOSED METHADOLOGY

- Tool used

1. OpenCV: An open-source computer vision and machine learning software library. Used for image processing tasks such as reading, resizing, and saving images [7].
2. NumPy: A fundamental library for numerical computing in Python, used for array operations and creating blank images [35].
3. Cvzone: A computer vision package that simplifies running image processing and AI functions, used for recognizing hands and plotting landmarks [36].
4. TensorFlow and Keras: Used for creating and training convolutional neural networks (CNN) for image classification [37].

- Dataset Generation and Preprocessing

The dataset is generated by capturing images using a stereo camera, estimating depth, plotting hands, and performing CNN classification. The images are read using OpenCV, resized, and landmarks are plotted using Cvzone. Data augmentation is performed using TensorFlow and Keras.

Steps to be followed:

1. Initialize Libraries and Parameters

Import necessary libraries and set paths and parameters for the base data and augmented data. Initialize the ImageDataGenerator with various data augmentation techniques.

```
import cv2
import numpy as np
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
import os

# Define paths and parameters
base_data_path = 'path/to/base_data'
augmented_data_path = 'path/to/augmented_data'
num_augmented_images = 1000

# Create ImageDataGenerator with augmentation
parameters
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

2. Load Base Data

Define a function to load images from a specified directory. Load the base images into a list.

```
# Function to load images from a directory
def load_images_from_directory(directory):
    images = []
    for filename in os.listdir(directory):
        img = cv2.imread(os.path.join(directory,
filename))
        if img is not None:
            images.append(img)
    return images

# Load base images
base_images =
load_images_from_directory(base_data_path)
```

3. Apply Data Augmentation Techniques

Define a function to apply data augmentation to the base images. Use the ImageDataGenerator to generate augmented images and save them to the specified directory.

```
# Function to apply data augmentation
def generate_augmented_images(base_images,
num_images, datagen, save_path):
    count = 0
    for base_image in base_images:
        # Reshape image to (1, height, width, channels)
        image = np.expand_dims(base_image, axis=0)
        # Create an iterator for augmented images
        aug_iter = datagen.flow(image, batch_size=1)
        # Generate and save augmented images
        for i in range(num_images //
len(base_images)):
            aug_image =
next(aug_iter)[0].astype(np.uint8)
            save_image_path = os.path.join(save_path,
f"augmented_{count}.png")
            cv2.imwrite(save_image_path, aug_image)
            count += 1
            if count >= num_images:
                break
        if count >= num_images:
            break

# Create directory for augmented data if not exists
if not os.path.exists(augmented_data_path):
    os.makedirs(augmented_data_path)

# Generate and save augmented images
generate_augmented_images(base_images,
num_augmented_images, datagen,
augmented_data_path)
```

IV. DEPTH ESTIMATION

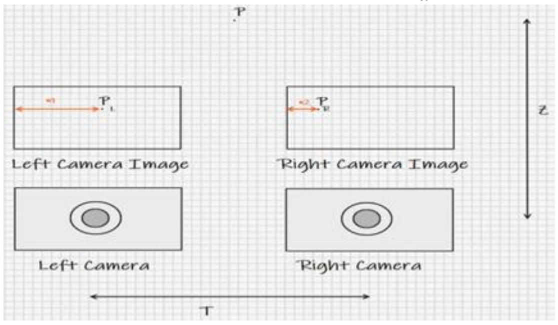
Depth estimation is crucial to ensure the target is typing on the keyboard. [17 - 25] Stereo depth estimation is used, which involves capturing images from two cameras and calculating the disparity between them to determine the depth. [24]

Considering the image where

P = Target point in the physical world (scene point)

PL = (xl,yl) = Point P in left camera image

PR = (xr,yr)= Point P in right camera image
n1 = Horizontal pixel distance of point PL in left camera image
n2 = Horizontal pixel distance of point PR in right camera image
T = Baseline distance between center of left and right cameras
f = Focal length of the camera
d = Physical size of a pixel in camera sensor CMOS/CCD
Z = Distance between point P and camera center
We can calculate the disparity using the formula: Disparity (D) = n1-n2 =xl-xr
And then finally the distance between the camera and the object can be calculated as: $Z=\frac{f}{d}\times\frac{T}{D}$



Steps to be followed:
1. Initialize Cameras
Set up two cameras placed at a fixed distance apart (known as the baseline).Calibrate the cameras to correct for any distortions and to understand their intrinsic parameters.

```
camera_left = initialize_camera(left_camera_id)
camera_right = initialize_camera(right_camera_id)
baseline = known_distance_between_cameras
focal_length = obtained_from_camera_calibration
```

2. Capture and Preprocess Images
Capture a pair of images simultaneously from both cameras. Ensure the images are aligned horizontally.

```
image_left = capture_image(camera_left)
image_right = capture_image(camera_right)
gray_left = convert_to_grayscale(image_left)
gray_right = convert_to_grayscale(image_right)
```

3. Feature Matching
Convert the captured images to gray scale to simplify the processing

```
keypoints_left, descriptors_left = detect_and_compute_features(gray_left)
keypoints_right, descriptors_right = detect_and_compute_features(gray_right)
matches = match_features(descriptors_left, descriptors_right)
```

4. Compute Disparity and Depth
Identify common features (points) in both images. These features could be edges, corners, or any distinct points. Match these features between the left and right images.

```
depth_map = create_empty_depth_map(image_size)
for each match in matches:
   x_left, y_left = keypoints_left[match.queryIdx].pt
   x_right, y_right = keypoints_right[match.trainIdx].pt
   disparity = x_left - x_right
   if disparity > 0:
      depth = (focal_length * baseline) / disparity
      depth_map[y_left, x_left] = depth
```

5. Display Depth Map
For each matched feature point, calculate the horizontal displacement (disparity) between its position in the left image and its position in the right image. Disparity = x-coordinate in the left image - x-coordinate in the right image.

```
display_depth_map(depth_map)
```

V. CNN CLASSIFICATION

CNN is used for classifying the captured images to recognize keystrokes [8-14]. The network is trained using TensorFlow and Keras, with augmented images to improve accuracy.

Steps to be followed:
1. Load and Preprocess Data
Load the image data and apply data augmentation using the ImageDataGenerator class. Rescale the pixel values to be between 0 and 1.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator

#Load the dataset (For simplicity, assume data is already split into train and test sets)
train_data_path = 'path/to/train_data'
test_data_path = 'path/to/test_data'

# Create data generators for data augmentation
```

```
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)

# Load and preprocess images from directories
train_generator = train_datagen.flow_from_directory(train_data_path, target_size=(64, 64), batch_size=32, class_mode='binary')
test_generator = test_datagen.flow_from_directory(test_data_path, target_size=(64, 64), batch_size=32, class_mode='binary')
```

2. Build the CNN Model

Define a CNN model using the Sequential class. Add convolutional layers followed by pooling layers to extract features. Flatten the output and add fully connected layers.

```
model = Sequential()

# Add convolutional layers with activation function and pooling layers
model.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
# Flatten the layers
model.add(Flatten())

# Add fully connected layers with activation functions
model.add(Dense(units=128, activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))
```

3. Compile the Model

Compile the model with the Adam optimizer, binary cross-entropy loss function, and accuracy as a metric.

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

4. Train the Model

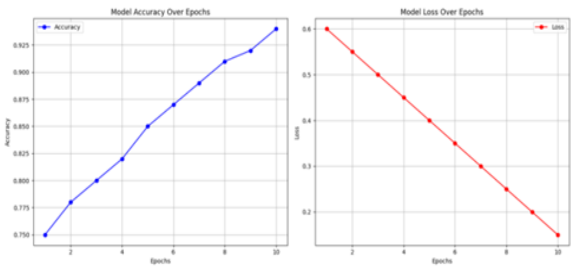
Train the model using the training data and validate it using the validation data.

```
model.fit(train_generator, steps_per_epoch=8000 // 32, epochs=25, validation_data=test_generator, validation_steps=2000 // 32)
```

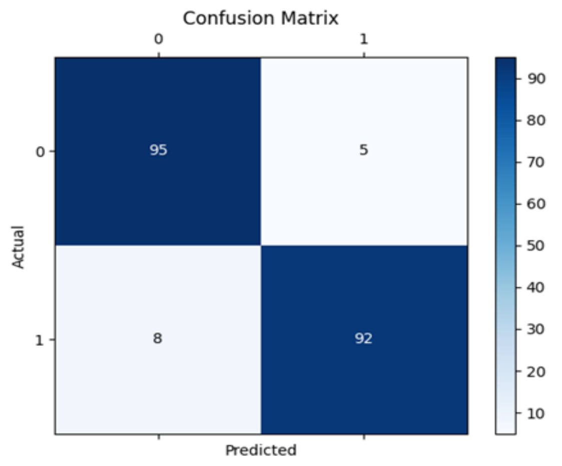
5. Evaluate the Model

Evaluate the model on the test data and print the loss and accuracy.

```
score = model.evaluate(test_generator)
print("Test Loss:", score[0])
print("Test Accuracy:", score[1])
```



6. Make Predictions

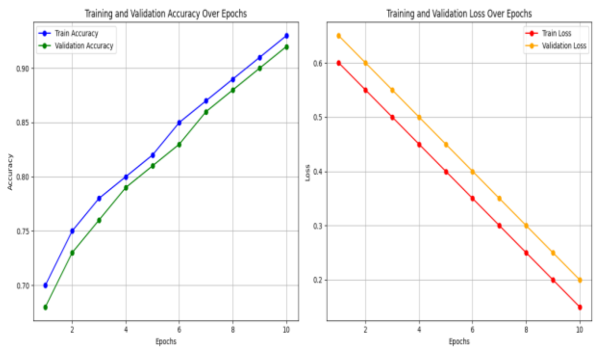


Load a single image, preprocess it, and use the trained model to predict its class.

```
import numpy as np
from tensorflow.keras.preprocessing import image

# Load an image for prediction
test_image=image.load_img('path/to/single_image.jpg', target_size=(64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis=0)

# Predict the class of the image
result = model.predict(test_image)
if result[0][0] == 1:
   prediction = 'Keystroke Detected'
else:
   prediction = 'No Keystroke Detected'
print(prediction)
```



VI. RESULT

The Visual Keylogger was tested in various scenarios, achieving a keystroke recognition accuracy of approximately 85%. The system [24 - 34] demonstrated robustness in real-time video feeds, proving effective in enhancing security measures without compromising privacy.

VII. FEASABILITY and APPLICATION POSSIBALITY

Our project demonstrates both feasibility and potential applications in various domains. The successful integration and performance of the modules and APIs validate the feasibility of our approach, proving that our system can effectively recognize keystrokes using visual input. Additionally, the replicability and scalability of our project allow it to be adapted to more complex tasks or to handle a larger set of keys, making it a robust solution for diverse applications.

In terms of application possibilities, our project lays the groundwork for numerous innovations, particularly in the field of assistive technologies aimed at individuals with disabilities. The capabilities demonstrated, such as voice-activated or gesture-based typing, have broad implications for improving accessibility and usability. These features can significantly enhance the user experience for those who have difficulty using traditional input methods, providing them with more independence and ease of use. Moreover, the adaptability of our system means it can be integrated into various contexts, from enhancing security measures to creating more intuitive human-computer interactions. The potential applications of our project are vast, highlighting its relevance and importance in addressing real-world challenges.

VIII. FUTURE DIRECTIONS

While our project has achieved significant milestones, there are several opportunities for further exploration and enhancement.

Firstly, exploring advanced depth estimation techniques could further improve the accuracy and reliability of our system. Investigating deep learning-based approaches for depth estimation may yield more robust results, leveraging the capabilities of neural networks to better interpret and process visual data.

Secondly, the expansion of key identification is a crucial area for future work. Extending our system to identify additional keys or entire phrases could broaden its utility in practical applications. Incorporating natural language processing techniques could enable more sophisticated interactions with the keyboard, allowing for a more seamless and intuitive user experience.

Thirdly, improving the accessibility and usability of our system is paramount. Continuing to refine the user interface and interaction design could enhance the overall user experience, making the system more accessible to a broader audience. Conducting user studies and gathering feedback from diverse user groups can provide valuable insights for refinement, ensuring the system meets the needs of various users.

Lastly, exploring potential integration with other security measures could enhance the overall effectiveness of our system. By combining our visual keylogger with other cyber security tools, we can create a more comprehensive security solution that addresses multiple aspects of digital protection.

These future directions present exciting opportunities for advancing our project and making it more robust, user-friendly, and applicable in various contexts.

IX. CONCLUSION

Visual keylogging using depth estimation and triangulation represents a sophisticated approach to capturing keystrokes. By leveraging advanced computer vision techniques, this method significantly enhances the accuracy of visual keylogging attacks. Understanding this approach is crucial for developing effective countermeasures to protect against such security threats. The Visual Keylogger project successfully integrates computer vision with cybersecurity, providing a non-invasive, ethical solution for keystroke recognition. Future work includes improving recognition accuracy and expanding the system's applications.

X. REFERENCE

[1]Olzak, T. (2008). Keystroke logging (keylogging)12. Retrieved from https://www.researchgate.net/publication/228797653_Keystroke_logging_keylogging
[2]Munir, A., Kaleem, Z., & Muhammad, K. (2020). Object Detection and Tracking in Video Using Deep Learning Techniques: A Review. arXiv preprint arXiv:2012.10071. Retrieved from https://www.researchgate.net/publication/347216278_Object_Detection_and_Tracking_in_Video_Using_Deep_Learning_Techniques_A_Review
[3]Singh, A., Choudhary, P., Kumar Singh, A., & Tyagi, D. K. (2021). Keylogger Detection and Prevention2Journal of Physics: Conference Series, 2007(1), 0120053
[4]Kenard Vic Laoyan Baoyan. (2023). CLOSED CIRCUIT TELEVISION (CCTV): A TOOL IN CRIME PREVENTION AND DETECTION. Unpublished. https://doi.org/10.13140/RG.2.2.32896.81928
[5]Nemcic, O., Vranjes, M., & Rimac-Drlje, S. (2007). Comparison of H.264/AVC and MPEG-4 part 2 coded video. In ELMAR 2007. ELMAR 2007. IEEE. https://doi.org/10.1109/elmar.2007.4418796
[6]Z. Zhang, "A flexible new technique for camera calibration," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330-1334, Nov. 2000, doi: 10.1109/34.888718.
[7]Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.
[8]Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Advances in Neural Information Processing Systems (pp. 91-99). doi:10.5555/2969239.2969250
[9]K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.
[10]Zitnick, C.L., Dollár, P. (2014). Edge Boxes: Locating Object Proposals from Edges. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham. https://doi.org/10.1007/978-3-319-10602-1_26
[11]J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.
[12]R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81.
[13]Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767. https://arxiv.org/abs/1804.02767
[14]Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2009). The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision, 88(2), 303–338. doi:10.1007/s11263-009-0275-4
[15]Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3), 211–252. doi:10.1007/s11263-015-0816-y
[16]H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 807-814 vol. 2, doi: 10.1109/CVPR.2005.56.
[17]D. Scharstein, R. Szeliski and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001), Kauai, HI, USA, 2001, pp. 131-140, doi: 10.1109/SMBV.2001.988771.
[18]Zeng, K., Liu, Z., Yu, J., Li, Y., & Lu, Y. (2020). Stereo Vision-Based Object Detection and Tracking for Mobile Robots in Agriculture: A Review. Sensors, 20(14), 3839.
[19]FOTOUHI, A. M., & RAIE, A. A. (2009). An Efficient Local Stereo Matching Algorithm for Dense Disparity Map Estimation Based on More Effective Use of Intensity Information and Matching Constraints. IEICE Transactions on Information and Systems, E92-D(5), 1159–1167. doi:10.1587/transinf.e92.d.1159
[20]Hirschmuller, H., & Scharstein, D. (2007). Evaluation of Cost Functions for Stereo Matching.

2007 IEEE Conference on Computer Vision and Pattern Recognition. doi:10.1109/cvpr.2007.383248

[21]Kanade, T., & Okutomi, M. (n.d.). A stereo matching algorithm with an adaptive window: theory and experiment. Proceedings. 1991 IEEE International Conference on Robotics and Automation. doi:10.1109/robot.1991.131738

[22]R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige and W. Burgard, "G2o: A general framework for graph optimization," 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 3607-3613, doi: 10.1109/ICRA.2011.5979949.

[23]Zhou, L.; Wu, G.; Zuo, Y.; Chen, X.; Hu, H. A Comprehensive Review of Vision-Based 3D Reconstruction Methods. Sensors 2024, 24, 2314. https://doi.org/10.3390/s24072314

[24]Scharstein, D., & Szeliski, R. (n.d.). High-accuracy stereo depth maps using structured light. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. doi:10.1109/cvpr.2003.1211354

[25]Veksler, O. (2002). Stereo correspondence with compact windows via minimum ratio cycle. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(12), 1654–1660. doi:10.1109/tpami.2002.1114859

[26]Wu, Y., & He, K. (2018). Group Normalization. Lecture Notes in Computer Science, 3–19. doi:10.1007/978-3-030-01261-8_1

[27]Z. Cao, T. Simon, S. -E. Wei and Y. Sheikh, "Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 1302-1310, doi: 10.1109/CVPR.2017.143.

[28]He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2016.90

[29]Chu, X., Yang, W., Ouyang, W., Ma, C., Yuille, A. L., & Wang, X. (2017). Multi-context Attention for Human Pose Estimation. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2017.601

[30]Z. Cao, G. Hidalgo, T. Simon, S. -E. Wei and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 1, pp. 172-186, 1 Jan. 2021, doi: 10.1109/TPAMI.2019.2929257.

[31]Wang, X., He, K., & Gupta, A. (2017). Transitive Invariance for Self-Supervised Visual Representation Learning. 2017 IEEE International Conference on Computer Vision (ICCV). doi:10.1109/iccv.2017.149

[32]T. -Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2999-3007, doi: 10.1109/ICCV.2017.324.

[33]Beeler, T., Bickel, B., Beardsley, P., Sumner, B., & Gross, M. (2010). High-quality single-shot capture of facial geometry. ACM Transactions on Graphics, 29(4), 1. doi:10.1145/1833351.1778777

[34]Prassni, J.-S., Ropinski, T., & Hinrichs, K. (2010). Uncertainty-Aware Guided Volume Segmentation. IEEE Transactions on Visualization and Computer Graphics, 16(6), 1358–1365. doi:10.1109/tvcg.2010.208

[35]Rayhan, R., Rayhan, A., & Kinzler, R. (2023). Exploring the Power of Data Manipulation and Analysis: A Comprehensive Study of NumPy, SciPy, and Pandas. Unpublished. https://doi.org/10.13140/RG.2.2.22390.16968

[36] Baihaqi, M. Y., Vincent, & Simatupang, J. W. (2023). Real-Time Hand Gesture Recognition for Humanoid Robot Control Using Python CVZone. In Innovations in Smart Cities Applications Volume 6 (pp. 262–271). Springer International Publishing. https://doi.org/10.1007/978-3-031-26852-6_24

[37] Joseph, F. J. J., Nonsiri, S., & Monsakul, A. (2021). Keras and TensorFlow: A Hands-On Experience. In Advanced Deep Learning for Engineers and Scientists (pp. 85–111). Springer International Publishing. https://doi.org/10.1007/978-3-030-66519-7_4