```python
# Import necessary libraries
import cv2
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense
import matplotlib.pyplot as plt

# Data collection
def collect_data(video_path):
    """
    Collects video frames from the given video path.

    Args:
    video_path (str): Path to the video file.

    Returns:
    frames (list): List of video frames.
    """
    # Code to collect video frames
    pass

# Data preprocessing
def preprocess_data(frames):
    """
    Preprocesses the collected frames for object detection and tracking.

    Args:
    frames (list): List of video frames.

    Returns:
    preprocessed_frames (list): Preprocessed frames ready for model input.
    """
    # Code for data preprocessing
    pass

# Model selection
def select_model(input_shape):
    """
    Selects a deep learning model for object detection and tracking.

    Args:
    input_shape (tuple): Shape of the input frames.

    Returns:
    model (tf.keras.Model): Selected deep learning model.
    """
    # Code for model selection
    input_layer = Input(shape=input_shape)
    # Define the model architecture
    # ...
    model = Model(inputs=input_layer, outputs=output_layer)
    return model

# Model training
def train_model(model, X_train, y_train):
    """
    Trains the selected model using the provided training data.

    Args:
    model (tf.keras.Model): Selected deep learning model.
    X_train (np.array): Input training data.
    y_train (np.array): Target training data.
    Returns:
    trained_model (tf.keras.Model): Trained deep learning model.
    """
    # Code for model training
    trained_model = model.fit(X_train, y_train, epochs=10, batch_size=32)
    return trained_model

# Real-time detection
def detect_objects(model, video_stream):
    """
    Performs real-time object detection using the trained model on the video
stream.

    Args:
    model (tf.keras.Model): Trained deep learning model.
    video_stream (cv2.VideoCapture): Video stream for real-time detection.

    Returns:
    None
    """
    # Code for real-time object detection
    pass

# Object tracking
def track_objects(video_stream, detected_objects):
    """
    Tracks the detected objects in the video stream.

    Args:
    video_stream (cv2.VideoCapture): Video stream for object tracking.
    detected_objects (list): List of detected objects.

    Returns:
    None
    """
    # Code for object tracking
    pass

# Visualization
def visualize_results(video_stream, tracked_objects):
    """
    Visualizes the tracked objects in the video stream.

    Args:
    video_stream (cv2.VideoCapture): Video stream for visualization.
    tracked_objects (list): List of tracked objects.

    Returns:
    None
    """
    # Code for visualization
    pass

# Main function
if __name__ == "__main__":
    # Define video path
    video_path = "path_to_video_file.mp4"

    # Collect data
    frames = collect_data(video_path)

    # Preprocess data
    preprocessed_frames = preprocess_data(frames)
```

```
# Select model
input_shape = preprocessed_frames[0].shape
model = select_model(input_shape)

# Train model
X_train = np.array(preprocessed_frames)
y_train = np.array([0, 1, 0, 1, 0])  # Example target data
trained_model = train_model(model, X_train, y_train)

# Real-time detection
video_stream = cv2.VideoCapture(video_path)
detect_objects(trained_model, video_stream)

# Object tracking
detected_objects = []  # Example detected objects
track_objects(video_stream, detected_objects)

# Visualization
tracked_objects = []  # Example tracked objects
visualize_results(video_stream, tracked_objects)
```