ARCIS – ssd document

1. מבוא

פרויקט זה נועד לפתח מערכת ניתוח וידאו המשתמשת באלגוריתמים מתקדמים לזיהוי אובייקטים כדי לזהות ולעקוב אחר ישויות בזמן אמת באמצעות קלט מצלמה. התוכנה מיועדת ליישומים הדורשים מודעות סביבתית, המספקות למשתמשים תובנות ניתנות לפעולה מנתוני וידאו מעובדים.

1. סקירת מערכת

ARCIS היא מערכת ניתוח וידאו שמטרתה לזהות עצמים (נשק ואנשים) בהזנות וידאו בזמן אמת. המערכת מאפשרת זיהוי איומים ומודעות סביבתית באמצעות זיהוי אוטומטי, התראות ושכבות-על. על ידי שילוב רכיבי זיהוי ותצוגה תחת מערכת בקרה מרכזית, היא משיגה פונקציונליות יעילה עבור יישומים טקטיים.

משתמש:

פועל כיוזם ומשקיף של המערכת. הפקודות העיקריות שלו כוללות הפעלה (()start) ועצירה (stop()) של המערכת, בעוד שהתרעות (אודיו וויזואליות) מספקות לו משוב בזמן אמת.

מערכת ARCIS:

מתפקדת כבקרה מרכזית המנהלת אינטראקציות של רכיבים ופקודות מערכת מרכזיות כמו אתחול, עיבוד וידאו וכיבוי מערכת כדי להבטיח פעולה חלקה. מתודות:initialize(), start(), stop(), updateConfiguration()

מצלמה:

מספקת ממשק וידאו ומזרימה נתונים פריים אחר פריים ל-מערכת ARCIS. דורש התאמה עבור רזולוציית פריים, קצב פריימים ותנאי סביבה כמו תאורה או מזג אוויר קשה. מתודות: connect(), disconnect(), startCapture(), stopCapture(), getFrame(), focus(), adjustSetting()

מערכת זיהוי:

אוספת תוצאות מתת-מערכות מיוחדות כמו גלאי נשק (Weapon detector) וגלאי דמויות (PersonDetector). יכולה להיות מורחבת לכלול מודולי זיהוי נוספים במידת הצורך (למשל, עבור כלי רכב או פנים). פלט המערכת יהיה נתונים מובנים המסכמים אובייקטים שזוהו והקואורדינטות המרחביות שלהם. מתודות: initializeModel(), processFrame(int frame), updateModels()

גלאי נשק Weapon detector)):

משתמש באלגוריתמים לזיהוי אובייקטים (למשל, YOLOv5 או RetinaNet) כדי לזהות צורות וגדלים פוטנציאליים של נשק בכל פריים וידאו. מתודות: detect(int frame), classifyWeapon(Detection), updateDatabase()

גלאי דמויותPersonDetector) ):

משתמש במודלים כגון HOG+SVM או ארכיטקטורות מבוססות על deep learning (למשל, Faster R-CNN, YOLOv8) לזיהוי צורות אנושיות. שיטות: detect(int frame), classifyPerson(Detection), updateDatabase()

מערכת תצוגה (DisplaySystem):

מאחדת תוצאות זיהוי להדמיה מונגשת למשתמש ומעבדת זרמי וידאו מתוגברים בשכבות-על כדי להדגיש זיהויים. התצוגה גם מפעילה התראות ברורות המבוססות על סוג זיהוי, תוך הבטחה להבנה ברורה מצד המשתמש. מתודות: initializeDisplay(), updateDisplay(List<Detection>), showAlert(Alert alert)

ניהול שכבות (Overlay manager):

אחראי על יצירה וניהול של שכבות גרפיות לתצוגה ומטפל בעדכונים דינמיים (למשל מעקב אחר תחימות בווידאו) וכולל תכונות אופטימיזציה של זיכרון כמו ניקוי שכבות שאינן בשימוש כדי לשפר ביצועים. מתודות: createOverlay(Detection), UpdateOverlay(Detection), RemoveOverlay(Detection)

אינטראקציות של רכיבים

הפעלה

המשתמש שולח את הפקודה start(), מאתחל את ARCIS\_System. ARCIS\_System מבטיחה שכל הרכיבים, במיוחד ה-CameraModule, מחוברים ומוכנים.

עיבוד

ה-CameraModule מזרים וידאו ל-ARCIS\_System.

פריימים מעובדים על ידי מערכת הזיהוי באמצעות ה-WeaponDetector ו-PersonDetector.

תוצאות הזיהוי (בעיקר חיוביות) נשלחות ל-DisplaySystem, המשתמשת ב-OverlayManager כדי להציג זיהויים ויזואלית ולהפעיל התראות.

שלב הכיבוי

המשתמש מנפיק את הפקודה stop() כדי לסיים פעולות.

מערכת ARCIS\_System מנתקת ומנקה את כל הרכיבים.

1. שיקולי עיצוב

* הנחות

תשתית זמינה:

המערכת מניחה גישה לWIFI מקומי מהיר ותשתית רשת חזקה כדי להתמודד עם הזרמת וניתוח וידאו בזמן אמת, זמינות של ספק כוח התומך בהפעלה 24/7 של מצלמות ויחידות עיבוד. הסביבה שבה מותקנות המצלמות כוללת תאורה מספקת או שניתן לבצע התאמות תאורה נחוצות.

System scalability:

המערכת תוכננה תוך מחשבה על מדרגיות (Scalability), המאפשרת הוספת מצלמות נוספות ומודולי זיהוי לפי הצורך ללא הגדרה מחדש משמעותית.

אלגוריתמי זיהוי:

מניח שאלגוריתמי זיהוי האובייקטים (לדוגמה, YOLO, Faster R-CNN) המשמשים מאומנים מראש ומכוונים עדין למקרה השימוש הספציפי של זיהוי נשק ובני אדם.

ההנחה היא שהדגמים תואמים לחומרת העיבוד ומסוגלים לפעול בזמן אמת.

* אילוצים

מגבלות חומרה:

המערכת עשויה להיות מוגבלת על ידי איכות המצלמות (למשל, רזולוציה, FPS threshold) ויחידות העיבוד (למשל, זמינות GPU) או על ידי קיבולת הזיכרון לאחסון ועיבוד פריימים נתונים לזיהוי.

עיבוד בזמן אמת:

הדרישה לעיבוד בזמן אמת מטילה מגבלות על מורכבות אלגוריתמי הזיהוי וכמות הנתונים שניתן לעבד בו זמנית ויש למזער בעיות אחזור (Latency) בהזרמת וידאו ובזיהוי כדי להבטיח התראות ומשוב בזמן.

* תקנים

תקני קידוד:

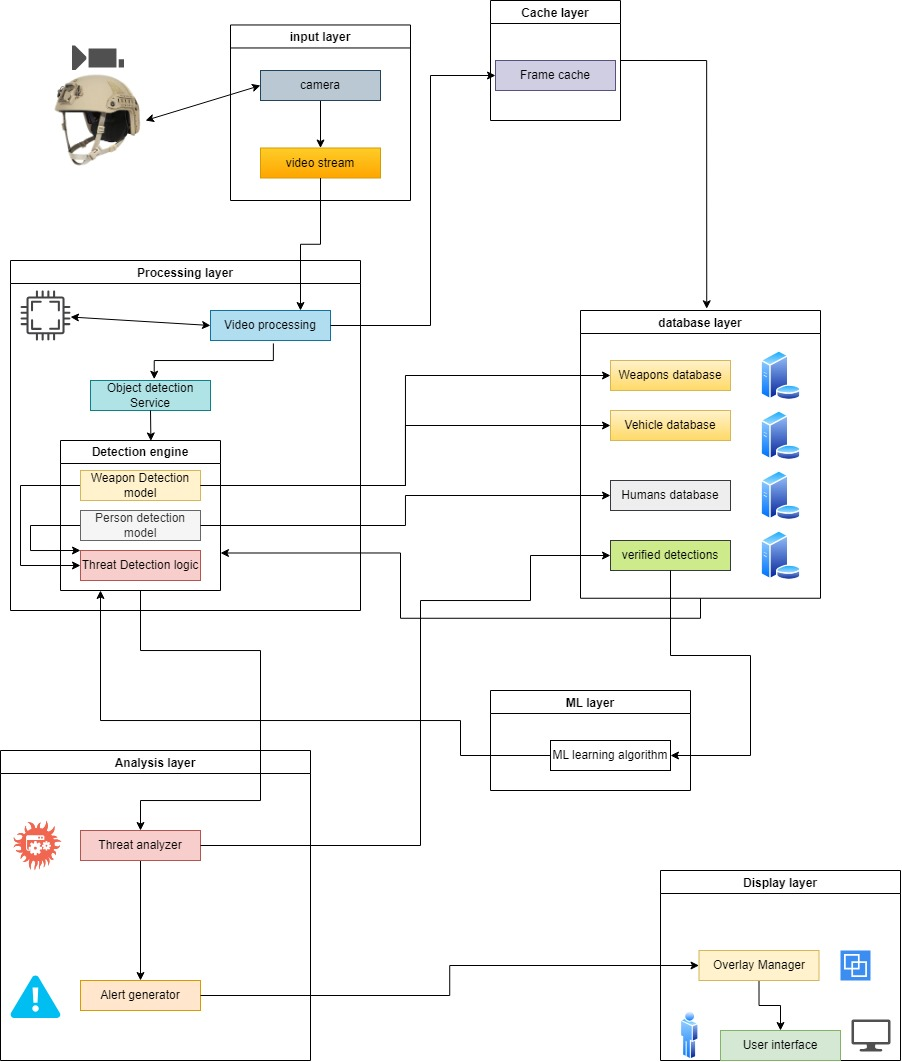
נכתוב תוך מעקב אחר השיטות המומלצות בתעשייה לקידוד, כולל עיצוב מודולרי, שימוש חוזר בקוד ותחזוקה ועמידה בתקני הקידוד של שפות התכנות בהן נעשה שימוש (למשל, PEP 8 עבור Python).

פרוטוקולים:

שימוש בפרוטוקולי תקשורת סטנדרטיים להעברת נתונים (למשל, HTTP/HTTPS, WebSockets) כדי להבטיח יכולת פעולה הדדית וחילופי נתונים מאובטחים. עמידה בפרוטוקולי אבטחת רשת חשובה כדי להגן מפני גישה בלתי מורשית והפרות נתונים.

מעקב אחר עקרונות עיצוב UI/UX עבור מערכת התצוגה כדי להבטיח ממשק אינטואיטיבי וידידותי למשתמש.

1. ארכיטקטורת מערכת



1. עיצוב רכיבים (component design):

1) CameraModule

מטרה: טיפול ברכישת וידאו על ידי יצירת חיבורים והזרמת וידאו.

קלט: אין (נדרשת תצורה ראשונית).

פלט: זרם פריימים של וידאו.

תלות: תלוי בהתקני מצלמת החומרה.

ממשקים:

connect(): יוצר חיבור עם המצלמה.

disconnect(): מפסיק את החיבור עם המצלמה.

startCapture(): מתחיל ללכוד מסגרות וידאו.

stopCapture(): מפסיק ללכוד מסגרות וידאו.

getFrame(): מאחזר את מסגרת הווידאו הנוכחית.

adjustSetting(): מכוונן את הגדרות המצלמה כמו רזולוציה וקצב פריימים.

2( ARCIS\_System

מטרה: בקר מרכזי ניהול אתחול, עיבוד וידאו וכיבוי מערכת.

קלט: פקודות משתמש (התחלה, עצירה).

פלט: פריימים מעובדים (שעברו analysis).

תלות במצב מערכת: מקיים אינטראקציה עם CameraModule, DetectionSystem ו-DisplaySystem.

ממשקים:

initialize(): מאתחל את המערכת ומרכיביה.

start(): מפעיל את המערכת.

stop(): עוצר את המערכת.

updateConfiguration(): עדכון הגדרות מערכת ותצורות.

3) DetectionSystem

מטרה: טיפול באיתור על ידי עיבוד פריימים באמצעות מודלים מיוחדים לזיהוי. קלט: מסגרות וידאו מ-ARCIS\_System

פלט: תוצאות זיהוי (נשק, אנשים).

תלות: משתמש ב-WeaponDetector ו-PersonDetector.

ממשקים:

initializeModel(): מאתחל מודלים של זיהוי.

processFrame(frame): מעבד פריים וידאו לזיהוי.

updateModels(): מעדכן מודלים של זיהוי עם נתונים או פרמטרים חדשים.

4) WeaponDetector

מטרה: זיהוי וסיווג כלי נשק במסגרות וידאו.

קלט: וידאו.

פלט: כלי נשק שזוהו עם תחימה בUI.

תלות: משתמש במודלים מאומנים מראש ובdatabase של נשק.

ממשקים:

detect(frame): מזהה כלי נשק במסגרת וידאו.

classifyWeapon(Detection): מסווג כלי נשק שזוהו.

updateDatabase(): מעדכן את מסד הנתונים של הנשק עם ערכים חדשים.

5) PersonDetector

מטרה: זיהוי וסיווג אנשים בוידאו.

קלט: וידאו.

פלט: אנשים שזוהו עם תחימה בUI.

תלות: משתמש במודלים מאומנים מראש ובdatabase של דמויות בתנוחות שונות.

ממשקים:

detect(frame): מזהה אנשים במסגרת וידאו.

classifyPerson(Detection): מסווג אנשים שזוהו.

updateDatabase(): מעדכן את מסד הנתונים של האדם בערכים חדשים.

6) DisplaySystem

מטרה: איחוד תוצאות זיהוי וניהול הדמיה מול המשתמש והתראות.

קלט: תוצאות זיהוי מ-DetectionSystem.

פלט: התראות ויזואליות ואודיו.

תלויות: עובד עם OverlayManager ו-AlertSystem.

ממשקים:

initializeDisplay(): מאתחל את מערכת התצוגה.

updateDisplay(List<Detection>): מעדכן את התצוגה עם תוצאות זיהוי חדשות.

showAlert(Alert Alert): מציג התראה למשתמש.

7) OverlayManager

מטרה: ניהול שכבות גרפיות עבור אובייקטים שזוהו.

קלט: תוצאות זיהוי.

פלט: עדכון תוצאות ל-UI.

תלות: משולבת עם DisplaySystem.

ממשקים:

createOverlay(Detection): יוצר שכבת-על עבור אובייקט שזוהה.

updateOverlay(Detection): מעדכן שכבת-על קיימת.

removeOverlay(Detection): מסיר שכבת-על כאשר האובייקט אינו מזוהה עוד.

8) AlertSystem

מטרה: ניהול מידע התראות והפקת התראות חזותיות/אודיו.

קלט: תוצאות זיהוי.

פלט: התראות ויזואליות ואודיו.

תלות: משולבת ב-DisplaySystem.

ממשקים:

generAlert(Detection): יוצר התראה עבור אובייקט שזוהה.

updateAlert(Alert Alert): מעדכן התראה קיימת.

removeAlert(Alert Alert): מסיר התראה כאשר האיום אינו מזוהה עוד.

זרימת נתונים

1. שלב ההפעלה:

המשתמש שולח את הפקודה start() אל ARCIS\_System.

ARCIS\_System מאתחל את כל הרכיבים, במיוחד CameraModule.

1. שלב העיבוד:

CameraModule מתחיל להזרים מסגרות וידאו ל-ARCIS\_System.

ARCIS\_System שולח מסגרות ל-DetectionSystem לניתוח.

DetectionSystem מעבד מסגרות באמצעות WeaponDetector ו-PersonDetector.

תוצאות הזיהוי נשלחות ל-DisplaySystem.

DisplaySystem משתמשת ב-OverlayManager ליצירת שכבות חזותיות וב-AlertSystem ליצירת התראות.

1. שלב הכיבוי:

המשתמש שולח את הפקודה stop() אל ARCIS\_System.

ARCIS\_System מפסיקה פעולות.

1. Detailed class/function design
2. CameraModule Class

class CameraModule:

def \_\_init\_\_(self, address):

self.address = address

self.isOn = False

self.status = 'disconnected'

self.settings = {}

def connect(self):

“Establishes a connection with the camera.”

# Implementation code here

self.status = 'connected'

def disconnect(self):

“Terminates the connection with the camera.”

# Implementation code here

self.status = 'disconnected'

def startCapture(self):

“Starts capturing video frames.”

if self.status == 'connected':

self.isOn = True

# Additional implementation code here

def stopCapture(self):

“Stops capturing video frames.”

if self.isOn:

self.isOn = False

# Additional implementation code here

def getFrame(self):

“Retrieves the current video frame.”

if self.isOn:

# Frame retrieval code here

pass

def adjustSetting(self, setting, value):

“Adjusts camera settings like resolution and frame rate.”

self.settings[setting] = value

# Apply setting code here

1. ARCIS\_System Class

class ARCIS\_System:

def \_\_init\_\_(self, videoSource, detectionSystem, display):

self.videoSource = videoSource

self.detectionSystem = detectionSystem

self.display = display

def initialize(self):

“Initializes the system and its components.”

self.videoSource.connect()

self.detectionSystem.initializeModel()

self.display.initializeDisplay()

def start(self):

“Starts the system.”

self.videoSource.startCapture()

# Additional start sequence code here

def stop(self):

“Stops the system.”

self.videoSource.stopCapture()

self.videoSource.disconnect()

# Additional stop sequence code here

def updateConfiguration(self, config):

“Updates system settings and configurations.”

# Update config code here

Pass

1. DetectionSystem Class

class DetectionSystem:

def \_\_init\_\_(self, models):

self.models = models

def initializeModel(self):

“Initializes detection models.”

for model in self.models:

model.initialize()

def processFrame(self, frame):

“Processes a video frame for detection.”

detections = []

for model in self.models:

detections.extend(model.detect(frame))

return detections

def updateModels(self, newModels):

“Updates detection models with new data or parameters.”

self.models = newModels

1. WeaponDetector Class

class WeaponDetector(DetectionModel):

def \_\_init\_\_(self, weaponDB):

self.weaponDB = weaponDB

def detect(self, frame):

“Detects weapons in a video frame.”

# Detection code here

return []

def classifyWeapon(self, detection):

“Classifies detected weapons.”

# Classification code here

pass

def updateDatabase(self, newWeapons):

“Updates the weapon database with new entries.”

self.weaponDB.extend(newWeapons)

1. WeaponDetector Class

class WeaponDetector(DetectionModel):

def \_\_init\_\_(self, weaponDB):

self.weaponDB = weaponDB

def detect(self, frame):

“Detects weapons in a video frame.”

# Detection code here

return []

def classifyWeapon(self, detection):

“Classifies detected weapons.”

# Classification code here

pass

def updateDatabase(self, newWeapons):

“Updates the weapon database with new entries.”

self.weaponDB.extend(newWeapons)

1. PersonDetector Class

class PersonDetector(DetectionModel):

def \_\_init\_\_(self, personDB):

self.personDB = personDB

def detect(self, frame):

“Detects people in a video frame.”

# Detection code here

return []

def classifyPerson(self, detection):

“Classifies detected people.”

# Classification code here

pass

def updateDatabase(self, newPersons):

“Updates the person database with new entries.”

self.personDB.extend(newPersons)

1. User Interface Design
2. Login page

+-------------------------------------+

| Login Page |

+-------------------------------------+

| Username: [\_\_\_\_\_\_\_\_\_\_\_\_\_] |

| Password: [\_\_\_\_\_\_\_\_\_\_\_\_\_] |

| |

| [Login Button] |

+-------------------------------------+

1. Dashboard

+-------------------------------------+

| Navigation Bar: |

| Home |

| Cameras |

Small collapsable menu

| Settings |

| Logout |

+-------------------------------------+

| Live Feed |

The large part of the UI

| Notifications |

+-------------------------------------+

1. Camera management page

+-------------------------------------+

| Camera Management |

+-------------------------------------+

| [Add Camera Button] |

| Camera List: |

| - Camera 1 [Edit] [Delete] |

| - Camera 2 [Edit] [Delete] |

+-------------------------------------+

1. Settings page

+----------------------------------------------+

| Settings |

+----------------------------------------------+

| Profile Settings |

| - Current Password: [\_\_\_\_\_\_\_\_\_\_\_\_\_] |

| - New Password: [\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_] |

| - Confirm Password: [\_\_\_\_\_\_\_\_\_\_\_\_\_] |

| System Settings |

| - [Update Configurations...] |

| |

| [Save Button] [Reset Button] |

+----------------------------------------------+