Wydział Inżynierii Mechanicznej i Robotyki

Katedra Robotyki i Mechatroniki

Podstawy sztucznej inteligencji i uczenia głębokiego
Kurs dla kierunku Inżynieria Mechatroniczna

Instrukcja 9:

Przetwarzanie wideo
i
Śledzenie obiektów/cech

Nauczysz się: jak ładować i przetwarzać wideo w programie MATLAB, jak wykonywać śledzenie obiektów na podstawie cech obiektów obrazu binarnego, jak stosować miarę korelacji do śledzenia obiektów, czym są cechy niezmienne (SIFT, SURF) i jak ich używać do śledzenia obiektów, czym jest przepływ optyczny i jak obliczać pole przepływu optycznego z klatek sekwencji wideo.

Materiały dodatkowe:

- Wykład kursowy

Opiekun przedmiotu:
Krzysztof Holak, holak @agh.edu.pl

Autor instrukcji:
Krzysztof Holak, holak @agh.edu.pl

Śledzenie obiektów

Wykrywanie ruchu i śledzenie obiektów to najważniejsze zastosowania przetwarzania wideo. W tym ćwiczeniu poznasz podstawy analizy ruchu za pomocą MATLAB-a. Zaczniesz od zadania śledzenia obiektów w sekwencji klatek wideo.

Śledzenie obiektów na binarnej sekwencji wideo

Aby wykonać to ćwiczenie, zastosujesz metody, których nauczyłeś się już w poprzednich ćwiczeniach. W przypadku wideo przetwarzanie i analiza obrazu muszą być zastosowane do każdej klatki sekwencji filmowej. W tym laboratorium nauczysz się, jak przetwarzać pliki wideo w MATLAB-ie.

W MATLAB-ie przechowujemy klatki wideo w obiekcie o nazwie **`VideoReader.`** Utwórzmy ten obiekt, wywołując funkcję

```
video = VideoReader(filename);
```

Zamiast filename podaj ścieżkę do pliku wideo, jak w przypadku ładowania pojedynczych obrazów. Obiekt wideo typu **`VideoReader`** zawiera różne informacje o sekwencji wideo. Na przykład,
`video.FrameRate` zawiera informacje o liczbie klatek na sekundę,
`video.NumFrames` zawiera informacje o liczbie klatek w sekwencji,
`video.Height , video.Width` zawiera informacje o rozdzielczości każdej klatki wideo.

Możesz odtworzyć wideo w oknie MATLAB za pomocą prostego kodu, na przykład

```
currAxes = axes;

while hasFrame (video)

    vidFrame = readFrame(video);
    image(vidFrame,"Parent",currAxes)
    currAxes.Visible = "off";
    pause(1/v.FrameRate)

end
```

`hasFrame()` służy do sprawdzenia, czy w sekwencji pozostały jeszcze klatki, a zmienna **`v.FrameRate`** określa odpowiedni czas wyświetlania klatki.

`readFrame()` służy do pobierania z filmu jednej klatki na raz.

Zadanie 4.1 Napisz prosty tracker, który oblicza trajektorię obiektu z sekwencji wideo. Znajdź środek ciężkości obiektu w każdej klatce filmu. Mając dane, narysuj trajektorię obiektu w 2D i pokaż jego przemieszczenie wzdłuż osi x i y jako funkcję czasu na wykresach. Spróbuj określić prędkość i przyspieszenie obiektu - poprzez różniczkowanie numeryczne.

Przykład prostego śledzenia

```
clear all
clc
close all

video = VideoReader('bouncing_ball.mp4'); % reading video file

% % info on video
% video.FrameRate;
% video.NumFrames;
% video.Height;
% video.Width;


% Finding a drawing trajectory of the centroid
index = 1;

figure,
h = imshow(rgb2gray(read(video,1))); hold on
h1 = plot(0,0,'*'); hold on


while hasFrame (video) % look for video frame processing

    vidFrame = readFrame(video);

    im = rgb2gray(vidFrame);
    imBW = im > 10;   % image thresholding, threshold set to 10 in this example

    imL = bwlabel(imBW); % labelling and object analysis - we need centroid
    features = regionprops(imL, 'centroid');
    Centroids(index,:) = features.Centroid;

    set(h, 'CData', im);  % Update the frame % how to update data in figures
    set(h1,'XData',Centroids(index,1),'YData',Centroids(index,2));
    title(['Time ' num2str(video.CurrentTime)]);
    drawnow;
    index = index + 1;

end
```

Śledzenie za pomocą korelacji obrazów

Jak wiadomo z poprzednich ćwiczeń, można znaleźć wzorzec na obrazie, używając miary podobieństwa, na przykład znormalizowanego współczynnika korelacji. Po zastosowaniu do sekwencji wideo, podejście to można wykorzystać do śledzenia wzorca na klatkach sekwencji.
Aby rozpocząć ćwiczenie, musisz załadować sekwencję wideo za pomocą funkcji **`VideoReader(),`** jak w poprzednim ćwiczeniu. Następnie odczytajmy pierwszą klatkę sekwencji i przekonwertujmy ją na skalę szarości.

```
% Read the first frame and convert it to grayscale
frame1 = read(video, 1);
frame1_gray = rgb2gray(frame1);
```

Teraz możesz wyświetlić klatkę i wybrać obiekt, który ma być śledzony w sekwencji. Aby to zrobić, użyj funkcji **`imcrop()`** omówionej w pierwszych ćwiczeniach laboratoryjnych dotyczących widzenia komputerowego i przetwarzania obrazu. Załóżmy, że przycięty wzór jest przechowywany w zmiennej **`template`** a jego rozmiar w zmiennej **`rectangle.`**

Możesz zainicjować śledzenie. Przygotuj zmienne do śledzenia wzorca.

```
objectPosition = rect;  % Store initial position
```

```
trackedPositions = objectPosition;  % To store the positions
```

Śledzenie można zobrazować na rysunku.

```
figure;
h = imshow(frame1_gray); hold on;
rect_handle = rectangle('Position', objectPosition, 'EdgeColor',
'r', 'LineWidth', 2);
```

Kontynuuj dla wszystkich klatek w sekwencji.

```
clear all
clc
close all

video01 = VideoReader('dvdscreensaver.mp4');

% Read the first frame and convert it to grayscale
frame1 = read(video01, 1);
frame1_gray = rgb2gray(frame1);

[template objectPosition] = imcrop(frame1_gray);

 % Store initial position
trackedPositions = objectPosition;

% visualization of tracking process

figure;
h = imshow(frame1_gray); %hold on;
rect_handle = rectangle('Position', objectPosition, 'EdgeColor', 'r', 'LineWidth', 2);

% Process each subsequent frame in the video
while hasFrame(video01)

    % Read the next frame and convert to grayscale
    frame2 = readFrame(video01);
    frame2_gray = rgb2gray(frame2);

    % Perform normalized cross-correlation (NCC) for template matching
    correlationMap = normxcorr2(template, frame2_gray);

    % Find the peak of the correlation (most similar region)
    [maxCorr, imax] = max(abs(correlationMap(:)));
    [ypeak, xpeak] = ind2sub(size(correlationMap), imax(1));

    % Compute the offset (translation) from the correlation peak
    corr_offset = [xpeak - size(template, 2), ypeak - size(template, 1)];

    % Update the object's position
    objectPosition = [corr_offset, size(template, 2), size(template, 1)];
    trackedPositions = [trackedPositions; objectPosition];  % Append new position

    % Update the displayed rectangle for the new object position
    set(rect_handle, 'Position', objectPosition);
    set(h, 'CData', frame2_gray);  % Update the frame
    title(['Time ' num2str(video01.CurrentTime)]);
    drawnow;

end
```

Zadanie 4.2 Napisz funkcję, która śledzi szablon na klatkach sekwencji wideo. Wizualizuj śledzenie za pomocą kodu podanego w instrukcji laboratoryjnej. Użytkownik powinien być w stanie ręcznie wybrać dowolny prostokątny wzór z pierwszej klatki sekwencji. Zmodyfikuj funkcję w taki sposób, aby rysowała trajektorię śledzonego obiektu w jednej z klatek sekwencji.

Śledzenie obiektów za pomocą przepływu optycznego

W przypadku pracy z filmami w skali szarości ze złożonymi scenami, generowanie ramek obrazu binarnego z obiektami przy użyciu progowania lub innych metod segmentacji może być bardzo trudne lub niemożliwe. Ponadto, jeśli wygląd obiektu zmienia się podczas ruchu, może to prowadzić do błędnego śledzenia, gdy stosowana jest korelacja obrazu. Ponadto, wszystkie dotychczas omówione śledzenie wymaga wykrywania wzorca albo za pomocą analizy obrazu binarnego, albo poprzez ręczne wybranie szablonu do śledzenia. Możliwe jest automatyczne analizowanie wideo w celu wykrywania interesujących obiektów na podstawie samego ruchu ich pikseli.
Możesz zastosować metodę estymacji ruchu, a w następnym kroku progować każdą klatkę wideo do obrazu binarnego na podstawie wartości przemieszczeń obiektów.

Jedną z najczęściej stosowanych metod estymacji ruchu jest tzw. przepływ optyczny. Wykorzystuje ona kolejne klatki sekwencji wideo do oszacowania przemieszczenia pikseli. W wersji podstawowej zakłada, że jasność poruszających się obiektów nie zmienia się między klatkami (założenie stałości jasności). Matematycznie wykorzystuje podobne podejście jak w obliczeniach detektora Harrisa. Jednym z wyzwań przepływu optycznego jest to, że pojedynczy piksel daje jedno równanie przepływu optycznego, a dla każdego piksela należy oszacować dwa parametry ruchu (współrzędne u, v). Dlatego zakłada się, że piksele w bliskim sąsiedztwie poruszają się tak samo, a wartość przemieszczenia znajduje się, łącząc równania przepływu optycznego dla każdego z nich w jednej macierzy. Rozwiązanie uzyskuje się w sensie najmniejszych kwadratów.

Zobaczmy, jak można przeprowadzić segmentację obrazu i śledzenie obiektów za pomocą metody przepływu optycznego.

Ponownie, najpierw musisz załadować plik wideo używając **`videoReader`** i odczytać pierwszą klatkę, konwertując ją na obraz w skali szarości.
Następnie należy zainicjować obiekt przepływu. W przykładzie używamy metody Farnebacka . Można również przetestować inne algorytmy przepływu optycznego dostępne w MATLAB-e (klasyczne Lukas-Kanade, Horn- Shunck).

```
% Initialize optical flow object
opticFlow = opticalFlowFarneback;

figure;
h = imshow(frame1_gray);
hold on;
```

Następnie oszacujmy ruch dla każdej klatki sekwencji wideo i znajdźmy obszary, które podlegają największemu ruchowi. Będą to nasze obiekty do śledzenia w sekwencji wideo. Oto fragment kodu, który wykonuje śledzenie na podstawie szacowanego ruchu obliczonego przez przepływ optyczny.

```
clear all
clc
close all

% Load video
```

```
videoFile = 'dvdscreensaver.mp4'; % path to video file
videoObj = VideoReader(videoFile);

% Display video
figure;
hold on;

% Farneback optical flow initialization
opticFlow = opticalFlowFarneback;

% A variable to store centroids
centroidsPrev = [];

while hasFrame(videoObj)
    % Read a frame
    frame = readFrame(videoObj);

    % Change into grayscale
    grayFrame = rgb2gray(frame);

    % Compute Optical flow
    flow = estimateFlow(opticFlow, grayFrame);

    % Compute motion mask based on a magnitude of optical flow
    motionMask = sqrt(flow.Vx.^2 + flow.Vy.^2) > 2; % Próg ruchu

    % Find object contours
    stats = regionprops(motionMask, 'Centroid', 'Area','BoundingBox');

    % Remove small objects - noise removal operation - may use morphology
    % before regionprops
    stats = stats([stats.Area] > 100); % Set area threshold

    % Show objects
    imshow(frame);
    hold on;

    % If there are objects, track them
    if ~isempty(stats)
        % Draw centroids of objects
        for i = 1:length(stats)
            centroid = stats(i).Centroid;
            plot(centroid(1), centroid(2), 'ro'); % Centroid
        end
    end

    % Display video
    pause(1/videoObj.FrameRate);
end
```

Zmodyfikuj kod, aby zapisać środek ciężkości każdego śledzonego obiektu dla wszystkich klatek sekwencji.

Zadanie 4.3 Napisz program do śledzenia obiektów w sekwencji wideo przy użyciu wybranej metody przepływu optycznego. Użyj zmodyfikowanej wersji, która przechowuje centroidy i pola ograniczające śledzonych obiektów. Narysuj trajektorię obiektu.

Można wizualizować ruch obiektów bez progowania i segmentowania sceny na obiekty. Możesz pokazać ruch każdego piksela w sekwencji obrazów, używając narzędzia MATLAB **`quiver()`** do wizualizacji pól przepływów. Możesz również użyć mapy cieplnej, aby pokazać wielkość ruchu – używając **`imshow()`** lub innej funkcji wizualizacyjnej, której nauczyłeś się w poprzednich ćwiczeniach.
Dlatego dla każdej klatki najpierw zwizualizuj wektory ruchu

```
flow_x = flow.Vx;
flow_y = flow.Vy;
```

```
[rows, cols] = size(frame2_gray); hold on;
quiver(repmat((1:cols), rows, 1), repmat((1:rows)', 1, cols),
flow_x, flow_y, 'r', 'MaxHeadSize', 1.5, 'LineWidth', 1.5);
```

I zobacz wielkość przepływu optycznego, korzystając z podejścia mapy cieplnej

```
imshow(flow_magnitude, []); %Display flow magnitude as a heatmap
```

Prosty program do wizualizacji wyników śledzenia przepływu optycznego – do porównywania metod przepływu optycznego dostępnych w MATLAB. Należy zwrócić uwagę, które części obiektów są uznawane za poruszające się w każdej z dostępnych metod przepływu optycznego.

```
clear all
clc
close all

video = VideoReader('dvdscreensaver.mp4');
% bouncing_ball.mp4

frame1 = read(video, 1);
frame1_gray = rgb2gray(frame1);

% Initialize optical flow object
% opticFlow = opticalFlowHS;
opticFlow = opticalFlowFarneback;
% opticFlow = opticalFlowLK;

figure;
h = imshow(frame1_gray); hold on;
h1 = quiver(0,0,0,0,'r', 'MaxHeadSize', 1.5, 'LineWidth', 1.5);

while hasFrame(video)

 % Read the next frame and convert to grayscale
    frame2 = readFrame(video);
    frame2_gray = rgb2gray(frame2);

     [rows, cols] = size(frame2_gray); hold on;

    % Compute optical flow between the current and previous frames
    flow = estimateFlow(opticFlow, frame2_gray);

    flow_x = flow.Vx;
    flow_y = flow.Vy;

    % Compute the magnitude of the flow vectors to detect moving objects
    flow_magnitude = sqrt(flow.Vx.^2 + flow.Vy.^2);

    % Update the figure with the current frame
    set(h, 'CData', frame2_gray);  % Update the frame
    set(h1,'UData',flow_x,'VData',flow_y,'XData',repmat((1:cols), rows,
1),'YData',repmat((1:rows)', 1, cols));
    %quiver(repmat((1:cols), rows, 1), repmat((1:rows)', 1, cols), flow_x, flow_y, 'r',
'MaxHeadSize', 1.5, 'LineWidth', 1.5);
    title(['Frame ' num2str(video.CurrentTime)]);
    drawnow;
end
```

Zadanie 4.4 Zmodyfikuj kod uzyskany w zadaniu 4.3, aby zwizualizować przepływ optyczny w każdej klatce, używając omówionej metody. Utwórz trzy sekwencje wideo, które pokazują śledzenie obiektów, używając podejścia progowego, i zwizualizuj ruch,

używając pola wektorowego

Śledzenie przy użyciu detektora narożników Harrisa

Śledzenie może być przeprowadzane przy użyciu lokalnych cech, takich jak te obliczane przez detektor narożników Harrisa. Każdy z narożników jest śledzony niezależnie, dlatego ten tracker może być stosowany do śledzenia zarówno sztywnych, jak i odkształcalnych obiektów. W laboratorium będziemy używać detektora naroży dostarczonego w MATLAB-ie – funkcji **`detectHarrisFeatures()`**.

Aby śledzić tylko wybrany zestaw punktów cech, najpierw zainicjuj tracker punktów w swoim programie. Oto przykład inicjalizacji, pamiętaj, że ustawienia mogą być inne dla Twojego przykładu wideo.

```
tracker = vision.PointTracker('MaxBidirectionalError', 2, ...
'NumPyramidLevels', 3, 'BlockSize', [51 51]);
```

Następnie należy odczytać pierwszą klatkę, przekonwertować ją na obraz w skali szarości i wykryć narożniki za pomocą jednej z metod poznanych w poprzednim ćwiczeniu, np. detektora Harrisa.

```
prevframe = read(video,1);
prevFrame = rgb2gray(frame);

corners = detectHarrisFeatures(prevFrame, 'MinQuality',0.001);
initialCorners = corners.Location; % Get the corners coordinates
```

Możesz wybrać najlepsze cechy do śledzenia, jak pokazano w poprzednim ćwiczeniu laboratoryjnym. Następnie musisz ustawić początkowy punkt śledzenia trackera na te, które znalazłeś za pomocą detektora narożników.

```
initialize(tracker, points.Location, prevFrame);
```

Następnie w pętli przetwarzasz klatki sekwencji wideo jedna po drugiej, tak jak w przypadku wszystkich poprzednich metod śledzenia. Aby śledzić narożniki wykryte w poprzednim kroku, użyj następującego kodu:

```
[points, isFound] = step(tracker, currFrame);
```

Wyniki śledzenia można wyświetlić za pomocą prostego kodu:

```
    imshow(currFrame);
    hold on;
    plot(points(:, 1), points(:, 2), 'go');
    hold off;
    drawnow;
```

Oto przykład kodu śledzenia przy użyciu wyłącznie punktów cech. Należy zauważyć, że istnieje fragment kodu, który sprawdza, czy wykryte cechy nie znajdują się poza granicami obrazu. Zapobiega to utracie punktów, gdy obiekt znajduje się na granicy. Jednak jeśli w Twoim filmie obiekt nie dotyka granicy podczas ruchu, ta część kodu nie jest konieczna.

```
clc
clear all
close all
% Load the video frames
vid = VideoReader('bouncing_ball.mp4');
numFrames = vid.NumberOfFrames;
height = vid.Height;
width = vid.Width;

% Initialize the point tracker
tracker = vision.PointTracker('MaxBidirectionalError', 2, ...
'NumPyramidLevels', 3, 'BlockSize', [51 51]);

% Read the first frame
prevFrame = rgb2gray(readFrame(vid));

% Detect good features to track in the first frame
points = detectHarrisFeatures(prevFrame, 'MinQuality', 0.001);

% Initialize the point tracker with the detected points
initialize(tracker, points.Location, prevFrame);

% Create a figure to display the results
figure;

% Loop through the frames
for i = 2:numFrames
    % Read the current frame
    currFrame = rgb2gray(readFrame(vid));

    % Track the points using the point tracker
    [points, isFound] = step(tracker, currFrame);

    % Boundary checking
    for j = 1:size(points, 1)
        if points(j, 1) < 1 || points(j, 1) > width || points(j, 2) < 1 || points(j,
2) > height
            isFound(j) = false;
        end
    end

    % Remove points that are outside the image boundaries
    points = points(isFound, :);

    % Display the results
    imshow(currFrame);
    hold on;
    plot(points(:, 1), points(:, 2), 'go');
    hold off;
    drawnow;

    % Update the previous frame
    prevFrame = currFrame;
end
```

Zadanie 4.5 Napisz program, który wykonuje rzadką estymację przepływu optycznego. Program powinien działać zarówno z jednym, jak i wieloma obiektami reprezentowanymi przez ich lokalne cechy. Napisz funkcję pomocniczą do wizualizacji trajektorii i prędkości punktów. Sprawdź funkcję na szczególnych przykładach, np. obiekt, który podlega translacji, powinien mieć taką samą prędkość dla wszystkich wybranych lokalnych cech, cechy obracających się obiektów powinny zachowywać się zgodnie z kinematyką ruchu obrotowego itp.

Innym podejściem do śledzenia cech jest użycie deskryptorów cech, które zostały krótko omówione w poprzednim ćwiczeniu laboratoryjnym. To podejście używa niezmiennych deskryptorów cech do śledzenia fragmentów obrazu, które mogą podlegać liniowym transformacjom. Jako przykład zaimplementujemy deskryptor cech SURF.

Zadanie zaczyna się od zainicjowania obiektu videoReader i odczytania pierwszej klatki sekwencji, zmieniając ją na skalę szarości.

Następnie należy wykryć cechy i deskryptory SURF w pierwszej ramce, korzystając z odpowiednich parametrów, tak jak omówiono w poprzednich instrukcjach laboratoryjnych.

```
% Detect SURF features in the first frame
surfPoints = detectSURFFeatures(grayFrame);

% Extract SURF descriptors
[features, validPoints] = extractFeatures(grayFrame, surfPoints);
```

Funkcje te można zwizualizować w następujący sposób

```
figure;
imshow(grayFrame);
hold on;
plot(validPoints.selectStrongest(100), 'showOrientation', true);
title('SURF Key Points in First Frame');
hold off;
```

Następnie w pętli musisz wykryć cechy w następujących po sobie klatkach sekwencji. Ponieważ cechy są wykrywane niezależnie w każdej klatce, muszą być dopasowane między dwiema kolejnymi klatkami. Kod tego podejścia do śledzenia cech:

```
clear all
clc
close all

video = VideoReader('dvdscreensaver.mp4');

frame = read(video,1);
grayFrame = rgb2gray(frame); % Convert to grayscale

% Detect SURF features in the first frame
surfPoints = detectSURFFeatures(grayFrame);

% Extract SURF descriptors
[features, validPoints] = extractFeatures(grayFrame, surfPoints);

figure;
imshow(grayFrame);
hold on;
plot(validPoints.selectStrongest(100), 'showOrientation', true); hold on
title('SURF Key Points in First Frame');

index = 1;

% Track the features in subsequent frames

figure,
 h = imshow(grayFrame); hold on
 h1 = plot(0,0,'*');
 title('Tracking objects');

while hasFrame(video)
    % Read the next frame
    frame = readFrame(video);
    grayFrame = rgb2gray(frame); % Convert to grayscale

    % Detect SURF features in the current frame
    surfPointsCurr = detectSURFFeatures(grayFrame);
```

```
    % Extract SURF descriptors from the current frame
    [featuresCurr, validPointsCurr] = extractFeatures(grayFrame, surfPointsCurr);

    % Match the features from the previous and current frames
    indexPairs = matchFeatures(features, featuresCurr);

    % Get the matched points
    matchedPointsPrev = validPoints(indexPairs(:, 1), :);
    matchedPointsCurr = validPointsCurr(indexPairs(:, 2), :);

    matches{index} = matchedPointsCurr.Location;

    % Display the matched features
   set(h,'CData',grayFrame); hold on

set(h1,'XData',matchedPointsCurr.Location(:,1),'YData',matchedPointsCurr.Location(:,2));
   drawnow;

    % Update the previous points and features for the next frame
    features = featuresCurr;
    validPoints = validPointsCurr;

    index = index + 1;

end
```

Zadanie 4.6 Napisz program, który śledzi obiekt na podstawie jego lokalnych punktów charakterystycznych. Sprawdź niezmienność śledzonych punktów na różne rodzaje ruchu, jakiemu obiekt może podlegać – translacja, obrót, zmiana odległości od kamery, zmiana kąta widzenia. Przetestuj, jak dobrze metoda śledzi lokalne cechy w przypadku tego typu ruchu obiektu.

Zadania dodatkowe

Zadanie 4.7. Napisz program śledzący do śledzenia większej liczby obiektów w obrazie binarnym. Obiekty i tło powinny być takie, aby możliwe było wyodrębnienie obiektów za pomocą operacji progowania. Dla każdego obiektu należy wykreślić trajektorię jego środka ciężkości. Oblicz również wektor prędkości dla każdego obiektu śledzonego.

Zadanie 4.8 Zmodyfikuj swój tracker korelacyjny tak, aby mógł śledzić wiele wzorców jednocześnie w tej samej sekwencji wideo. Narysuj trajektorię każdego śledzonego obiektu w pierwszej klatce sekwencji. Oblicz wektory prędkości dla każdego z obiektów.

Zadanie 4.9 Przeanalizuj kształt odkształcalnego ciała w sekwencji wideo. Podziel obraz ciała na okna wzorców i śledź każde z okien za pomocą znormalizowanego współczynnika korelacji. Aby uprościć obliczenia, możesz ograniczyć obszar wyszukiwania dla każdego szablonu. Narysuj kształt ciała w danej klatce na wykresie.

Zadanie 4.10 Napisz program, który przeprowadza śledzenie lokalnych cech obiektu odkształcalnego. Na podstawie wyników dokonuje wizualizacji zmiany kształtu obiektu w całej ramce sekwencji. Jak takie podejście można zastosować do problemu klasyfikacji?

Zadanie 4.11. Napisz program , który klasyfikuje obiekty na podstawie ich zachowania w sekwencji wideo. Wektor cech może zawierać cechy związane z geometrią obiektów, jak również cechy opisujące ich kinematykę, np. kierunek ruchu lub prędkość