In [2]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/multiple-cameras-fall-dataset/technicalReport.pdf
/kaggle/input/multiple-cameras-fall-dataset/data_tuple3.csv
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute23/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute23/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute23/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute23/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute23/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute23/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute23/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute23/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute15/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute15/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute15/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute15/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute15/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute15/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute15/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute15/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute16/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute16/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute16/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute16/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute16/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute16/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute16/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute16/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute08/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute08/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute08/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute08/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute08/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute08/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute08/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute08/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute21/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute21/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute21/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute21/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute21/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute21/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute21/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute21/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute12/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute12/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute12/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute12/cam6.avi
```

```
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute12/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute12/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute12/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute12/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute22/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute22/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute22/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute22/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute22/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute22/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute22/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute22/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute03/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute03/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute03/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute03/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute03/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute03/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute03/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute03/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute14/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute14/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute14/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute14/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute14/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute14/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute14/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute14/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute05/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute05/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute05/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute05/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute05/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute05/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute05/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute05/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute06/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute06/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute06/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute06/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute06/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute06/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute06/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute06/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute09/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute09/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute09/cam4.avi
```

```
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute09/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute09/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute09/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute09/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute09/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute20/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute20/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute20/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute20/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute20/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute20/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute20/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute20/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute17/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute17/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute17/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute17/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute17/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute17/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute17/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute17/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute11/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute11/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute11/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute11/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute11/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute11/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute11/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute11/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute02/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute02/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute02/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute02/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute02/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute02/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute02/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute02/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute24/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute24/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute24/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute24/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute24/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute24/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute24/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute24/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute07/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute07/cam3.avi
```

```
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute07/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute07/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute07/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute07/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute07/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute07/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute19/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute19/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute19/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute19/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute19/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute19/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute19/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute19/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute04/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute04/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute04/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute04/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute04/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute04/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute04/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute04/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute13/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute13/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute13/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute13/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute13/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute13/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute13/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute13/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute01/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute01/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute01/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute01/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute01/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute01/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute01/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute01/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute10/cam7.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute10/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute10/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute10/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute10/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute10/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute10/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute10/cam2.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute18/cam7.avi
```

```
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute18/cam3.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute18/cam4.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute18/cam6.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute18/cam8.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute18/cam5.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute18/cam1.avi
/kaggle/input/multiple-cameras-fall-dataset/dataset/dataset/chute18/cam2.avi
/kaggle/input/copy-for-separate-cells/__results__.html
/kaggle/input/copy-for-separate-cells/__notebook__.ipynb
/kaggle/input/copy-for-separate-cells/__output__.json
/kaggle/input/copy-for-separate-cells/custom.css
/kaggle/input/copy-for-separate-cells/models/confidence_model.pkl
```

In [3]:

```
!pip install scikit-image
```

```
Requirement already satisfied: scikit-image in /usr/local/lib/python3.11/dist-packages (0.
25.1)
Requirement already satisfied: numpy>=1.24 in /usr/local/lib/python3.11/dist-packages (fro
m scikit-image) (1.26.4)
Requirement already satisfied: scipy>=1.11.2 in /usr/local/lib/python3.11/dist-packages (f
rom scikit-image) (1.15.2)
Requirement already satisfied: networkx>=3.0 in /usr/local/lib/python3.11/dist-packages (f
rom scikit-image) (3.4.2)
Requirement already satisfied: pillow>=10.1 in /usr/local/lib/python3.11/dist-packages (fr
om scikit-image) (11.1.0)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in /usr/local/lib/python3.11/dist-pa
ckages (from scikit-image) (2.37.0)
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.11/dist-packa
ges (from scikit-image) (2025.1.10)
Requirement already satisfied: packaging>=21 in /usr/local/lib/python3.11/dist-packages (f
rom scikit-image) (24.2)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.11/dist-packages
(from scikit-image) (0.4)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.11/dist-packages (from nu
mpy>=1.24->scikit-image) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python3.11/dist-packages (from
numpy>=1.24->scikit-image) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.11/dist-packages (from
numpy>=1.24->scikit-image) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.11/dist-packages (from numpy>
=1.24->scikit-image) (2025.1.0)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.11/dist-packages (from num
py>=1.24->scikit-image) (2022.1.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python3.11/dist-packages (fro
m numpy>=1.24->scikit-image) (2.4.1)
Requirement already satisfied: intel-openmp<2026,>=2024 in /usr/local/lib/python3.11/dist-
packages (from mkl->numpy>=1.24->scikit-image) (2024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.11/dist-packages (fro
m mkl->numpy>=1.24->scikit-image) (2022.1.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.11/dist-packages (fro
m tbb==2022.*->mkl->numpy>=1.24->scikit-image) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in /usr/local/lib/python3.11/dist-packag
es (from mkl_umath->numpy>=1.24->scikit-image) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in /usr/local/lib/python3.11/d
ist-packages (from intel-openmp<2026,>=2024->mkl->numpy>=1.24->scikit-image) (2024.2.0)
```

In [4]:

```
import os
import cv2
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from skimage.feature import local_binary_pattern
from skimage.measure import moments, moments_central
from scipy.stats import skew, kurtosis
import matplotlib.pyplot as plt
from tqdm import tqdm
import pickle
import warnings
warnings.filterwarnings('ignore')
```

```
2025-04-20 00:02:14.609767: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477]
Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one
has already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1745107334.813293      31 cuda_dnn.cc:8310] Unable to register cuDNN factory:
Attempting to register factory for plugin cuDNN when one has already been registered
E0000 00:00:1745107334.866707      31 cuda_blas.cc:1418] Unable to register cuBLAS factor
y: Attempting to register factory for plugin cuBLAS when one has already been registered
```

In [5]:

```
WINDOW_SIZE = 15
DATA_PATH = "/kaggle/input/multiple-cameras-fall-dataset/"  # Main dataset folder
DATASET_FOLDER = os.path.join(DATA_PATH, "dataset", "dataset")  # Path to actual dataset video
s with chute folders
CSV_PATH = os.path.join(DATA_PATH, "data_tuple3.csv")  # Path to the CSV file
MODEL_PATH = "./models/"  # Path to save models (in the Kaggle working directory)
RESULTS_PATH = "./results/"  # Path to save results (in the Kaggle working directory)
```

In [6]:

```
os.makedirs(MODEL_PATH, exist_ok=True)
os.makedirs(RESULTS_PATH, exist_ok=True)
```

In [7]:

```python
def read_dataset_info(csv_path):
    """Read the dataset information from the CSV file"""
    df = pd.read_csv(csv_path)
    # Add debug statements right after reading
    print("CSV data sample:")
    print(df.head())
    print(f"Fall events in CSV: {len(df[df['label'] == 1])}")
    print(f"Unique chute values in CSV: {df['chute'].unique()}")
    return df
```

In [8]:

```python
def extract_bounding_box(frame, background_subtractor):
    """Extract bounding box of the human silhouette"""
    fgmask = background_subtractor.apply(frame)

    # Apply morphological operations to remove noise
    kernel = np.ones((5, 5), np.uint8)
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_OPEN, kernel)
    fgmask = cv2.morphologyEx(fgmask, cv2.MORPH_CLOSE, kernel)

    # Find contours
    contours, _ = cv2.findContours(fgmask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    if len(contours) == 0:
        return None, fgmask

    # Find the largest contour (assumed to be the person)
    largest_contour = max(contours, key=cv2.contourArea)

    # Get bounding box
    x, y, w, h = cv2.boundingRect(largest_contour)

    return (x, y, w, h), fgmask

def calculate_silhouette_ratio(bbox):
    """Calculate the ratio of the silhouette's bounding box (width/height)"""
    if bbox is None:
        return None
    x, y, w, h = bbox
    return w / h if h > 0 else 0

def calculate_orientation(fgmask, bbox):
    """Calculate the orientation of the silhouette"""
    if bbox is None or fgmask is None:
        return None

    x, y, w, h = bbox
    # Extract the silhouette region
    silhouette = fgmask[y:y+h, x:x+w]

    # Calculate moments
    m = moments_central(silhouette)

    # Calculate orientation
    # Check if denominator would be zero
    if abs(m[2, 0] - m[0, 2]) < 1e-10:
        return 0
```

```
    # Calculate the orientation using central moments
    theta = 0.5 * np.arctan2(2 * m[1, 1], (m[2, 0] - m[0, 2]))
    return theta

def calculate_centroid_height(bbox, frame_height):
    """Calculate the height of the centroid from the bottom of the frame"""
    if bbox is None:
        return None

    x, y, w, h = bbox
    centroid_y = y + h/2
    return frame_height - centroid_y

def calculate_optical_projection(fgmask):
    """Calculate the projection of the silhouette on the optical x-axis"""
    if fgmask is None:
        return None

    # Sum pixels along columns to get x-axis projection
    x_projection = np.sum(fgmask, axis=0)
    return x_projection

def calculate_brightness(frame):
    """Calculate the average brightness of the frame"""
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    return np.mean(gray)

def calculate_blind_quality_score(frame):
    """Calculate blind quality score using BRISQUE-like features"""
    # This is a simplified version as full BRISQUE requires a pre-trained model
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Calculate LBP
    lbp = local_binary_pattern(gray, 8, 1, method='uniform')

    # Calculate histogram of LBP
    hist, _ = np.histogram(lbp, bins=59, range=(0, 59))
    hist = hist.astype("float")
    hist /= (hist.sum() + 1e-7)

    # Calculate simple statistics
    mean_val = np.mean(gray)
    std_val = np.std(gray)
    skewness = skew(gray.flatten())
    kurt = kurtosis(gray.flatten())
```

```
    # Simple quality score (lower is better)
    score = std_val * (skewness ** 2) * (kurt ** 2)

    return score

def calculate_silhouette_size(bbox):
    """Calculate the size of the silhouette"""
    if bbox is None:
        return 0

    x, y, w, h = bbox
    return w * h
```

In [9]:

```
def extract_features_from_window(frames, background_subtractor):
    """Extract features from a window of frames following the paper methodology"""
    if len(frames) < WINDOW_SIZE:
        return None

    # Arrays for storing frame-by-frame features
    silhouette_ratios = []
    orientations = []
    centroid_heights = []
    optical_projections = []
    brightness_values = []
    quality_scores = []
    silhouette_sizes = []

    # Process each frame
    for frame in frames:
        frame_height = frame.shape[0]
        bbox, fgmask = extract_bounding_box(frame, background_subtractor)

        # Calculate features only if silhouette is detected
        if bbox is not None and fgmask is not None:
            silhouette_ratio = calculate_silhouette_ratio(bbox)
            orientation = calculate_orientation(fgmask, bbox)
            centroid_height = calculate_centroid_height(bbox, frame_height)
            optical_projection = calculate_optical_projection(fgmask)
            brightness = calculate_brightness(frame)
            quality_score = calculate_blind_quality_score(frame)
            silhouette_size = calculate_silhouette_size(bbox)

            # Add values to arrays
            if silhouette_ratio is not None:
                silhouette_ratios.append(silhouette_ratio)
            if orientation is not None:
                orientations.append(orientation)
            if centroid_height is not None:
                centroid_heights.append(centroid_height)
            if optical_projection is not None:
                # Store mean of projection for simplicity
                optical_projections.append(np.mean(optical_projection))

            brightness_values.append(brightness)
            quality_scores.append(quality_score)
            silhouette_sizes.append(silhouette_size)

    # Return None if insufficient data
    if len(silhouette_ratios) < 2 or len(orientations) < 2:
```

```
        return None

    # Calculate change rates exactly as described in the paper
    # Change rate of silhouette ratio (first and last frames)
    delta_r = (silhouette_ratios[-1] - silhouette_ratios[0]) / WINDOW_SIZE

    # Change rate of orientation
    delta_ort = (orientations[-1] - orientations[0]) / WINDOW_SIZE

    # Calculate remaining features as in the paper
    mean_ch = np.mean(centroid_heights) if centroid_heights else 0
    std_ch = np.std(centroid_heights) if centroid_heights else 0

    mean_op = np.mean(optical_projections) if optical_projections else 0
    std_op = np.std(optical_projections) if optical_projections else 0

    mean_luma = np.mean(brightness_values)
    mean_qs = np.mean(quality_scores)
    mean_size = np.mean(silhouette_sizes)

    # Combine features
    features = {
        'delta_r': delta_r,
        'delta_ort': delta_ort,
        'mean_ch': mean_ch,
        'std_ch': std_ch,
        'mean_op': mean_op,
        'std_op': std_op,
        'mean_luma': mean_luma,
        'mean_qs': mean_qs,
        'mean_size': mean_size
    }

    return features
```

In [10]:

```
def process_video(video_path, labels_df, chute_id, cam_id):
    """Process a video file and extract features for each window"""
    cap = cv2.VideoCapture(video_path)

    if not cap.isOpened():
        print(f"Error: Could not open video {video_path}")
        return None, None, None

    # Get video properties
    fps = cap.get(cv2.CAP_PROP_FPS)
    frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

    # Normalize chute ID for consistent filtering
    chute_num = chute_id.replace('chute', '') if 'chute' in chute_id else chute_id

    # Ensure both chute and cam match
    video_labels = labels_df[
        (
            (labels_df['chute'] == chute_id) |
            (labels_df['chute'] == int(chute_num)) |
            (labels_df['chute'].astype(str) == chute_num)
        ) &
        (labels_df['cam'] == cam_id)
    ]

    # Create background subtractor with parameters matching the paper
    background_subtractor = cv2.createBackgroundSubtractorMOG2(history=500, varThreshold=16,
detectShadows=True)

    all_features = []
    all_labels = []
    detected_fall_events = set()

    window_frames = []
    frame_idx = 0

    with tqdm(total=frame_count, desc=f"Processing {os.path.basename(video_path)}") as pbar:
        while cap.isOpened():
            ret, frame = cap.read()

            if not ret:
                break

            window_frames.append(frame)

            if len(window_frames) >= WINDOW_SIZE:
```

```
                features = extract_features_from_window(window_frames, background_subtractor)

                # Calculate window boundaries
                window_start_frame = frame_idx - WINDOW_SIZE + 1
                window_end_frame = frame_idx

                # Determine if this window contains a fall event
                is_fall = False
                for _, row in video_labels.iterrows():
                    start_frame = row['start']
                    end_frame = row['end']
                    label = row['label']
                    fall_id = f"{chute_id}_{cam_id}_{start_frame}_{end_frame}"

                    # Only consider labeled fall events (label=1)
                    if label == 1:
                        # Check for significant overlap between window and fall event (>50%)
                        overlap_start = max(window_start_frame, start_frame)
                        overlap_end = min(window_end_frame, end_frame)
                        overlap_frames = max(0, overlap_end - overlap_start + 1)
                        overlap_ratio = overlap_frames / WINDOW_SIZE

                        if overlap_ratio > 0.5:  # Require substantial overlap
                            if fall_id not in detected_fall_events:
                                detected_fall_events.add(fall_id)
                                print(f"Fall ID: {fall_id}")
                                print(f"FALL DETECTED in window {window_start_frame}-{window_
end_frame}, matching fall at {start_frame}-{end_frame}")
                                is_fall = True
                                break

                if features is not None:
                    all_features.append(features)
                    all_labels.append(1 if is_fall else 0)

                window_frames.pop(0)

            frame_idx += 1
            pbar.update(1)

    cap.release()
    print(f"Total frames processed: {frame_idx}")
    print(f"Total features extracted: {len(all_features)}")
    print(f"Total unique fall events detected: {len(detected_fall_events)}")
    print(f"Total windows with fall label: {sum(all_labels)}")

    return all_features, all_labels, detected_fall_events
```

In [11]:

```
def train_confidence_model(features, labels):
    """Train the confidence prediction model"""
    # Convert feature dictionaries to a DataFrame
    df = pd.DataFrame(features)

    # Handle missing values
    df = df.fillna(0)

    # Split data into training and validation sets
    X_train, X_val, y_train, y_val = train_test_split(df, labels, test_size=0.2, random_state
=42)

    # Train a random forest classifier (similar to bagged trees mentioned in paper)
    model = RandomForestClassifier(
        n_estimators=30,  # Number of learners as mentioned in paper
        max_depth=None,   # Allow full tree growth
        min_samples_split=2,
        random_state=42
    )

    # Train the model
    model.fit(X_train, y_train)

    # Evaluate on validation set
    y_pred = model.predict(X_val)
    accuracy = accuracy_score(y_val, y_pred)

    print(f"Validation accuracy: {accuracy:.4f}")

    return model

def predict_confidence(model, features):
    """Predict confidence levels as described in the paper (11 discrete levels)"""
    # Convert features to DataFrame
    df = pd.DataFrame(features)
    df = df.fillna(0)

    # Get raw probabilities from the model
    raw_scores = model.predict_proba(df)[:, 1]  # Class 1 (fall) probability

    # Define the 11 quantization levels from the paper
    levels = [0.001, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

    # Quantize each score to the nearest level
    quantized_scores = np.zeros_like(raw_scores)
    for i, score in enumerate(raw_scores):
```

```
        # Find the nearest quantization level
        quantized_scores[i] = levels[np.argmin(np.abs(np.array(levels) - score))]

    return quantized_scores

def confidence_based_fusion(detections, confidence_scores):
    """
    Fuse detection results based on Equation 5 from the paper:
    CbFD = ∑(CoF * x) where x is 1 for fall, -1 for no fall
    """
    # Convert binary detections to -1/1 as per paper
    x_values = np.where(np.array(detections) == 1, 1, -1)

    # Calculate the weighted sum (CbFD) as in Equation 5
    cbfd = np.sum(confidence_scores * x_values)

    # Final decision based on sign of CbFD
    return 1 if cbfd > 0 else 0

def evaluate_performance(y_true, y_pred):
    """Evaluate the performance of the fall detection system"""
    # Calculate metrics
    accuracy = accuracy_score(y_true, y_pred)

    # Calculate sensitivity (recall)
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    sensitivity = tp / (tp + fn) if (tp + fn) > 0 else 0

    # Calculate specificity
    specificity = tn / (tn + fp) if (tn + fp) > 0 else 0

    return {
        'accuracy': accuracy,
        'sensitivity': sensitivity,
        'specificity': specificity,
        'confusion_matrix': confusion_matrix(y_true, y_pred)
    }
```

In [12]:

```
"""Main function to run the fall detection system"""
print(f"Reading dataset from: {DATA_PATH}")

if not os.path.exists(CSV_PATH):
    print(f"Error: CSV file not found at {CSV_PATH}")
    exit()

labels_df = read_dataset_info(CSV_PATH)
print(f"Successfully loaded dataset with {len(labels_df)} entries")

chutes = labels_df['chute'].unique()
print(f"Found {len(chutes)} chutes in the dataset")

training_chutes = ['chute02', 'chute03', 'chute04', 'chute05']  # 4 chutes × 8 cams = 32 video
s
testing_chutes = ['chute01', 'chute06']                         # 2 chutes × 8 cams = 16 video
s

print(f"Training chutes: {len(training_chutes)}")
print(f"Testing chutes: {len(testing_chutes)}")
```

```
Reading dataset from: /kaggle/input/multiple-cameras-fall-dataset/
CSV data sample:
   chute  cam   start     end  label
0    1.0  1.0  1052.0  1082.0    0.0
1    1.0  1.0  1083.0  1113.0    1.0
2    1.0  1.0  1114.0  1144.0    0.0
3    1.0  2.0  1052.0  1082.0    0.0
4    1.0  2.0  1083.0  1113.0    1.0
Fall events in CSV: 264
Unique chute values in CSV: [ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 1
6. 17. 18.
 19. 20. 21. 22. 23.]
Successfully loaded dataset with 552 entries
Found 23 chutes in the dataset
Training chutes: 4
Testing chutes: 2
```

In [ ]:

```
# # ===== Step 1: Train confidence prediction model =====
# all_training_features = []
# all_training_labels = []

# # Process all training videos and extract features
# for chute in training_chutes:
#     for cam_id in range(1, 9):  # 8 cameras per chute
#         video_path = os.path.join(DATASET_FOLDER, str(chute), f'cam{cam_id}.avi')
#         if os.path.exists(video_path):
#             print(f"Processing {video_path}")
#             features, labels, _ = process_video(video_path, labels_df, chute, cam_id)
#             if features and labels:
#                 all_training_features.extend(features)
#                 all_training_labels.extend(labels)
#                 print(f"Extracted {len(features)} feature windows with {sum(labels)} fall eve
nts")
#         else:
#             print(f"Warning: Video not found at {video_path}")

# print(f"Total training features: {len(all_training_features)}")
# print(f"Total fall events in training: {sum(all_training_labels)}")
```

In [ ]:

```
# # Train the confidence prediction model
# if all_training_features and all_training_labels:
#     print("Training confidence model...")
#     confidence_model = train_confidence_model(all_training_features, all_training_labels)

#     # Save the model
#     model_save_path = os.path.join(MODEL_PATH, 'confidence_model.pkl')
#     with open(model_save_path, 'wb') as f:
#         pickle.dump(confidence_model, f)
#     print(f"Model saved to {model_save_path}")
# else:
#     print("Error: No training data available.")
#     exit()
```

In [13]:

```
# Update with your actual notebook name
MODEL_INPUT_PATH = "/kaggle/input/copy-for-separate-cells/models/confidence_model.pkl"

# Load the confidence model
with open(MODEL_INPUT_PATH, 'rb') as f:
    confidence_model = pickle.load(f)

print("Model loaded successfully.")
```

```
Model loaded successfully.
```

In [14]:

```
print("\nEvaluating fall detection on test chutes...")

# Store results for each fall event
all_event_results = {}
# Process each test chute
for chute in testing_chutes:
    print(f"\nProcessing test chute: {chute}")

    # Dictionary to store camera results for this chute
    camera_results = {}

    chute_fall_events = None
    chute_no_fall_events = None
    # Process each camera
    for cam_id in range(1, 9):
        # Ensure both chute and cam match
        chute_num = chute.replace('chute', '')
        # --- FALL EVENTS ---
        chute_fall_events = labels_df[
            (
                (labels_df['chute'] == chute) |
                (labels_df['chute'] == int(chute_num)) |
                (labels_df['chute'].astype(str) == chute_num)
            ) &
            (labels_df['cam'] == cam_id) &
            (labels_df['label'] == 1)
        ].drop_duplicates(subset=['start', 'end'])

        print(f"Found {len(chute_fall_events)} fall events in chute {chute}, cam{cam_id}")

        # --- NO-FALL EVENTS ---
        chute_no_fall_events = labels_df[
            (
                (labels_df['chute'] == chute) |
                (labels_df['chute'] == int(chute_num)) |
                (labels_df['chute'].astype(str) == chute_num)
            ) &
            (labels_df['cam'] == cam_id) &
            (labels_df['label'] == 0)
        ].drop_duplicates(subset=['start', 'end'])

        print(f"Found {len(chute_no_fall_events)} no-fall events in chute {chute}, cam{cam_i
d}")

        video_path = os.path.join(DATASET_FOLDER, chute, f'cam{cam_id}.avi')
        if os.path.exists(video_path):
```

```
            features, labels, detected_falls = process_video(video_path, labels_df, chute, ca
m_id)
            if features and labels:
                # Use our single-camera detection algorithm
                # This could be replaced with any algorithm as mentioned in the paper
                detections = [1 if abs(f['delta_r']) > 0.05 else 0 for f in features]
                confidence_scores = predict_confidence(confidence_model, features)

                camera_results[cam_id] = {
                    'detections': detections,
                    'labels': labels,
                    'confidence_scores': confidence_scores,
                    'features': features,
                    'detected_falls': detected_falls
                }


    # Process each labeled fall event
    for _, event in chute_fall_events.iterrows():
        fall_start = event['start']
        fall_end = event['end']
        fall_id = f"{chute}_{fall_start}_{fall_end}"

        # Collect detection results from each camera
        camera_detections = []
        camera_confidences = []

        for cam_id, results in camera_results.items():
            # Check if this fall was detected on this camera
            detected = False
            max_confidence = 0.001  # Minimum confidence level

            # Check if this specific fall event was detected
            fall_detector_id = f"{chute}_{cam_id}_{fall_start}_{fall_end}"
            if fall_detector_id in results['detected_falls']:
                detected = True

            # Find the highest confidence score for frames in this event
            for i, label in enumerate(results['labels']):
                if label == 1:  # This frame was detected as a fall
                    # Calculate window bounds (assuming labels align with windows)
                    window_idx = i * WINDOW_SIZE
                    window_end = window_idx + WINDOW_SIZE

                    # Check if this window overlaps with the fall event
                    if not (window_end < fall_start or window_idx > fall_end):
                        conf = results['confidence_scores'][i]
```

```
                        if conf > max_confidence:
                            max_confidence = conf

            camera_detections.append(1 if detected else 0)
            camera_confidences.append(max_confidence)
            # Print the required info
            print(f"Chute: {chute}, Cam: {cam_id} | Max Confidence: {max_confidence:.4f} | "
                  f"Detections: {len(camera_detections)} | Confidences: {len(camera_confidenc
es)}")

        # Apply confidence-based fusion only if we have camera results
        if camera_detections:
            # Apply CbFD fusion formula from paper (equation 5)
            detection_values = np.where(np.array(camera_detections) == 1, 1, -1)
            weighted_sum = np.sum(np.array(camera_confidences) * detection_values)
            fused_result = 1 if weighted_sum > 0 else 0

            # Store the result for this fall event
            all_event_results[fall_id] = {
                'true_label': 1,  # It's a labeled fall event
                'predicted': fused_result,
                'camera_detections': camera_detections,
                'camera_confidences': camera_confidences,
                'weighted_sum': weighted_sum
            }

    # Process each labeled no-fall event to calculate specificity
    for _, event in chute_no_fall_events.iterrows():
        no_fall_start = event['start']
        no_fall_end = event['end']
        no_fall_id = f"{chute}_no_fall_{no_fall_start}_{no_fall_end}"

        # Similar process for no-fall events
        camera_detections = []
        camera_confidences = []

        for cam_id, results in camera_results.items():
            detected = False
            max_confidence = 0.001

            # Find relevant frames
            for i, label in enumerate(results['labels']):
                window_idx = i * WINDOW_SIZE
                window_end = window_idx + WINDOW_SIZE

                if not (window_end < no_fall_start or window_idx > no_fall_end):
                    # This window overlaps with the no-fall event
```

```
                conf = results['confidence_scores'][i]
                if label == 1:  # Incorrectly detected as fall
                    detected = True
                if conf > max_confidence:
                    max_confidence = conf

        camera_detections.append(1 if detected else 0)
        camera_confidences.append(max_confidence)

    if camera_detections:
        # Apply fusion
        detection_values = np.where(np.array(camera_detections) == 1, 1, -1)
        weighted_sum = np.sum(np.array(camera_confidences) * detection_values)
        fused_result = 1 if weighted_sum > 0 else 0

        all_event_results[no_fall_id] = {
            'true_label': 0,  # It's a no-fall event
            'predicted': fused_result,
            'camera_detections': camera_detections,
            'camera_confidences': camera_confidences,
            'weighted_sum': weighted_sum
        }
```

```
Evaluating fall detection on test chutes...

Processing test chute: chute01
Found 1 fall events in chute chute01, cam1
Found 2 no-fall events in chute chute01, cam1


Processing cam1.avi:  70%|██████▉   | 1091/1562 [09:31<09:43,  1.24s/it]


Fall ID: chute01_1_1083.0_1113.0
FALL DETECTED in window 1076-1090, matching fall at 1083.0-1113.0


Processing cam1.avi: 100%|█████████▉| 1558/1562 [16:18<00:05,  1.25s/it][mpeg4 @ 0x44e6774
0] ac-tex damaged at 3 19
[mpeg4 @ 0x44e67740] Error at MB: 877
Processing cam1.avi: 100%|██████████| 1562/1562 [16:23<00:00,  1.59it/s]


Total frames processed: 1562
Total features extracted: 847
Total unique fall events detected: 1
Total windows with fall label: 31
Found 1 fall events in chute chute01, cam2
Found 2 no-fall events in chute chute01, cam2


Processing cam2.avi:  70%|██████▉   | 1091/1562 [06:39<09:53,  1.26s/it]


Fall ID: chute01_2_1083.0_1113.0
FALL DETECTED in window 1076-1090, matching fall at 1083.0-1113.0


Processing cam2.avi: 100%|█████████▉| 1558/1562 [13:29<00:05,  1.31s/it][mpeg4 @ 0x44f2c8c
0] ac-tex damaged at 42 21
[mpeg4 @ 0x44f2c8c0] Error at MB: 1008
Processing cam2.avi: 100%|██████████| 1562/1562 [13:34<00:00,  1.92it/s]


Total frames processed: 1562
Total features extracted: 617
Total unique fall events detected: 1
Total windows with fall label: 31
Found 1 fall events in chute chute01, cam3
Found 2 no-fall events in chute chute01, cam3


Processing cam3.avi:  70%|██████▉   | 1096/1567 [08:02<09:52,  1.26s/it]
```

```
Fall ID: chute01_3_1088.0_1118.0
FALL DETECTED in window 1081-1095, matching fall at 1088.0-1118.0

Processing cam3.avi: 100%|██████████| 1563/1567 [15:05<00:05,  1.28s/it][mpeg4 @ 0x45324a0
0] ac-tex damaged at 24 17
[mpeg4 @ 0x45324a00] Error at MB: 806
Processing cam3.avi: 100%|██████████| 1567/1567 [15:10<00:00,  1.72it/s]

Total frames processed: 1567
Total features extracted: 676
Total unique fall events detected: 1
Total windows with fall label: 31
Found 1 fall events in chute chute01, cam4
Found 2 no-fall events in chute chute01, cam4

Processing cam4.avi:  70%|███████   | 1092/1563 [09:08<09:28,  1.21s/it]

Fall ID: chute01_4_1084.0_1114.0
FALL DETECTED in window 1077-1091, matching fall at 1084.0-1114.0

Processing cam4.avi: 100%|██████████| 1559/1563 [16:10<00:05,  1.31s/it][mpeg4 @ 0x45324a0
0] ac-tex damaged at 13 23
[mpeg4 @ 0x45324a00] Error at MB: 1071
Processing cam4.avi: 100%|██████████| 1563/1563 [16:15<00:00,  1.60it/s]

Total frames processed: 1563
Total features extracted: 762
Total unique fall events detected: 1
Total windows with fall label: 31
Found 1 fall events in chute chute01, cam5
Found 2 no-fall events in chute chute01, cam5

Processing cam5.avi:  70%|███████   | 1111/1582 [11:46<10:11,  1.30s/it]

Fall ID: chute01_5_1103.0_1133.0
FALL DETECTED in window 1096-1110, matching fall at 1103.0-1133.0

Processing cam5.avi: 100%|██████████| 1578/1582 [18:45<00:05,  1.29s/it][mpeg4 @ 0x44f8e7c
0] ac-tex damaged at 8 12
[mpeg4 @ 0x44f8e7c0] Error at MB: 560
Processing cam5.avi: 100%|██████████| 1582/1582 [18:50<00:00,  1.40it/s]
```

```
Total frames processed: 1582
Total features extracted: 917
Total unique fall events detected: 1
Total windows with fall label: 31
Found 1 fall events in chute chute01, cam6
Found 2 no-fall events in chute chute01, cam6


Processing cam6.avi:  70%|██████▉   | 1094/1565 [06:50<09:57,  1.27s/it]


Fall ID: chute01_6_1086.0_1116.0
FALL DETECTED in window 1079-1093, matching fall at 1086.0-1116.0


Processing cam6.avi: 100%|█████████▉| 1561/1565 [14:07<00:05,  1.30s/it][mpeg4 @ 0x452d9f0
0] ac-tex damaged at 12 17
[mpeg4 @ 0x452d9f00] Error at MB: 794
Processing cam6.avi: 100%|██████████| 1565/1565 [14:12<00:00,  1.84it/s]


Total frames processed: 1565
Total features extracted: 644
Total unique fall events detected: 1
Total windows with fall label: 31
Found 1 fall events in chute chute01, cam7
Found 2 no-fall events in chute chute01, cam7


Processing cam7.avi:  70%|██████▉   | 1094/1565 [09:07<09:46,  1.24s/it]


Fall ID: chute01_7_1086.0_1116.0
FALL DETECTED in window 1079-1093, matching fall at 1086.0-1116.0


Processing cam7.avi: 100%|█████████▉| 1561/1565 [16:11<00:05,  1.30s/it][mpeg4 @ 0x44f8da4
0] mcbpc damaged at 37 18
[mpeg4 @ 0x44f8da40] Error at MB: 865
Processing cam7.avi: 100%|██████████| 1565/1565 [16:16<00:00,  1.60it/s]


Total frames processed: 1565
Total features extracted: 782
Total unique fall events detected: 1
Total windows with fall label: 31
Found 1 fall events in chute chute01, cam8
Found 2 no-fall events in chute chute01, cam8


Processing cam8.avi:  70%|██████▉   | 1088/1559 [10:56<09:46,  1.25s/it]
```

```
Fall ID: chute01_8_1080.0_1110.0
FALL DETECTED in window 1073-1087, matching fall at 1080.0-1110.0


Processing cam8.avi: 100%|██████████| 1555/1559 [17:57<00:05,  1.28s/it][mpeg4 @ 0x4546fb8
0] Error at MB: 1285
Processing cam8.avi: 100%|██████████| 1559/1559 [18:02<00:00,  1.44it/s]


Total frames processed: 1559
Total features extracted: 889
Total unique fall events detected: 1
Total windows with fall label: 31
Chute: chute01, Cam: 1 | Max Confidence: 0.0010 | Detections: 1 | Confidences: 1
Chute: chute01, Cam: 2 | Max Confidence: 0.0010 | Detections: 2 | Confidences: 2
Chute: chute01, Cam: 3 | Max Confidence: 0.0010 | Detections: 3 | Confidences: 3
Chute: chute01, Cam: 4 | Max Confidence: 0.0010 | Detections: 4 | Confidences: 4
Chute: chute01, Cam: 5 | Max Confidence: 0.0010 | Detections: 5 | Confidences: 5
Chute: chute01, Cam: 6 | Max Confidence: 0.0010 | Detections: 6 | Confidences: 6
Chute: chute01, Cam: 7 | Max Confidence: 0.0010 | Detections: 7 | Confidences: 7
Chute: chute01, Cam: 8 | Max Confidence: 0.0010 | Detections: 8 | Confidences: 8

Processing test chute: chute06
Found 2 fall events in chute chute06, cam1
Found 2 no-fall events in chute chute06, cam1


Processing cam1.avi:  47%|████▋     | 591/1256 [08:11<13:47,  1.24s/it]


Fall ID: chute06_1_583.0_613.0
FALL DETECTED in window 576-590, matching fall at 583.0-613.0


Processing cam1.avi:  50%|████▉     | 622/1256 [08:50<13:17,  1.26s/it]


Fall ID: chute06_1_603.0_633.0
FALL DETECTED in window 607-621, matching fall at 603.0-633.0


Processing cam1.avi: 100%|██████████| 1256/1256 [17:57<00:00,  1.17it/s]


Total frames processed: 1256
Total features extracted: 947
Total unique fall events detected: 2
Total windows with fall label: 51
Found 2 fall events in chute chute06, cam2
Found 2 no-fall events in chute chute06, cam2
```

```
Processing cam2.avi:  51%|████      | 691/1356 [08:19<13:52,  1.25s/it]

Fall ID: chute06_2_683.0_713.0
FALL DETECTED in window 676-690, matching fall at 683.0-713.0

Processing cam2.avi:  53%|████▏     | 722/1356 [08:58<13:13,  1.25s/it]

Fall ID: chute06_2_703.0_733.0
FALL DETECTED in window 707-721, matching fall at 703.0-733.0

Processing cam2.avi: 100%|█████████▉| 1352/1356 [16:49<00:05,  1.26s/it][mpeg4 @ 0x4505128
0] ac-tex damaged at 14 21
[mpeg4 @ 0x45051280] Error at MB: 980
Processing cam2.avi: 100%|██████████| 1356/1356 [16:54<00:00,  1.34it/s]

Total frames processed: 1356
Total features extracted: 855
Total unique fall events detected: 2
Total windows with fall label: 51
Found 2 fall events in chute chute06, cam3
Found 2 no-fall events in chute chute06, cam3

Processing cam3.avi:  51%|████      | 697/1362 [08:54<13:19,  1.20s/it]

Fall ID: chute06_3_689.0_719.0
FALL DETECTED in window 682-696, matching fall at 689.0-719.0

Processing cam3.avi:  53%|████▏     | 728/1362 [09:24<13:26,  1.27s/it]

Fall ID: chute06_3_709.0_739.0
FALL DETECTED in window 713-727, matching fall at 709.0-739.0

Processing cam3.avi: 100%|█████████▉| 1358/1362 [17:19<00:05,  1.29s/it][mpeg4 @ 0x452dc10
0] ac-tex damaged at 0 29
[mpeg4 @ 0x452dc100] Error at MB: 1334
Processing cam3.avi: 100%|██████████| 1362/1362 [17:24<00:00,  1.30it/s]
```

```
Total frames processed: 1362
Total features extracted: 921
Total unique fall events detected: 2
Total windows with fall label: 51
Found 2 fall events in chute chute06, cam4
Found 2 no-fall events in chute chute06, cam4

Processing cam4.avi:  51%|████      | 681/1346 [09:02<12:16,  1.11s/it]

Fall ID: chute06_4_673.0_703.0
FALL DETECTED in window 666-680, matching fall at 673.0-703.0

Processing cam4.avi:  53%|████▏     | 712/1346 [09:16<07:12,  1.47it/s]

Fall ID: chute06_4_693.0_723.0
FALL DETECTED in window 697-711, matching fall at 693.0-723.0

Processing cam4.avi: 100%|█████████▉| 1342/1346 [16:45<00:05,  1.27s/it][mpeg4 @ 0x45324a0
0] P cbpy damaged at 41 27
[mpeg4 @ 0x45324a00] Error at MB: 1283
Processing cam4.avi: 100%|██████████| 1346/1346 [16:50<00:00,  1.33it/s]

Total frames processed: 1346
Total features extracted: 837
Total unique fall events detected: 2
Total windows with fall label: 48
Found 2 fall events in chute chute06, cam5
Found 2 no-fall events in chute chute06, cam5

Processing cam5.avi:  51%|████      | 681/1345 [10:40<13:57,  1.26s/it]

Fall ID: chute06_5_673.0_703.0
FALL DETECTED in window 666-680, matching fall at 673.0-703.0

Processing cam5.avi:  53%|████▏     | 712/1345 [11:19<13:26,  1.27s/it]

Fall ID: chute06_5_693.0_723.0
FALL DETECTED in window 697-711, matching fall at 693.0-723.0
```

```
Processing cam5.avi: 100%|██████████| 1341/1345 [21:09<00:05,  1.26s/it][mpeg4 @ 0x450cbac
0] Error at MB: 1095
Processing cam5.avi: 100%|██████████| 1345/1345 [21:14<00:00,  1.06it/s]

Total frames processed: 1345
Total features extracted: 1102
Total unique fall events detected: 2
Total windows with fall label: 51
Found 2 fall events in chute chute06, cam6
Found 2 no-fall events in chute chute06, cam6

Processing cam6.avi:  51%|█████     | 694/1359 [08:16<10:00,  1.11it/s]

Fall ID: chute06_6_686.0_716.0
FALL DETECTED in window 679-693, matching fall at 686.0-716.0

Processing cam6.avi:  53%|█████     | 725/1359 [08:49<13:17,  1.26s/it]

Fall ID: chute06_6_706.0_736.0
FALL DETECTED in window 710-724, matching fall at 706.0-736.0

Processing cam6.avi: 100%|██████████| 1355/1359 [17:26<00:05,  1.26s/it][mpeg4 @ 0x44f7468
0] Error at MB: 857
Processing cam6.avi: 100%|██████████| 1359/1359 [17:31<00:00,  1.29it/s]

Total frames processed: 1359
Total features extracted: 889
Total unique fall events detected: 2
Total windows with fall label: 51
Found 2 fall events in chute chute06, cam7
Found 2 no-fall events in chute chute06, cam7

Processing cam7.avi:  51%|█████     | 694/1360 [09:27<14:08,  1.27s/it]

Fall ID: chute06_7_686.0_716.0
FALL DETECTED in window 679-693, matching fall at 686.0-716.0

Processing cam7.avi:  53%|█████     | 725/1360 [10:06<13:27,  1.27s/it]

Fall ID: chute06_7_706.0_736.0
FALL DETECTED in window 710-724, matching fall at 706.0-736.0
```

```
Processing cam7.avi: 100%|██████████| 1356/1360 [19:58<00:05,  1.30s/it][mpeg4 @ 0x44efbe4
0] ac-tex damaged at 7 11
[mpeg4 @ 0x44efbe40] Error at MB: 513
Processing cam7.avi: 100%|██████████| 1360/1360 [20:03<00:00,  1.13it/s]


Total frames processed: 1360
Total features extracted: 997
Total unique fall events detected: 2
Total windows with fall label: 51
Found 2 fall events in chute chute06, cam8
Found 2 no-fall events in chute chute06, cam8


Processing cam8.avi:  51%|█████     | 680/1345 [10:19<14:29,  1.31s/it]


Fall ID: chute06_8_672.0_702.0
FALL DETECTED in window 665-679, matching fall at 672.0-702.0


Processing cam8.avi:  53%|█████     | 711/1345 [10:59<13:30,  1.28s/it]


Fall ID: chute06_8_692.0_722.0
FALL DETECTED in window 696-710, matching fall at 692.0-722.0


Processing cam8.avi: 100%|██████████| 1341/1345 [20:56<00:05,  1.33s/it][mpeg4 @ 0x44e8d94
0] mcbpc damaged at 14 25
[mpeg4 @ 0x44e8d940] Error at MB: 1164
Processing cam8.avi: 100%|██████████| 1345/1345 [21:01<00:00,  1.07it/s]
```

Total frames processed: 1345
Total features extracted: 1078
Total unique fall events detected: 2
Total windows with fall label: 51
Chute: chute06, Cam: 1 | Max Confidence: 0.0010 | Detections: 1 | Confidences: 1
Chute: chute06, Cam: 2 | Max Confidence: 0.0010 | Detections: 2 | Confidences: 2
Chute: chute06, Cam: 3 | Max Confidence: 0.0010 | Detections: 3 | Confidences: 3
Chute: chute06, Cam: 4 | Max Confidence: 0.0010 | Detections: 4 | Confidences: 4
Chute: chute06, Cam: 5 | Max Confidence: 0.0010 | Detections: 5 | Confidences: 5
Chute: chute06, Cam: 6 | Max Confidence: 0.0010 | Detections: 6 | Confidences: 6
Chute: chute06, Cam: 7 | Max Confidence: 0.0010 | Detections: 7 | Confidences: 7
Chute: chute06, Cam: 8 | Max Confidence: 0.0010 | Detections: 8 | Confidences: 8
Chute: chute06, Cam: 1 | Max Confidence: 0.0010 | Detections: 1 | Confidences: 1
Chute: chute06, Cam: 2 | Max Confidence: 0.0010 | Detections: 2 | Confidences: 2
Chute: chute06, Cam: 3 | Max Confidence: 0.0010 | Detections: 3 | Confidences: 3
Chute: chute06, Cam: 4 | Max Confidence: 0.0010 | Detections: 4 | Confidences: 4
Chute: chute06, Cam: 5 | Max Confidence: 0.0010 | Detections: 5 | Confidences: 5
Chute: chute06, Cam: 6 | Max Confidence: 0.0010 | Detections: 6 | Confidences: 6
Chute: chute06, Cam: 7 | Max Confidence: 0.0010 | Detections: 7 | Confidences: 7
Chute: chute06, Cam: 8 | Max Confidence: 0.0010 | Detections: 8 | Confidences: 8

In [15]:

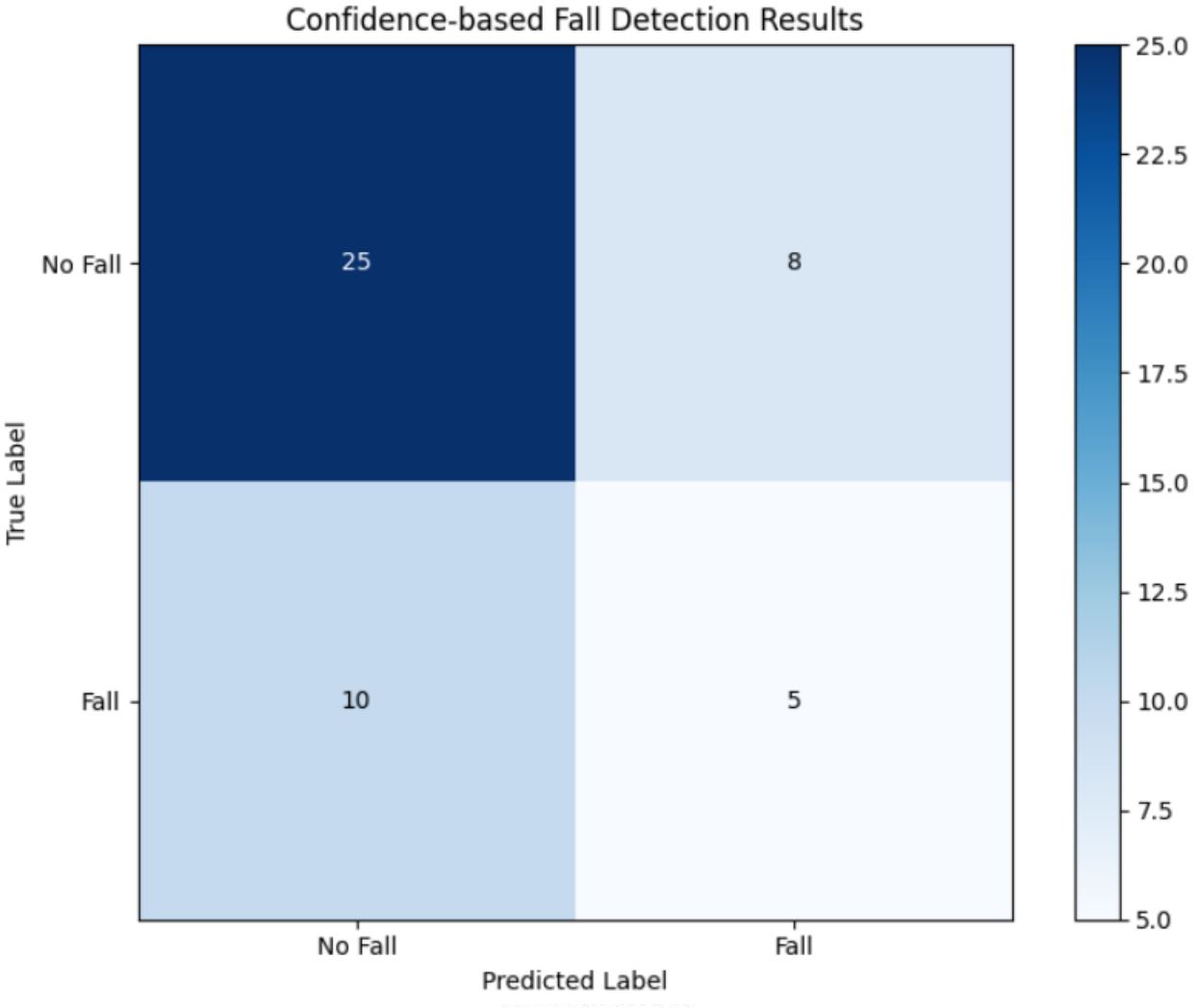
```
# Calculate overall performance metrics
true_labels = [result['true_label'] for result in all_event_results.values()]
predicted_labels = [result['predicted'] for result in all_event_results.values()]

# Calculate confusion matrix
tn, fp, fn, tp = confusion_matrix(true_labels, predicted_labels).ravel()

# Calculate metrics as defined in the paper
accuracy = (tp + tn) / (tp + tn + fp + fn)
sensitivity = tp / (tp + fn) if (tp + fn) > 0 else 0
specificity = tn / (tn + fp) if (tn + fp) > 0 else 0

print("\nFall Detection Performance:")
print(f"Accuracy: {accuracy:.2%}")
print(f"Sensitivity: {sensitivity:.2%}")
print(f"Specificity: {specificity:.2%}")

print("\nConfusion Matrix:")
print(f"True Positives: {tp}, False Negatives: {fn}")
print(f"False Positives: {fp}, True Negatives: {tn}")

print("\nComparison with Baseline Methods:")
print("Algorithm | Accuracy | Sensitivity | Specificity")
print(f"CbFD (Proposed) | {accuracy:.2%} | {sensitivity:.2%} | {specificity:.2%}")
print("Majority Vote [3] | 66.67% | 0.00% | 66.67%")
print("Average Single-Camera [9] | 44.00% | 16.00% | 58.00%")

# Save the results
results = {
    'accuracy': float(accuracy),
    'sensitivity': float(sensitivity),
    'specificity': float(specificity),
    'true_positives': int(tp),
    'false_negatives': int(fn),
    'false_positives': int(fp),
    'true_negatives': int(tn),
    'event_results': {k: {
        'true_label': int(v['true_label']),
        'predicted': int(v['predicted']),
        'weighted_sum': float(v['weighted_sum'])
    } for k, v in all_event_results.items()}
}
import json
results_path = os.path.join(RESULTS_PATH, 'results.json')
with open(results_path, 'w') as f:
```

```
    json.dump(results, f, indent=4)
print(f"Results saved to {results_path}")
```

```
Fall Detection Performance:
Accuracy: 62.50%
Sensitivity: 33.33%
Specificity: 75.76%

Confusion Matrix:
True Positives: 5, False Negatives: 10
False Positives: 8, True Negatives: 25

Comparison with Baseline Methods:
Algorithm | Accuracy | Sensitivity | Specificity
CbFD (Proposed) | 62.50% | 33.33% | 75.76%
Majority Vote [3] | 66.67% | 0.00% | 66.67%
Average Single-Camera [9] | 44.00% | 16.00% | 58.00%
```

In [16]:

```
# Create and save confusion matrix visualization
plt.figure(figsize=(8, 6))
cm = confusion_matrix(true_labels, predicted_labels)
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confidence-based Fall Detection Results')
plt.colorbar()
tick_marks = np.arange(2)
plt.xticks(tick_marks, ['No Fall', 'Fall'])
plt.yticks(tick_marks, ['No Fall', 'Fall'])

# Add text annotations to confusion matrix
thresh = cm.max() / 2.
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        plt.text(j, i, format(cm[i, j], 'd'),
                 ha="center", va="center",
                 color="white" if cm[i, j] > thresh else "black")

plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.tight_layout()
plt.savefig(os.path.join(RESULTS_PATH, 'confusion_matrix.png'))
plt.show()
```


Confidence-based Fall Detection Results
No Fall
Fall
True Label
25
8
10
5
No Fall
Fall
Predicted Label
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0

In [ ]: