## **C++ Object Tracking & Surveillance System Project (OpenCV & Qt GUI)**

### **Objective:**

Develop a C++ Object Tracking and Surveillance System based on the requirements outlined by KPIT Team.

### **1. Project Statement Summary:**

The core goal is to build a real-time system using OpenCV C++ to detect objects (people, vehicles), track them across frames (with persistent unique IDs), classify them, provide tracking information (location, velocity, trajectory), generate alerts (e.g., restricted area, speed), record video, and provide a user-friendly interface. A minimum performance of 30 FPS is required. Key deliverables include source code, CMakeLists.txt, UML diagram, documentation, and test cases.

### **2. Development Environment:**

* **OS:** Windows
* **Environment:** MSYS2 with MinGW-w64 (mingw64 terminal)
* **Language:** C++17
* **Build System:** CMake (using MinGW Makefiles generator)
* **Core Libraries:**
  + OpenCV 4.11.0 (Modules: core, highgui, videoio, imgproc, objdetect, tracking, dnn)
  + Qt5 (Modules: Core, Gui, Widgets - used for the GUI)
  + Protobuf (as a dependency for OpenCV DNN)
* **Required MSYS2 pacman Packages:**
  + mingw-w64-x86\_64-toolchain
  + mingw-w64-x86\_64-cmake
  + mingw-w64-x86\_64-opencv
  + mingw-w64-x86\_64-protobuf
  + mingw-w64-x86\_64-qt5-base
  + mingw-w64-x86\_64-qt5-svg
  + mingw-w64-x86\_64-qt5-tools

### **3. Project Folder Structure:**

<Project\_Root\_Folder>/

├── build/

├── data/

│ ├── yolov4-tiny.weights

│ ├── yolov4-tiny.cfg

│ └── coco.names

├── install/

│ ├── bin/

│ │ ├── ObjectTrackingApp.exe

│ │ └── platforms/

│ └── data/

├── src/

│ ├── main.cpp

│ ├── MainWindow.h

│ ├── MainWindow.cpp

│ ├── VideoProcessor.h

│ └── VideoProcessor.cpp

├── include/

├── resources/

├── CMakeLists.txt

└── README.md

### **4. Final Code Summary (Last working state)**

#### **CMakeLists.txt**

* Configures project for C++17.
* Finds OpenCV and Qt5 packages.
* Enables AUTOMOC, AUTOUIC, and AUTORCC.
* Links OpenCV + Qt5 libraries.
* Sets CMAKE\_BUILD\_TYPE to Release.
* Defines installation rules for executable, data, and Qt platform plugins.

#### **src/main.cpp**

* Launches the Qt GUI with QApplication.
* Creates and shows MainWindow.
* Starts event loop via a.exec().

#### **src/MainWindow.h**

* Declares the GUI QMainWindow.
* UI elements: QPushButton, QLabel, QCheckBox, Layouts.
* Slots for button events and worker communication.
* Declares VideoProcessor object and its QThread.

#### **src/MainWindow.cpp**

* Sets up the UI layout (setupUi()).
* Connects GUI buttons and checkboxes to worker slots.
* Moves VideoProcessor to a background thread.
* Ensures clean thread shutdown on exit.
* Uses QFileDialog for file selection.
* Uses Qt::QueuedConnection to avoid thread-safety issues.

#### **src/VideoProcessor.h**

* Defines the VideoProcessor class (QObject-based).
* Contains OpenCV elements (VideoCapture, dnn::Net, VideoWriter).
* Stores tracking state and UI toggles (\_drawRestrictedZone, \_drawTrajectory, \_checkSpeedAlert).
* Defines LegacyTrackerWrapper to use cv::legacy::TrackerMOSSE.

#### **src/VideoProcessor.cpp**

* **Model Loading:** Loads YOLOv4-tiny from data/.
* **Tracking:** MOSSE tracker wrapped via LegacyTrackerWrapper.
* **Detection:** Periodic detection via YOLOv4-tiny.
* **ID Assignment:** Uses IoU-based matching (not deep Re-ID).

**Velocity Calculation:** Uses pixel displacement over time (not real-world units).  
  
 double velocity = cv::norm(currCenter - prevCenter) / frameTimeSec; // pixels/sec

* **Trajectory Drawing:** Toggleable in UI.
* **Restricted Zone:** Defined via polygon, checked using cv::pointPolygonTest().
* **UI Integration:** Processes frames via QTimer, emits QPixmap to main UI.
* **Video Recording:** Uses cv::VideoWriter, filename includes timestamp.
* **Signals/Slots:** Emit frameProcessed and statusUpdated signals.

### **5. Build, Install, and Run Process (Final):**

# Run from MSYS2 MinGW 64-bit terminal in <Project Root>

rm -rf build install # Clean build

mkdir build && cd build

cmake .. -G "MinGW Makefiles" -DCMAKE\_INSTALL\_PREFIX=../install

mingw32-make # Build in Release mode

mingw32-make install # Install to install/bin

# Copy DLLs to install/bin (script or manual)

cd ../install/bin

cp /mingw64/bin/libopencv\_core-411.dll .

cp /mingw64/bin/libopencv\_highgui-411.dll .

cp /mingw64/bin/libopencv\_imgproc-411.dll .

cp /mingw64/bin/libopencv\_videoio-411.dll .

cp /mingw64/bin/libopencv\_objdetect-411.dll .

cp /mingw64/bin/libopencv\_tracking-411.dll .

cp /mingw64/bin/libopencv\_dnn-411.dll .

cp /mingw64/bin/libprotobuf-lite.dll .

cp /mingw64/bin/Qt5Core.dll .

cp /mingw64/bin/Qt5Gui.dll .

cp /mingw64/bin/Qt5Widgets.dll .

cp /mingw64/bin/Qt5Svg.dll .

cp /mingw64/bin/libgcc\_s\_seh-1.dll .

cp /mingw64/bin/libstdc++-6.dll .

cp /mingw64/bin/libwinpthread-1.dll .

cp libprotobuf-lite.dll libprotobuf-lite320.dll # DNN fix

# Run the app

./ObjectTrackingApp.exe

### **6. Key Troubleshooting History Summary**

* **DLL Dependency Fix:** Manual copy of required .dll files from /mingw64/bin to install/bin. A script (copy\_dlls.bat) was created to speed this up.
* **Tracking Performance Bottleneck:** Initial use of cv::TrackerKCF caused FPS drop. Replaced with **cv::legacy::TrackerMOSSE**, wrapped in a LegacyTrackerWrapper to handle API differences.
* **Camera Access Issues:** Fallback logic added for alternative camera indices if 0 fails.
* **OpenCV/Qt Linking Errors:** Fixed linker issues with correct order of libraries and use of AUTOMOC, AUTOUIC in CMakeLists.txt.
* **Thread-Safe GUI Updates:** Ensured GUI updates are done via Qt::QueuedConnection from worker thread.
* **Video Saving Path Issue:** Ensured correct relative pathing for output videos and data/ folder references from install/bin.
* **Velocity Units:** Computed in **pixels/sec** due to lack of scene calibration.

### **7. Security Features (Recommended for Future)**

* Add user authentication to restrict access to GUI features.
* Encrypt video recordings using a standard encryption algorithm.
* Validate and sanitize all file paths used for saving outputs.
* Implement hash-based integrity checks on saved videos or alerts.
* Blur or mask sensitive objects (e.g., human faces) before saving video.
* Maintain a log of all alerts and actions for auditing.
* Restrict access to known camera devices using device ID.
* Digitally sign the final executable to verify source authenticity.

### **8. How to Run This Project from the Submitted ZIP File**

* Extract the ZIP file to a clean directory without special characters or spaces in the path.
* Install MSYS2 and the required packages using the following commands:  
    
   pacman -Syu

pacman -S mingw-w64-x86\_64-toolchain mingw-w64-x86\_64-cmake \

mingw-w64-x86\_64-opencv mingw-w64-x86\_64-protobuf \

mingw-w64-x86\_64-qt5-base mingw-w64-x86\_64-qt5-svg \

mingw-w64-x86\_64-qt5-tools

* Open the MSYS2 MinGW 64-bit terminal and build the project:  
    
   cd <Extracted\_Folder>

mkdir build && cd build

cmake .. -G "MinGW Makefiles" -DCMAKE\_INSTALL\_PREFIX=../install

mingw32-make

mingw32-make install

* Copy the required DLL files to install/bin either manually or using the included copy\_dlls.bat script.
* Navigate to install/bin and run ObjectTrackingApp.exe.