ЕЛЕКТРОНІКА

УДК 004.85
DOI https://doi.org/10.32782/2663-5941/2022.6/47

Переверзєв О.А.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Трапезон К.О.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

ДОСЛІДЖЕННЯ ПРОГРАМНИХ АЛГОРИТМІВ ВІДСТЕЖЕННЯ РУХУ ОБ'ЄКТІВ В ЕЛЕКТРОННИХ СИСТЕМАХ БЕЗПЕКИ

З розвитком технологій в області комп'ютерного зору алгоритми виявлення рухомих об'єктів все частіше використовуються в різних областях науки і техніки, серед яких можна виділити інтерактивне відео, комп'ютерні ігри та симулятори, військові прилади трекінгу, роботизовані системи, тощо. Разом з тим, якщо розглядати в якості рухомого об'єкту людину або кілька людей, то в цьому випадку задача відео спостереження доволі ускладнюється, адже тут необхідно враховувати вимоги до швидкості оброблення зображень об'єктів і до точності розпізнавання останніх на основі підходів ідентифікації унікальних ознак. В роботі розглянуто два програмні алгоритми відстеження об'єктів, на підставі яких можна сформулювати вимоги при створенні електронних систем безпеки. Зокрема, знайдено, що при відстеженні об'єктів в режимі реального часу необхідно враховувати кольорові характеристики зображення, і шум зображення. Також на точність алгоритму впливають різкість зображення та форма об'єкту відстеження. Порівняння двох розглянутих алгоритмів відстеження об'єктів дозволяє стверджувати, що алгоритм CAMShift є простішим, але його стабільність роботи безпосередньо залежить від кольорових характеристик зображення. Насамперед це стосується таких параметрів, як колір, освітлення та шум зображення. Разом з тим, алгоритм на основі побудови оптичного потоку передбачає процедуру вибору та перетворення кольорової схеми зображення і побудова маски зображення для відслідковування особливих крапок при цьому може бути ускладнена, особливо, коли в схемі залучені такі параметри як відтінок, насиченість та яскравість пікселів зображення. Алгоритм CAMShift таких обмежень немає, оскільки в даному випадку вже використовується в основі алгоритму кольорова схема HSV і побудова гістограми області зображення проводиться для кадру після застосування маски та нормалізації значень.

Ключові слова: *система безпеки, зображення, колір, ознака, об'єкт, гістограма, маска, точність.*

Постановка проблеми. В теперішній час, враховуючи стрімкий розвиток інформаційних технологій у світі, використання комп'ютерного обладнання, де для створення зображень, використовуються алгоритми виявлення рухомих об'єктів в реальному часі, вже не є чимось абсолютно новим [1, с. 2383; 2, с. 72558]. Так, подібні алгоритми на сьогодні дуже поширені в різних областях науки і техніки, серед яких варто виділити інтерактивне відео, комп'ютерні ігри та симулятори, військові прилади трекінгу, роботизовані системи, тощо. Разом з тим, якщо розглядати в якості рухомого об'єкту людину або декілька людей в натовпі, то в цьому випадку задача відео спостереження доволі ускладнюється, адже тут висуваються окремі вимоги до швидкості оброблення зображень об'єктів і до точності розпізнавання останніх на основі підходів ідентифікації унікальних ознак. Відео спостереження за об'єктами визначається, як проблема визначення точного місцезнаходження рухомого об'єкта протягом певного часу на відео або на основі аналізу зображень з камери в реальному часі.

З розвитком технологій в області комп'ютерного зору різними авторами були запропоновані різні підходи до вирішення проблеми відстеження людей у реальному часі. Наприклад, у роботах [3, с. 1963; 4, с. 1675] розглядається лазерний далекомір для відстеження людини з метою ідентифікації її ознак – нога людини та окремі рухи

людини. Однак вимірювання цим методом має обмеження у випадку великої відстані між ногою людини та датчиком. Системи з декількома камерами [5, с. 43905; 6, с. 291] також призначені для відстеження людей, але ці підходи обробляються лише в невеликій зоні, де встановлено камери. Система ручних камер для спортсменів розглянута в [6, с. 296], щоб синтезувати стробоскопічне зображення рухомої цілі. Проте в цій реалізації, обчислювальна швидкість і точність не можуть бути оптимальними у випадку складних середовищ.

Насправді, можна виділити декілька проблем, з якими можна зіткнутись під час роботи над алгоритмами відстеження об'єктів. Оклюзія об'єктів у відео є однією з найпоширеніших проблем при аналізі зображень. В системах відео спостереження це стосується ситуації, коли на об'єкт впливає фон або передній план, у якому алгоритм відстеження втрачає відстеження самого об'єкта. Іншими словами, алгоритм заплутується, коли кілька об'єктів наближаються до камери. Це призводить до проблеми, коли спочатку ідентифікований об'єкт знову (помилково) відстежується, як новий об'єкт. Регулювання та контроль чутливості системи до оклюзії дозволяє користувачу визначити, яка особливість об'єкта саме заплутає мережу.

Вплив фону при відстеженні моделей об'єктів може спричинити додаткові проблеми при функціонуванні систем безпеки. Теоретично, чим щільніше фон, тим важче виділити ознаки, за якими можна виявити і відстежити об'єкт. Густозаселений фон створює надлишкову інформацію і це можна вважати за певний шум, який впливає на швидкість та точність відслідковування рухомих об'єктів.

Алгоритм відстеження не є алгоритмом окремого завдання, як це можна описати при класифікації зображень і виявленні об'єктів. Це насправді багатофункціональний алгоритм, який вирішує задачі виявлення об'єктів, їх локалізацію, класифікацію, а також відстеження за ознаками. Такими чином, на основі розглянутих проблем, які виникають при відстеженні рухомих об'єктів, виникає потреба в описі та аналізі методу відстеження, який б за основними особливостями дозволив б нівелювати зазначені обмеження.

Метою статті є розгляд програмних алгоритмів відслідковування рухомих об'єктів в електронних систем безпеки, їх порівняння та як результат знаходження слабких місць при їх подальшій технічній реалізації.

Для досягнення поставленої мети були сформульовані наступні завдання:

– розглянути особливості алгоритму CAMShift та побудувати програмний алгоритм його реалізації в системах відслідковування об'єктів. Додатково необхідно визначити критерії, які впливають на стабільність та точність роботи цього алгоритму;

– визначити поняття особливих крапок зображення при використанні алгоритму Optical flow. Побудувати програмний алгоритм реалізації обчислення оптичного потоку з використанням бібліотеки OpenCV та програмного модуля Pyramid Lucas & Kanade із вдосконаленим алгоритмом Shi-Tomasi.

Виклад основного матеріалу дослідження. Відеоспостереження широко використовується для моніторингу дорожнього руху, а також реалізовано в безпілотних автомобілях та в системах безпеки. За визначенням, відеоспостереження об'єктів – це процес отримання початкового набору характерних ознак об'єктів, створення унікального ідентифікатора для кожної ознаки і подальше слідкування за змінами в ознаках у відео послідовності. Реалізація цього процесу можна розглянути на основі алгоритмічних методів та підходів відеоспостереження об'єктів.

Алгоритм CAMShift (Continuously Adaptive Mean Shift – безперервний адаптивний зсув середнього) є удосконаленою версією класичного алгоритму зсуву середнього [8, с. 357]. Тобто, для області пошуку, де знаходиться об'єкт, формується набір крапок на основі кольорового розподілу пікселів в цій області. Далі розраховується умовний центр цієї області і якщо розташування даного умовного центру співпадає з геометричним центром області, то це означає що сам об'єкт не рухається. Якщо ж не співпадає, то це є сигнал до того, щоб перемістити контури області аналізу і це переміщення вказує на напрям руху об'єкту. Алгоритм CAMShift дозволяє адаптувати розміри області пошуку під розміри об'єкту. Разом з тим, алгоритм Camshift слідує за контурами об'єкта спостереження на основі зворотної проекції гістограми, щоб знайти, які пікселі найкраще відповідають кольору об'єкта, захопленого в початковій досліджуваній області. Для реалізації цього алгоритму використаємо дві бібліотеки мови об'єктно-орієнтованого програмування python – NumPy та OpenCV. За алгоритмом, після підключення бібліотек потрібно перевірити, чи коректно камера системи безпеки записує кадри з об'єктом відстеження. Після цього, встановлюється початок області пошуку – пара-

метр-область roi, який є просто областю екрана, на якій зображено об'єкт, що відстежується:

```
# set up initial coordinates for the tracking window
x, y = 0, 0
# Set up initial size of the tracking window
height, width = 25, 25
track_window = (x,y,width,height)
# set up region of interest (roi)
roi = frame[y:y + height, x:x + width]
```

Змінні x та y є координатами в рамці, яка формується у формі контуру області пошуку і подальшого відстеження об'єкту. У цьому прикладі використовуються початкові координати (0, 0), оскільки їх легко знайти, коли камера починає процес "трансляції" зображення, де є об'єкт. Висота та ширина вказують на розмір roi. Дуже важливо, щоб розмір roi хоча б приблизно відповідав розміру об'єкта, який відстежується.

Змінна track_window зберігає позицію та розмір roi. Незалежно від того, що камера «бачить» у цій області roi, все одно буде зв'язок з діапазоном кольорів, який вона відстежує. Наприклад, якщо червона кулька є в цій початковій області roi, то алгоритм знатиме, що слід відстежувати червоний об'єкт. Через це за даним алгоритмом кадр, який аналізується, має бути в певному визначеному кольоровому просторі. За алгоритмом передбачається використання кольорової системи HSV (hue, saturation, value). Наступним кроком алгоритму CAMShift є створення «маски», за якою відбувається пошук пікселів, які належать об'єкту. Причому, розмір маски враховує контури рамки, яка обмежує область пошуку.

```
# apply mask
hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(hsv_frame, np.array((0, 20, 20)), np.array((180, 250, 250)))
hist_frame = cv2.calcHist([hsv_frame], [0], mask, [180], [0,180])
cv2.normalize(hist_frame, hist_frame, 0, 255, cv2.NORM_MINMAX)
```

Гістограма області зображення обчислюється для кадру після застосування маски та нормалізації значень. Ця інформація використовуються для обчислення кольорів, які алгоритм відстежуватиме в майбутньому.

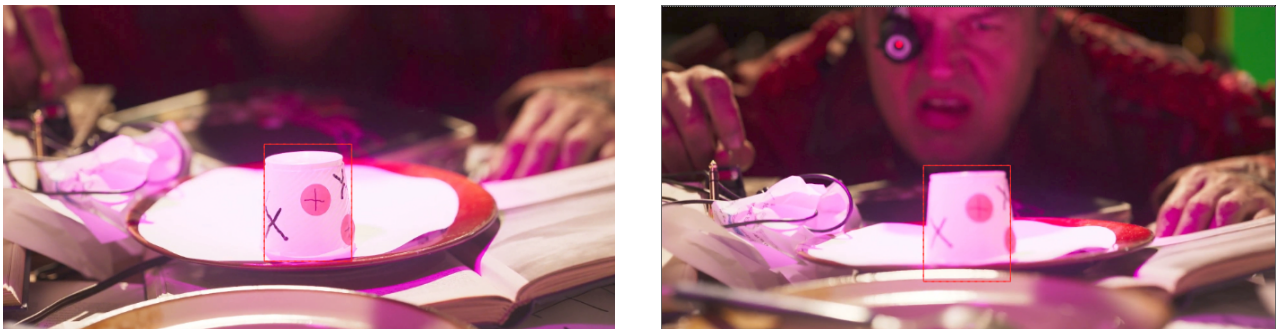
Після запису початкових значень необхідно встановити критерії завершення алгоритму CAMShift. Згадуючи, що середній зсув є ітераційним алгоритмом, який безперервно зсуває крапки, необхідно встановити певний критерій, щоб зупинити його повторення та повернути остаточний набір значень. Встановимо умови, що алгоритм, наприклад, зупиняється якщо або об'єкт не рухається, або пройшло 15 ітерацій:

```
# Setup the termination criteria: 15 iteration 1 pxl movement
term_crit = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 15, 1)
```

На рисунку 1 показано результат роботи алгоритму CAMShift.

З аналізу отриманих результатів на рисунку 1а та 1б, можна відмітити, що на стабільність цього алгоритму впливають колір, освітлення та шум. Тобто, на рисунку 1, б можна помітити, що область відстеження є менш точною, оскільки різкість кадру зменшується. Особливо це проявляється за нижніми рамками області відстеження.

Optical flow – це алгоритм, який використовується для відстеження видимого руху об'єктів у відео. Для відстеження об'єктів використовується поняття особлив их крапок (feature points)



а) **б)**

Рис. 1. Результат роботи алгоритму CAMShift: а) початковий кадр відслідковування об'єкту; б) кінцевий кадр відслідковування об'єкту

зображення [8, с. 365]. Так для набору особливих крапок розраховуються вектори зміщень, або по-іншому вектори руху (motion vectors). Особливі крапки [9, с. 2] повинні мати характерні особливості:

– виразність – тобто крапка в області має виділятись на фоні в своєму околі;

– інваріантність – крапка, її визначення не повинно залежати від афінних перетворень;

– стабільність – визначення особливої крапки має бути стійким до шуму та помилок;

– унікальність – крапка має визначатись глобальною унікальністю, задля покращення помітності при повторі паттернів зображення;

– точність – виявлені крапки повинні точно визначатись, як в оригінальному зображенні, так і у випадку коли зображення проходить крізь процедуру масштабування.

На основі алгоритму Optical flow передбачається розрахунок вектора руху (рис. 2, б) для пікселя, як різниці зміщення об'єкта між двома сусідніми зображеннями. Алгоритм оптичного потоку можна використовувати для відстеження об'єктів, стабілізації та стиснення відео.

Функції оптичного потоку відстежують піксель у послідовних кадрах. Це дозволяє згенерувати шлях руху цього пікселя. Разом з тим, функціонування алгоритму передбачає використання двох припущень – інтенсивність пікселів не змінюється швидко між послідовними кадрами та групи пікселів рухаються разом. Існує багато різних реалізацій розрахунку оптичного потоку і серед них, найбільш поширеним є метод Лукаса-Канаде [10, с. 124]. За цим методом обирається область (вікно) квадратної форми розмірами $n \times n$ пікселів, де центральна крапка, її положення фіксується. Вважається, що всі крапки в області рухаються однаково. Для кожної цієї області проводиться пошук на кращу відповідність у попередньому кадрі на основі розрахунку помилки за значеннями збігу освітленості пікселів. Як тільки аналогічна область знайдена за критерієм мінімуму похибки розходження, шлях між центральною крапкою даної області та попередньої крапки вважається вектором руху. Таким самим чином проводиться аналіз інших областей зображення. Бібліотека OpenCV (Open Source Computer Vision Library) в програмі pyCharm має реалізацію Pyramid Lucas & Kanade із вдосконаленим алгоритмом Shi-Tomasi для обчислення оптичного потоку.

Метод Лукаса-Канаде через бібліотеку OpenCV передбачає, що спочатку необхідно визначити функцію, на основі якою відбудеться запуск відслідковування з використанням оптичного потоку. Крім цього, потрібно провести ініціалізацію об'єкта зображення:

```
def lucas_kanade_method(video_path):
  # Read the video
  cap = cv2.VideoCapture(video_path)
```

Після цього, треба задати параметри для детектора на основі алгоритму Shi-Tomasi та алгоритму оптичного потоку:

```
  # Parameters for ShiTomasi corner detection
     feature_params  =  dict(maxCorners=100,
qualityLevel=0.2, minDistance=5, blockSize=10)
  # Parameters for Lucas Kanade optical flow
  lk_params = dict(
    winSize=(10, 10),
    maxLevel=2,
     criteria=(cv2.TERM_CRITERIA_EPS | cv2.
TERM_CRITERIA_COUNT, 10, 0.03),
```

Відмітимо, що для роботи детектора на основі кутового визначення областей (алгоритм Shi-Tomasi) потрібно задати параметри відслідковування (ознаки особливих крапок), а саме: гра-

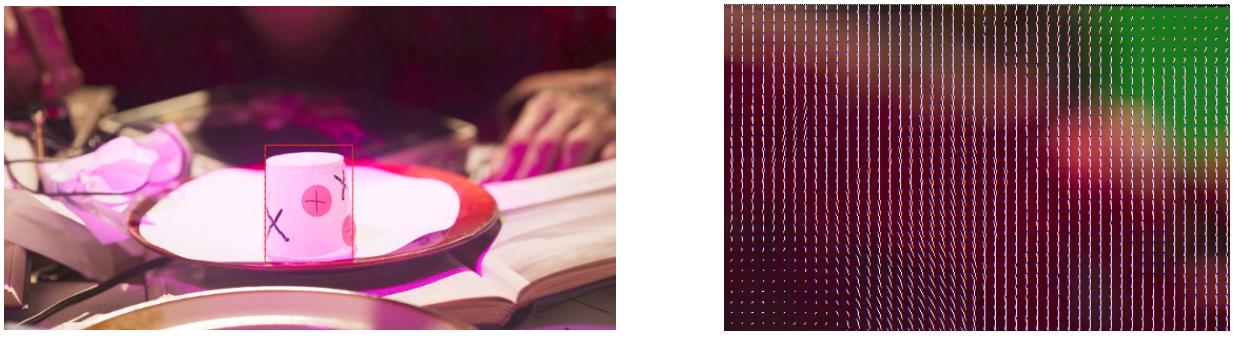


Рис. 2. Приклад використання векторів руху зображення: а) кадр зображення; б) збільшена копія зображення з векторами руху

ничні значення кутів області, параметр якості, мінімальна відстань шляху між особливими крапками в сусідніх кадрах. Наступним кроком визначаються параметри кольорової моделі, адже розрахований оптичний потік візуалізується у вигляді кольорових кривих (рис. 3):

```
   # Create random colors
  color = np.random.randint(0, 255, (100, 3))
  # Take first frame and find corners in it
  ret, first_frame = cap.read()
   old_gray = cv2.cvtColor(first_frame, cv2.
COLOR_BGR2GRAY)
    p0 = cv2.goodFeaturesToTrack(old_gray,
mask=None, **feature_params)
```

Після цього, задається маска області і розраховується для даного кадра зображення оптичний потік.

```
  # Create a mask image
  mask = np.zeros_like(first_frame)
while True:
  # Read new frame
  ret, frame = cap.read()
  if not ret:
    break
  frame_gray = cv2.cvtColor(frame, cv2.COLOR_
BGR2GRAY)
  # Calculate Optical Flow
  p1, st, err = cv2.calcOpticalFlowPyrLK(
      old_gray, frame_gray, p0, None, **lk_
params
  )
  # Select good points
  good_new = p1[st == 1]
  good_old = p0[st == 1]
  # Draw the tracks
   for i, (new, old) in enumerate(zip(good_
new, good_old)):
    a, b = new.ravel()
    c, d = old.ravel()
     mask = cv2.line(mask, (a, b), (c, d),
color[i].tolist(), 2)
     frame = cv2.circle(frame, (a, b), 5,
color[i].tolist(), -1)
```

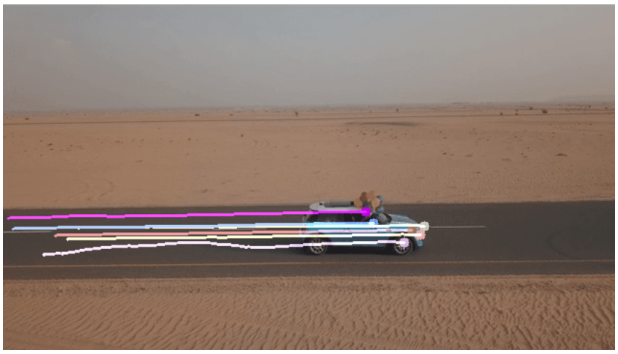


Рис. 3. Результат відслідковування за алгоритмом Optical flow

Висновки. На основі отриманих результатів можна дійти до висновку, що алгоритм оптичного потоку використовує поняття кутового детектора і у випадку зміни форми об'єкту, наприклад зведення його до крапки зображення, не може гарантувати високу точність. Крім цього, на основі наведеного програмного алгоритму оптичного потоку при програмній реалізації мають бути задані параметри детектора і їх числові значення, зокрема розмір вікна спостереження і може виникнути ситуація, що розміри можуть бути недостатніми при створенні векторів руху особливих крапок.

Порівняння двох розглянутих алгоритмів відстеження об'єктів дозволяє стверджувати, що алгоритм CAMShift є простішим, але його стабільність роботи безпосередньо залежить від кольорових характеристик зображення. Насамперед це стосується кольору, освітлення та шуму зображення. Разом з тим, алгоритм на основі побудови оптичного потоку передбачає процедуру вибору та перетворення кольорової схеми зображення і побудова маски зображення для відслідковування особливих крапок при цьому може бути ускладнена, особливо, коли в схемі залучені такі параметри як відтінок, насиченість та яскравість пікселів зображення. Алгоритм CAMShift таких обмежень немає, оскільки в даному випадку вже використовується в основі алгоритму кольорова схема HSV і побудова гістограми області зображення проводиться для кадру після застосування маски та нормалізації значень.

Список літератури:

1. S. Damotharasamy, "Approach to model human appearance based on sparse representation for human tracking in surveillance," IET Image Processing, 14(11), 2383-2394, 2020, doi: 10.1049/iet-ipr.2018.5961

2. K.M. Abughalieh, S.G. Alawneh, "Predicting Pedestrian Intention to Cross the Road," IEEE Access, 8, 72558–72569, 2020, doi: 10.1109/ACCESS.2020.2987777

3. E.J. Jung, et al, ” Development of a laser-rangefinder-based human tracking and control algorithm for a marathoner service robot,” IEEE/ASME transactions on mechatronics, 19(6), 1963-1976, 2014, doi: 10.1109/TMECH.2013.2294180

4. D. Li, et al, “A multitype features method for leg detection in 2-D laser range data, ” IEEE Sensors Journal, 18(4), 1675–1684, 2017, doi: 10.1109/JSEN.2017.2784900

5. V. Carletti, A. Greco, A. Saggese, M. Vento, “Multi-object tracking by flying cameras based on a forward-backward interaction,” IEEE Access, 6, 43905-43919, 2018, doi: 10.1109/ACCESS.2018.2864672

6. F.L. Zhang, et al, “Coherent video generation for multiple hand-held cameras with dynamic foreground,” Computational Visual Media, 6(3), 291–306, 2020, doi: https://doi.org/10.1007/s41095-020-0187-3

7. Інтернет ресурс “CV-Tricks.comLearn Machine Learning, AI & Computer vision” https://cv-tricks.com/object-tracking/quick-guide-mdnet-goturn-rolo/

8. Prateek J. Artificial Intelligence with Python. Packt Publishing, 2017. 446 p.

9. S. Isik, “A Comparative evaluation of well-known feature detectors and descriptors,” International Journal of Applied Mathematics Electronics and Computers, 3(1), 1–6, 2015, doi: 10.18100/ijamec.60004

10. Lucas B., Kanade T. “An Iterative Image Registration Technique with an Application to Stereo Vision,” Proceedings of Imaging Understanding Workshop, 4, 121–130, 1981.

Pereverziev O.A., Trapezon K.O. RESEARCH OF SOFTWARE ALGORITHMS FOR TRACKING THE MOVEMENT OF OBJECTS IN ELECTRONIC SECURITY SYSTEMS

With the development of technologies in the field of computer vision, algorithms for detecting moving objects are increasingly used in various fields of science and technology, among which we can highlight interactive video, computer games and simulators, military tracking devices, robotic systems, etc. At the same time, if we consider a person or several people as a moving object, then in this case the task of video surveillance becomes quite complicated, because here it is necessary to take into account the requirements for the speed of processing images of objects and the accuracy of recognizing the latter based on approaches for identifying unique features . The paper considers two software algorithms for object tracking, based on which requirements can be formulated when creating electronic security systems. In particular, it was found that when tracking objects in real time, it is necessary to take into account the color characteristics of the image and the noise of the image. Also, the accuracy of the algorithm is affected by the sharpness of the image and the shape of the tracking object. A comparison of the two considered object tracking algorithms allows us to state that the CAMShift algorithm is simpler, but its stability of operation directly depends on the color characteristics of the image. First of all, this applies to parameters such as color, lighting and image noise. At the same time, the algorithm based on the construction of optical flow involves the procedure of selecting and transforming the color scheme of the image, and building an image mask for tracking special points can be complicated, especially when such parameters as hue, saturation, and brightness of the image pixels are involved in the scheme. The CAMShift algorithm has no such limitations, since in this case the HSV color scheme is already used as the basis of the algorithm and the histogram of the image area is built for the frame after applying the mask and normalizing the values.

Key words: *security system, image, color, feature, object, histogram, mask, accuracy.*