e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:05/May-2024 Impact Factor- 7.868 www.irjmets.com

AUTOMATED BORDER SURVEILLANCE SYSTEM WITH REAL-TIME THREAT DETECTION

Vigneshwaran V[*1], Dhanush Adithya S[*2], Mohamed Fazid S[*3]
[*1,2,3]Mohamed Sathak AJ College Of Engineering, India.

DOI : https://www.doi.org/10.56726/IRJMETS57283

ABSTRACT

This paper proposes an automated border surveillance system utilizing machine vision and real-time image processing. This system aims to Reduce Soldier Casualties by eliminating the need for physical border patrols, the system minimizes exposure to enemy attacks and potential casualties, Enhance Border Security by Continuous monitoring and automatic detection of enemy intrusions, weapons, and drones provide a significant tactical advantage, Expand Surveillance Capabilities The system enables surveillance in hostile or challenging environments where physical patrolling is impractical and Protect Wildlife The system can be deployed in forests to monitor human activities, deter illegal hunting, and detect individuals carrying weapons.

The system leverages machine vision algorithms to analyze real-time video feeds from strategically placed cameras. The proposed approach can detect weapons carried by potential intruders, enabling a faster and more effective response. Additionally, the integration of radar technology allows for the detection of incoming drones and missiles, further enhancing border security. This paper details the system architecture, including camera network design, real-time image acquisition and transmission, object detection algorithms, and threat classification mechanisms. The paper concludes by discussing potential future advancements and applications of the proposed system.

Keywords: Border Security, Machine Vision, Real-Time Image Processing, Threat Detection, Weapon Detection, Drone Detection, Wildlife Protection.

I. INTRODUCTION

The rise of modern warfare necessitates a shift in tactics, with technology playing a crucial role. Traditional battlefields are giving way to urban environments, and adversaries are becoming increasingly sophisticated. Defense robots offer a solution, providing troops with an advantage and minimizing casualties.

Evolving Warfare and Technological Solutions:

- Warfare has undergone a dramatic transformation, with technology dictating tactical approaches.
- Contemporary threats involve populated areas as battlegrounds, demanding innovative and intelligent adversaries.
- This poses a challenge to conventional military capabilities, but technological advancements offer solutions for efficient and effective counter-attacks.

Defense Robots: Diverse Capabilities:

- Defense robots, professional service robots deployed in combat, enhance a soldier's capabilities while reducing risk.
- These robots come in various shapes and sizes, catering to specific needs, and can be remotely controlled or autonomous.
- They are equipped with a range of payloads depending on the application, including sensors, detectors, weapons, and programmed software.

Types of Defense Robots and Their Applications:

- **Intelligence, Surveillance, and Reconnaissance (ISR):** Drones (UAVs), unmanned ground vehicles (UGVs), and other robotics are extensively used for ISR, gathering battlefield intelligence.
- **Search and Rescue:** Robots play a vital role in saving lives by reaching victims quickly in dangerous situations.

- **Combat Support:** Robots are deployed in combat support for diverse tasks like mine laying, fire support, and electronic warfare.
- **Mine Clearance:** Robots minimize the risk of casualties by detecting and removing landmines and sea mines.
- **Explosive Ordnance Disposal (EOD):** These robots safely identify and disarm explosive devices in confined spaces.

The Future of Defense Robotics:

- The military currently employs robots for reconnaissance and bomb disposal.
- The future holds promise for entire classes of robots fulfilling battlefield roles, categorized as light, medium, and heavy.
- Robotic tanks, artillery vehicles, and reconnaissance platforms may replace traditional military technologies.

Our Project: A Surveillance and Security System for Defense:

- This project focuses on a surveillance and security system for border security.
- The system utilizes cameras, real-time image capturing, video transmission, and object detection to monitor enemy forces and secure border areas.
- It enables surveillance in hostile and challenging environments where physical patrolling is impractical.

II. METHODOLOGY

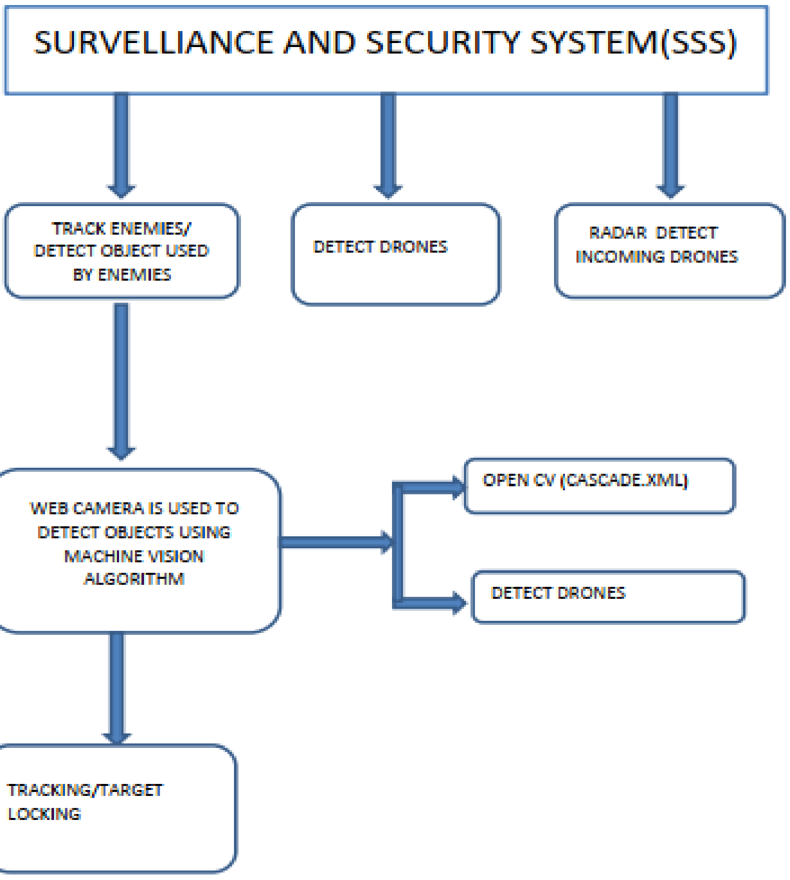
SURVELLIANCE AND SECURITY SYSTEM(SSS)
TRACK ENEMIES/ DETECT OBJECT USED BY ENEMIES
DETECT DRONES
RADAR DETECT INCOMING DRONES
WEB CAMERA IS USED TO DETECT OBJECTS USING MACHINE VISION ALGORITHM
OPEN CV (CASCADE.XML)
DETECT DRONES
TRACKING/TARGET LOCKING

Figure 1: Methodology of Surveillance and Security System (SSS)

III. LEVERAGING PYSERIAL FOR MICROCONTROLLER-BASED OBJECT TRACKING

This paper explores the application of the Python library pyserial in establishing communication between a computer and a microcontroller board for object tracking purposes. pyserial offers a robust and versatile solution for serial port communication across various operating systems, including Windows and IronPython.

Key Advantages of pyserial:

- **Platform Independence:** pyserial provides a consistent interface, simplifying development regardless of the underlying operating system.
- **Comprehensive Configuration:** It allows for direct access and configuration of essential serial port settings through Python properties. This includes parameters like byte size, stop bits, parity, and flow control methods, enabling tailored communication for specific hardware requirements.
- **Flexible Data Handling:** pyserial supports working with or without timeouts during data reception, catering to different communication scenarios. Additionally, it offers a file-like interface for interacting with the serial port, mimicking familiar file reading and writing operations.
- **Pure Python Implementation:** Written entirely in Python, pyserial ensures portability and seamless integration into Python projects.
- **Binary Mode Operation:** By operating in binary mode, pyserial maintains data integrity without alterations like null byte stripping or carriage return translations. This universality makes it suitable for various communication protocols.
- **Compatibility with io Library:** pyserial's compatibility with the standard Python 'io' library enhances its flexibility and facilitates integration with other Python libraries for data processing or visualization.

Project Application:

In this project, pyserial serves as the foundation for establishing communication between a computer and a microcontroller board. This communication channel will be utilized for object tracking. The specific implementation details, including the microcontroller board, object tracking algorithms, and data processing techniques, will be further elaborated upon in subsequent sections of the paper.

IV. FUNCTIONALITY OF ELECTRONIC CIRCUITS

This paper presents the design and implementation of a control system for autonomous platforms utilizing dual servo motors and real-time threat detection. The system leverages an Arduino Uno microcontroller for coordinated servo control and integrates with a computer for image processing and communication.

System Components:

- **Microcontroller:** Arduino Uno serves as the central processing unit, facilitating communication with various components and executing control algorithms.
- **Servo Motors:** Two servo motors are employed:
  - Base Servo Motor: Controls the platform's base rotation (e.g., pan movement).
  - Tilt Servo Motor: Controls the platform's tilt movement (e.g., tilt for vertical aiming).
  - Signal wires connect to digital pins 8 and 9 for individual control.
  - Power is supplied from a 5V, 2A power supply with positive connected to Vin and negative to ground.
- **Image Processing Unit:** An external computer performs image processing tasks. Upon weapon detection, it transmits data to the Arduino Uno.
- **Communication Interface:** pyserial, a Python library, facilitates serial communication between the computer and Arduino Uno via the COM3 port (USB connection). This channel transmits weapon detection data for object tracking.
- **Threat Detection (Optional):** An ultrasonic sensor acts as a radar, detecting incoming rockets and drones. (Details on integration and data processing can be further elaborated upon based on the specific implementation.)

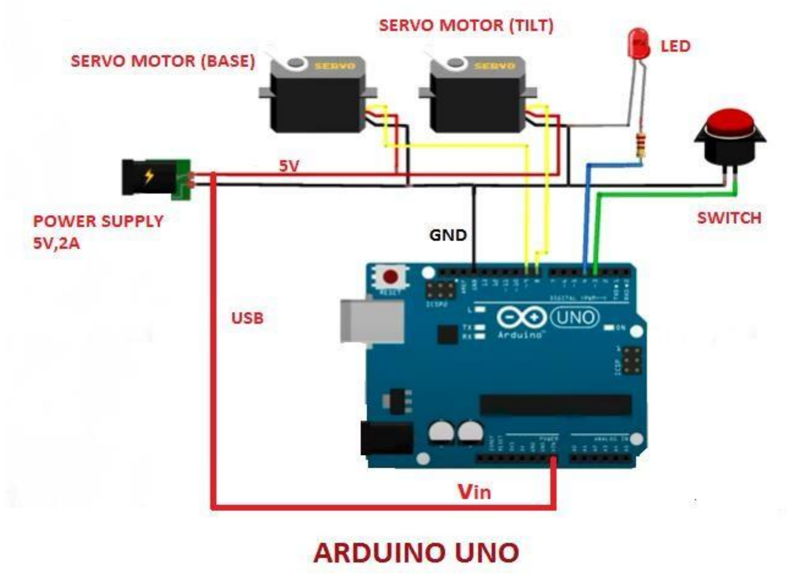

Figure 2: Circuit Diagram to Track the Object

System Functionality:

1. **Image Processing and Weapon Detection:** The external computer performs real-time image processing to identify weapons.
2. **Data Transmission via pyserial:** Upon weapon detection, the computer transmits relevant data (e.g., object location) to the Arduino Uno through the COM3 port using pyserial.
3. **Object Tracking with Servo Motors:** The Arduino Uno receives the data and executes control algorithms to adjust the servo motors. The base and tilt servos work in tandem to track the detected weapon by adjusting the platform's orientation.
4. **Optional Threat Detection:** The ultrasonic sensor (if implemented) continuously monitors for incoming threats like rockets and drones. Detected threats can trigger additional actions or be integrated with the object tracking logic for a comprehensive defense system.

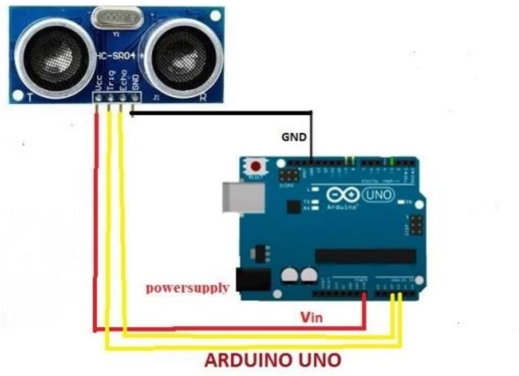

Figure 3: Circuit Diagram for Object Tracking using Ultrasonic Sensor and Arduino Uno

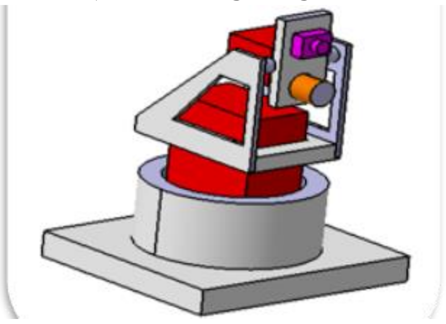

Figure 4: 3D Model of the Image Targeting system

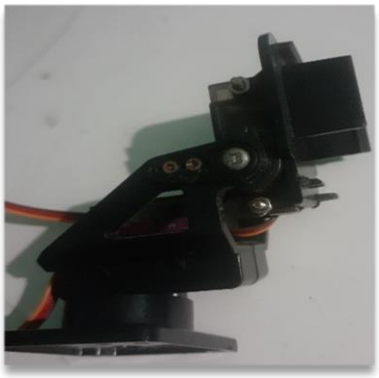

Figure 5: Real Time Working Model of the Image Targeting system

V. FROM PIXELS TO PERCEPTION: BUILDING THE CODE FOR IMAGE TRACKING

Coding for Weapon Detection using HAAR CASCADE (CASCADE.XML):

```
import cv2
import numpy as np
import imutils
import time
import os
import serial
ard= serial.Serial()   # pyserial communication assigning port
ard.port = "COM3"
ard.baudrate = 9600
ard.open()
watch_cascade=cv2.CascadeClassifier('cascade.xml')
cap=cv2.VideoCapture(0,cv2.CAP_DSHOW) # real time capturing
firstFrame = None
gun_exist = False
while True:
    ret, frame = cap.read()
    frame = imutils.resize(frame, width = 500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  # open CV images
  as BRG
    gray = cv2.GaussianBlur(gray,(21,21),0)
    gun = watch_cascade.detectMultiScale(gray,
                    1.3, 5,
                    minSize = (100, 100))

  if len(gun) > 0:
    gun_exist = True


  for (x, y, w, h) in gun:
    print(gun_exist)              # Naming the object
    print(gun)
    frame = cv2.rectangle(frame, # draw a rectangle box and every recognized object
                    (x, y),
                    (x + w, y + h),
                    (255, 100, 50), 2)
    roi_gray = gray[y:y + h, x:x + w]
    roi_color = frame[y:y + h, x:x + w]
    cv2.putText(frame, "gun",
      (x, y),
      cv2.FONT_HERSHEY_SIMPLEX,
      1, (0, 0, 255), 1)
    Xpos = x+(w/2)       #calculates the X coordinate of the center of the face.
   Ypos = y+(h/2)        #calculates the Y coordinate of the center of the face.
    if Xpos >= 380:
      ard.write('L'.encode())  The following code check if the face is on the left,
      time.sleep(0.01)         # right, top or botton,

        elif Xpos <= 260:        #with respect to the center of the frame

          ard.write('R'.encode())  # if any conditions are true, it send a command

          time.sleep(0.01)         # to the arduino throught the serial bus.


    #    else:

    #      ard.write('S'.encode())

#        #time.sleep(0.01)
      if Ypos > 300:
        ard.write('D'.encode())
        time.sleep(0.01)
      elif Ypos < 180:
        ard.write('U'.encode())
        time.sleep(0.01)
#     else:
#         ard.write('S'.encode())
#         time.sleep(0.01)
      break
    if firstFrame is None:
      firstFrame = gray
      continue
    # print(datetime.date(2019))
    # draw the text and timestamp on the frame
    cv2.putText(frame, time.strftime("%m-%d-%Y %T:%M%p"),
                (10, frame.shape[0] - 10),
                cv2.FONT_HERSHEY_SIMPLEX,
                0.35, (0, 0, 255), 1)
        cv2.imshow("Security Feed", frame)
        key = cv2.waitKey(1) & 0xFF

        if key == ord('q'):
            break
```

Coding for Weapon and Drone Detection using YOLOV V3 Program:

```
import cv2
import numpy as np
import imutils
import time
import os
import serial
ard= serial.Serial()
ard.port = "COM3"
ard.baudrate = 9600
ard.open()
net = cv2.dnn.readNet("yolov3 trained.weights", "yolov3_testing.cfg")
#load  yolo
classes = ["Weapon"]
layer_names = net.getlayernames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))
ef value():
  val = input("Enter file name or press enter to start webcam : \n") # we can
give                file name for testing code in vedio
  if val == "":
     val = 0
  return val
cap = cv2.VideoCapture(value(),cv2.CAP_DSHOW) # Vedio capture real time
while True:
  _, img = cap.read()
  height, width, channels = img.shape
  # width = 512
  # height = 512
  blob = cv2.dnn.blobFromImage(img 0.00392, (416, 416), (0, 0, 0), True,
CROP=False)                                        # Detecting objects
  net.setInput(blob)
  outs = net.forward(output_layers)
  class_ids = []                                 # Showing information on the screen
  confidences = []
  boxes = []
  for out in outs:
    for detection in out:
      scores = detection[5:]
      class_id = np.argmax(score)
      confidence = scores[class_id]
      if confidence > 0.5:
      center_x = int(detection[0] * width)      # Object detected
        center_y = int(detection[1] * height)
        w = int(detection[2] * width)
        h = int(detection[3] * height)
        x = int(center_x - w / 2)                 # Rectangle coordinates
        y = int(center_y - h / 2)

        boxes.append([x, y, w, h])
        confidences.append(float(confidence))
        class_ids.append(class_id)

 indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
 print(indexes)
 if indexes == 0: print("weapon detected in frame")
  font = cv2.FONT_HERSHEY_PLAIN
  for i in range(len(boxes)):
    if i in indexes:
      x, y, w, h = boxes[i]
      label = str(classes[class_ids[i]])
      color = colors[class_ids[i]]
      cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
      cv2.putText(img, label, (x, y + 30), font, 3, color, 3)
      Xpos = x+(w/2)   #calculates the X coordinate of the center of the object
      Ypos = y+(h/2)   #calculates the Y coordinate of the center of the object
      if Xpos >= 380:
     ard.write('L'.encode())    #The following code check if the face is on the left,
     time.sleep(0.01)           # right, top or botton,
     elif Xpos <= 260:           #with respect to the center of the frame
      ard.write('R'.encode())    #if any conditions are true, it send a command
      time.sleep(0.01)            #to the arduino throught the serial bus.
      if Ypos > 300:
        ard.write('D'.encode())
        time.sleep(0.01)
      elif Ypos < 180:
            ard.write('U'.encode())
            time.sleep(0.01)
        cv2.imshow("Image", img)
        key = cv2.waitKey(1)
        if key == 27:
            break
```

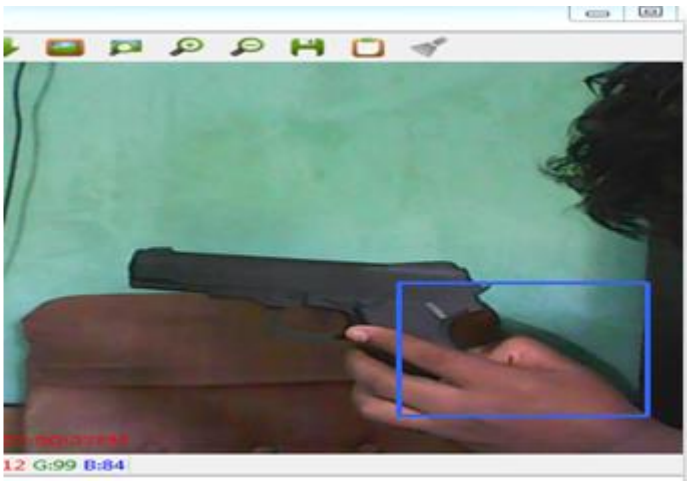

Figure 6: Gun is detected using OPEN CV

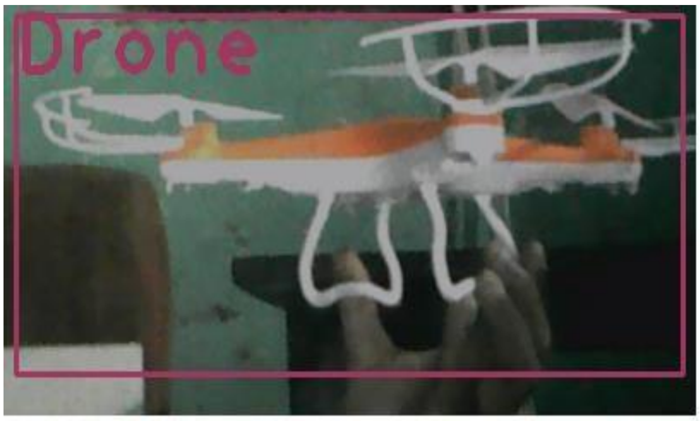

Figure 7: Drone is detected using Yolo V3

Inference:

Open CV (Haar cascade method):

- Detects objects and detects motion
- Less in Accuracy

Yolo v3:

- Detects objects and detects motion
- Accuracy is more
- Requires high processor and graphics processing unit

VI. CONCLUSION

This paper presented the design and implementation of an automated surveillance and security system for border security applications. The system leverages machine vision algorithms and real-time image processing to achieve the following objectives:

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:05/May-2024 Impact Factor- 7.868 www.irjmets.com

- **Reduced Soldier Exposure:** By eliminating the need for physical border patrols, the system minimizes soldier exposure to enemy attacks and potential casualties.
- **Enhanced Border Security:** Continuous monitoring and automatic detection of enemy intrusions, weapons, and drones provide a significant tactical advantage for border defence.
- **Improved Situational Awareness:** The system offers real-time video output, enabling personnel to monitor border activities and make informed decisions.

The proposed approach utilizes object detection algorithms to analyse video feeds from strategically placed cameras. The system can effectively detect weapons carried by potential intruders, allowing for a faster and more effective response. The inclusion of weaponized functionalities, however, is a complex issue with ethical and legal considerations beyond the scope of this paper.

VII. FUTURE RECOMMENDATION

Future research directions include:

- **Advanced Object Recognition:** Implementing algorithms for more sophisticated object recognition, such as distinguishing between weapon types or specific drone models.
- **Integration with Existing Systems:** Exploring seamless integration with existing military communication and command-and-control systems.
- **Enhanced Environmental Resilience:** Developing algorithms that are robust to challenging weather conditions and varying lighting scenarios.

This automated surveillance and security system has the potential to significantly improve border security by reducing soldier risk, enhancing situational awareness, and facilitating a faster response to potential threats. It lays the groundwork for further advancements in autonomous border defence systems while acknowledging the importance of responsible development and deployment within ethical and legal frameworks.

VIII. REFERENCES

[1] Unmanned Aerial Vehicles (UAVs) for Border Surveillance Applications: A Review" by A.A. Agrama et al. in The Journal of Unmanned Vehicle Systems (2013)
[2] A Survey of Border Surveillance Technologies and Algorithms" by M.A. Habib et al. in Sensors (2020)
[3] Deep Learning for Real-Time Video Object Detection: A Survey" by J. Redmon et al. in Pattern Recognition Letters (2016)
[4] Machine Vision for Perimeter Intrusion Detection and Alarm Classification" by M. Nixon et al. in IET Computer Vision (2008)
[5] A Review of Automatic Target Recognition (ATR) System Design Approaches" by X. Li et al. in Pattern Recognition (2011)
[6] Ethical Considerations of Autonomous Weapon Systems" by M. Nissenbaum in International Journal of Robotics and Automation (2017)
[7] Legal Frameworks for Autonomous Weapon Systems: A Comparative Analysis" by A. Duarte et al. in Journal of Law and the Biosciences (2018)
[8] A Survey on Object Detection Methods for Video Surveillance" by I. Marzano et al. in Sensors (2017)
[9] Deep Learning for Military Applications: A Survey" by H. Li et al. in Artificial Neural Networks and Machine Learning - ICANN (2019
[10] Border Security and Technology: Challenges and Opportunities" by M.S. Breen in Security Journal (2007)