ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

UDC 621.3 doi: 10.32620/reks.2025.1.09

Vitalii NAUMENKO, Sergiy ABRAMOV, Volodymyr LUKIN

National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine

COMPARATIVE ANALYSIS OF IMAGE HASHING ALGORITHMS FOR VISUAL OBJECT TRACKING

Subject of the research – visual object tracking using various image hashing algorithms for real-time tracking tasks. The goal of this study is to evaluate the tracking success and processing speed of existing and new hashing algorithms for object tracking and to identify the most suitable algorithms to be used under limited computational resources. The objectives of the research include: developing and implementing object tracking based on the aHash, dHash, pHash, mHash, LHash, and LDHash algorithms; comparing the processing speed and accuracy of these methods on the video sequences "OccludedFace2," "David," and "Sylvester"; determining the tracking success rate (TSR) and frames per second (FPS) metrics for each algorithm; analyzing the impact of the search window size, search strategy, and type of hashing on tracking quality, and providing recommendations for their use. The study also explores the trade-off between accuracy and processing speed for each algorithm considering the constraints of limited computational resources. The methods of this study involve testing and evaluating the accuracy and speed of image hashing algorithms on different test video sequences, as well as the use of metrics to determine object similarity using the Hamming distance. The results demonstrate that the aHash and mHash algorithms demonstrate the best accuracy indicators for all hash window sizes, aHash has a higher processing speed, and mHash offers better robustness to changes in lighting and object position. The dHash and pHash algorithms were less effective than the aHash and mHash algorithms due to their sensitivity to changes in scale and rotation. However, perceptual hashing-based methods, such as pHash, are more robust to contrast and blurring. Conclusions. The best hashing algorithms for real-time object-tracking tasks are aHash and mHash. This study underscores the significance of selecting suitable hashing algorithms and search strategies tailored to specific application scenarios and offers possibilities for further optimization.

Keywords: *visual object tracking; single object tracking; image hashing; perceptual hashing.*

Introduction

Visual object tracking is a key research area in computer vision. The proposed method estimates the state of an arbitrary object in a video sequence by knowing only its location in the first frame. This task has several applications, such as autonomous driving, surveillance, augmented reality, and robotics. However, building a universal system for object tracking under real-world conditions using only initial information about their location is an extremely challenging task due to numerous distortions of the observed object, such as edge cases where only part of the object is visible (occlusions), deformations, blurring, illumination changes, and the presence of a highly textured background.

In cases where visual object tracking must operate on an unmanned aerial vehicle, limited computational capability becomes one of the main challenges, along with other specific issues. The inability to accurately identify the object to be tracked from the control panel of an unmanned aerial vehicle is another challenge. FPV (First-Person View) refers to a method of controlling unmanned aerial vehicles (drones) in which the operator receives real-time video feedback from a camera mounted on the drone, viewed through goggles or a screen. This mode requires fast data processing to ensure smooth and accurate video transmission during rapid maneuvers.

This paper proposes an improvement to existing visual object tracking methods based on image hashing. The main advantages of image hashing-based trackers are their simplicity, lack of need for training machine learning models, and high operational speed.

The objective of this research was to perform a comparative analysis of various image hashing algorithms to determine the best methods for visual object tracking based on tracking success rate (TSR) and processing speed (FPS). This study investigates how different hashing techniques, parameter choices, and search strategies influence tracking accuracy and computational efficiency in real-time visual object tracking tasks.

Paper structure

Section 1, "Current state of research in visual object tracking methods," provides an overview of visual object tracking approaches, categorizing them into generative and discriminative types. This section discusses various tracking models and analyzes their advantages and challenges for real-time applications.

Section 2, "Current state of research in perceptual hashing methods," reviews the characteristics, benefits, and limitations of different perceptual hashing techniques, focusing on their accuracy, robustness, and computational efficiency.

Section 3, "Objective and approach," outlines the purpose of the research and details the tasks undertaken to evaluate image hashing algorithms for real-time tracking on computationally limited devices. The main goals of this study include developing tracking methods and analyzing their processing speed and tracking accuracy.

Section 4, "Materials and methods of research," describes the experimental setup, including the video data used and the methods applied to implement the tracking algorithms. The methodology includes an evaluation of each algorithm's tracking accuracy (tracking success rate, TSR) and processing speed (frames per second, FPS) using relevant performance metrics. Specifically, TSR quantitatively measures the overlap between the predicted tracking region and the ground truth, which indicates the tracking accuracy of the algorithms.

Section 5, "Results and discussion," presents the experimental findings, focusing on the tracking success and processing speed of the hashing algorithms. This provides comparative insights, highlighting which algorithms effectively balance speed and accuracy and discussing each method's limitations.

The paper ends with the Conclusions section, summarizing the study's findings and emphasizing the suitability of specific hashing approaches for real-time visual object tracking. The results also suggest future research directions.

1. Current state of research research in visual object tracking methods

A typical tracking system comprises three main components: an appearance model that assesses the probability of the target`s presence at a specific location, a motion model that connects the object`s positions across different time frames, and a search strategy that identifies the most probable location in the current frame. The materials of this paper relate to all three components but mostly focus on the first.

Visual object tracking methods can be broadly divided into two main classes: generative methods and discriminative methods(see Fig.1). Each of these approaches has its own characteristics and is used depending on the tracking accuracy, speed, and performance requirements.

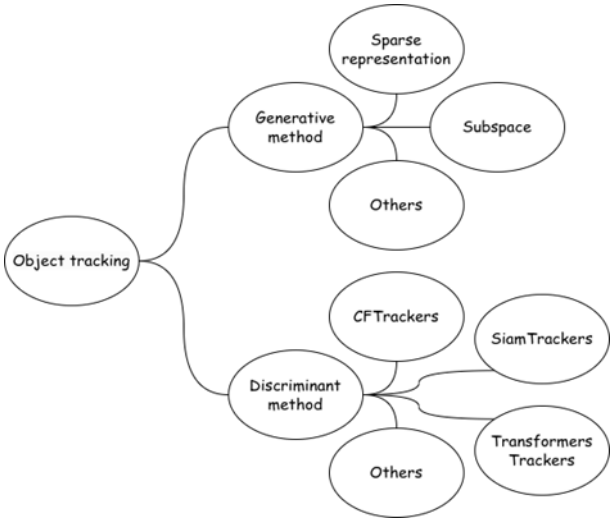

Fig. 1. Tracking methods classification

Generative object-tracking methods are based on creating a model of the object that generates possible variations. A popular approach in this category is sparse representation methods. These methods use a limited number of features to represent an object, focusing on its key elements. For example, the L1 Tracker[1] and IVT (Incremental Visual Tracker)[2] trackers employ this approach to accurately track objects, even when they are partially occluded or temporarily disappear from view.

Another generative method approach deals with subspace methods. These methods use dimensionality reduction techniques, such as Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA), to model changes in the object's appearance. Trackers like TLD (Tracking-Learning-Detection)[3] and MIL (Multiple Instance Learning) Tracker[4] utilize an approach based on separating the object from the background to ensure stable tracking under challenging conditions, such as changes in illumination or object deformation.

Generative methods also include other approaches that use statistical models for object tracking. For example, particle filter-based trackers, such as the PF (Particle Filter) Tracker[5], work effectively under high data noise or complex object dynamics.

In contrast to generative methods, discriminative methods use an approach based on distinguishing the object from the background or other objects. One of the most successful approaches in this category is correlation-filter-based trackers (CFTrackers)[6], [7]. They employ correlation filters to create a discriminative

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

model of the object, thereby allowing for fast and accurate tracking. Examples of such trackers include KCF (Kernelized Correlation Filter)[8] and MOSSE (Minimum Output Sum of Squared Error filter)[9], which deliver excellent performance on real-world videos.

Discriminative methods also include Siamese trackers (SiamTrackers)[10], [11]which use a Siamese neural network architecture to simultaneously learn two different features of an object. This enables high tracking accuracy even under challenging conditions. Examples of such trackers include SiamFC (Siamese Fully Convolutional)[12], SiamRPN (Region Proposal Network)[13], and SiamRPN++[14], which are known for their ability to effectively track objects with minimal latency. LightTrack[15] made a significant step in developing a lightweight network for tracking on mobile devices, while NanoTrack further improved this result and is currently the optimal Siamese neural network-based tracker for CPU, managing to process 20-30 Full HD frames per second on low-power devices like the Raspberry Pi 4.

A recent discriminative method has been connected with transformer-based trackers (Transformers Trackers), which apply transformer architectures to detect long-term dependencies between objects and the background. Among the trackers of this type, TrTr[16] and STARK (Space-Time Attention for Rapid Tracking)[17] are noteworthy. The latter introduces a transformer-based encoder-decoder architecture in a Siamese style: flattened and concatenated search and template feature maps serve as input to the transformer network. STARK also presents a dynamic template update module for efficiently encoding both spatial and temporal information.

In addition, discriminative methods include other approaches, such as support vector machines (SVM)[18] and Adaboost-based learning methods[19]. For example, the STRUCK (Structured SVM Tracker)[20] demonstrates the effective application of these methods under real-world conditions.

Overall, each of these methods has its advantages and disadvantages, and the choice of the appropriate approach depends on the specific task requirements: processing speed, object detection accuracy, or the ability to handle various disturbances and environmental changes.

However, the need to "wrap" an object in a bounding box during tracking initialization remains a challenging task. This task is addressed on relatively powerful computers using segmentation models like SAM[21], [22], where the input can be a single click on the object to provide coordinates, and the output can be a segmentation mask and bounding box, which are well-suited for initializing any tracker. However, the processing time of such object-segmentation models on low-end devices without a GPU does not allow real-time operation on these weaker devices.

In this case, it is more rational to use perceptual hashing-based object trackers[23, 24], which are related to other discriminative trackers and only require the region to be tracked for the start.

2. Current state of research in Perceptual hashing methods

Perceptual hashing has become well-known for its ability to match the content of an image with a template, regardless of data formats (audio and video) and any manipulations it has undergone.

Hashing is applied in many areas, for example, it is a traditional solution for multimedia content authentication[25], [26], detecting unauthorized access[27], accelerating the reconstruction of 3D surfaces from multi-view images[28], and is a popular solution for measuring the similarity between images during image searches[29]. Here, we discuss several hashing methods.

aHash method

The aHash (Average Hash) method converts an image to grayscale, reduces its size to 8x8 pixels, and creates a 64-bit hash matrix based on comparing each pixel to the average brightness value of all pixels in the image. If a pixel's brightness exceeds the average value, the corresponding bit of the hash matrix is set to "1"; otherwise, it is set to "0."

Although aHash is a simple and fast image comparison method, it is overly sensitive to changes in brightness or contrast. For example, operations such as gamma correction or color histogram equalization can alter the average brightness of an image, leading to significant changes in the hash matrix and reducing aHash's robustness to such modifications.

mHash method

The mHash(Median Hash) differs from aHash only in that the median value is used instead of the average value.

Compared to aHash, mHash is slower because it requires sorting; however, it is more robust to changes in scene illumination.

dHash method

The dHash (Difference Hash) method creates an image hash matrix by measuring the brightness differences between adjacent pixels. First, the image was reduced to a size of 9x8 pixels and converted to grayscale. Then, the relative differences between horizontally adjacent pixels are computed: if the pixel to the right is brighter than the current one, the

corresponding bit in the hash matrix is set to "1"; otherwise, it is set to "0." This generates a 64-bit hash matrix that encodes the brightness differences between the pixels. A hash matrix can also be created based on vertically adjacent pixels by resizing the image to 8x9.

The advantage of dHash lies in its robustness to minor brightness and contrast changes because the hash is based on relative differences in brightness rather than absolute values. However, this method is sensitive to image scaling or rotation, which may result in significant changes to the hash matrix.

pHash method

The pHash (Perceptual Hash) method creates an image hash matrix based on its visual features, which allows it to consider the human perception of similarity between images. First, the image was reduced to 32x32 pixels and converted to grayscale. Then, a Discrete Cosine Transform (DCT)[30] is applied to obtain the image in the frequency domain. Only the low-frequency components (the most significant 8x8) are selected, as they best describe the overall structure and visual characteristics of the image. Next, the average value of these low-frequency components is calculated, and each component is compared to the average value: if the component is greater than the average, the corresponding bit in the hash matrix is set to "1"; otherwise, it is set to "0." This generates a 64-bit hash matrix, which is a unique representation of an image's visual properties.

The advantage of pHash is its robustness against many types of image alterations, including scaling, rotation, and brightness and contrast changes, because it is based on frequency analysis, which is less sensitive to these changes. However, this method is more complex and slower to implement than simpler methods such as aHash or dHash.

LHash method

Proposed in [31], LHash (Laplace-based Hash) method creates an image hash by enhancing its edge features using the Laplacian operator. First, the image was reduced to a size of 8x8 pixels and converted to grayscale. The Laplacian transformation is then applied to the image to highlight edges and enhance brightness changes between adjacent pixels. Then, the average value of the resulting image is computed and used as a threshold for building the hash: if a pixel's value is greater than the average, the corresponding bit in the hash matrix is set to "1"; otherwise, it is set to "0." This generates a 64-bit hash that encodes information about the image's edge features.

The advantage of the LHash method lies in its robustness against blurring and minor changes in an object because it enhances the edge details of the image. However, the method is sensitive to changes in scale or image rotation, which may significantly alter the hash.

LdHash method

In addition, as proposed in [31], the LdHash differs from LHash only in that, before applying the Laplacian transformation, the gradient information (the difference between adjacent pixels) is computed.

Compared to LHash, LdHash is somewhat slower because of the additional computations; however, it is more robust to changes in lighting.

3. Objective and Approach

The objective of this study was to evaluate the success rate (tracking accuracy) and processing speed of various image-hashing algorithms for real-time visual object tracking on devices with limited computational resources. This study identifies the most suitable hashing algorithms for use in constrained computational environments, such as devices without a dedicated GPU.

The main research tasks are as follows:

1. Developing and implementing an object-tracking approach based on multiple hashing algorithms.

2. The processing speed and accuracy of these methods were compared across different video sequences.

3. We investigated the impact of search window size and hashing type on tracking quality.

4. Recommendations for selecting and using hashing algorithms in the context of limited computational resources.

The proposed approach compares the calculated object hashes between frames and assesses similarity, thereby evaluating the balance between accuracy and processing speed. This study is particularly relevant for applications in which algorithm robustness against common challenges, such as partial occlusion and dynamic backgrounds, is essential.

4. Materials and methods of the research

Tracking algorithm implementation

This study considers using aHash, mHash, dHash_horizontal, and dHash_vertical (dHash where cells of adjacent columns and adjacent rows are compared, respectively), pHash, and LHash and LdHash.

The tracking algorithm appearance model is implemented by comparing the hash matrices of objects between consecutive frames using the Hamming distance to determine similarity.

After selecting the tracked target, the entire input frame is scanned, and perceptual hashing is computed for

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

each scanning window to measure its similarity to the target using the Hamming distance. The scanning window with the smallest Hamming distance is the most similar to the target and determines the object's location in the frame. The implementation process is described as follows:

1. In the first frame, a rectangle of size A x B is selected, defining the target region r_0, and the size of the scanning window r_w is fixed.

2. In the subsequent frame i of the video sequence, the scanning region r_s is selected, which is an area of the image around r_0 from the previous frame.

3. A search is conducted in region r_s using the window r_w, with n steps, and at each step j, the hash H_w is computed individually. After calculating the Hamming distance $d=H_0-H_w$ between H_0 (the hash from r_0 in the previous frame) and each H_w, all the obtained Hamming distances within this scanning region are compared, and the position (x, y) of the scanning window r_w with the smallest Hamming distance is selected.

4. The window with the smallest Hamming distance $r_w(x, y)$ becomes the new r_0.

5. The cycle ends when the video sequence of length L frames is completed; otherwise, the process is repeated starting from step 2.

Perceptual Hashing based Tracking Algorithm:

Initialize: L, r_0, r_s, r_w, n, i=1, j=1
while: i<L do
while j<n do
d (H_0, H_w) = Hamming (H_0, H_w);
j= j+ 1;
end
$r_w(x,y)$ = arg min d(H_0, H_w);
$r_0(x,y)$ = $r_w(x,y)$;
H_0 = H_w;
end while

Thus, it is possible to implement an object tracking system with the option to select a hash algorithm as desired. Scanning not a full image but a portion of the image r_s around the probable position of the object from the previous frame r_0 significantly reduces the number of computations and improves the average frame per second (FPS) processing speed. In addition, this eliminates the need to account for the probable position of the object from the previous frame $r_0(x,y)$ when searching for the most suitable object position in the current frame $r_w(x,y)$, which positively impacts the accuracy of the search.

Used datasets

Our study used three video sequences: "OccludedFace2", "David", and "Sylvester", taken from [32]. The main reasons for selecting these video sequences are: 1) the absence of significant size changes between the initial and subsequent frames, and 2) the smooth movement of the objects due to the sufficient number of FPS.

Table 1 lists the characteristics of each dataset.

Table 1

Characteristics of test videos

Parameter	Dataset name		
	David	Occluded Face2	Sylvester
Total frame number	770	815	1344
Init height	96	86	46
Init width	82	82	54
Init area	7872	7052	2484
Min bbox_height	54	77	36
Min bbox width	44	73	37
Min bbox area	2376	5767	1332
Max bbox height	122	88	50
Max bbox width	93	82	55
Max bbox area	11346	7216	2750
Average bbox height	95.576	82.787	44.728
Average bbox width	75.583	77.647	47.982
Average bbox area	7371.729	6441.867	2152.64

In the first frame, the search window size is determined (Fig. 2, with the object's location pre-determined by a human, marked in green), and will remain constant.

a b c

Fig. 2. Example frames of David(a), OccludedFace2(b) and Sylvester(c) datasets

Used hash sizes

To compute the hashes H_0 and H_w for windows r_0 and r_w, regardless of their pixel size, it is proposed to adjust the output size of the hash matrix to a power of two, ranging from 2^3 to 2^6(e.g., 8x8, 16x16, 32x32, 64x64). This proposal is motivated by computational efficiency and memory alignment considerations. These sizes optimize memory usage and processing speed on most hardware platforms because modern processors handle binary powers more efficiently due to native architecture alignment.

It is expected, that smaller hash sizes, such as 8x8 and 16x16, should generally yield higher processing speed, but with some trade-off in accuracy. On the other hand, larger hash sizes like 32x32 and 64x64 may tend to increase accuracy but at the cost of lower processing speed.

Tables 2 to 4 present examples of hash matrices, calculated using different hashing methods and with varying hash matrix sizes for the Sylvester, David, and Occluded Face 2 datasets.

Table 2

Init hashed template visualization using different perceptual hashing algorithms and different hash sizes for David dataset

Method	Hash size			
	8	16	32	64
aHash				
dHash horizontal				
dHash vertical				
LdHash				
LHash				
mHash				
pHash				

Used quality metrics

For quantitative performance comparison, it is customary to use the tracking success rate (TSR) and frames per second (FPS) to evaluate the compared tracking algorithms.

TSR is defined as

$$\mathrm{TSR} = \frac{\mathrm{area}\,(\mathrm{ROI_t}\ \mathrm{I}\ \mathrm{ROI_{gt}})}{\mathrm{area}\,(\mathrm{ROI_{gt}})} \in [0,1], \qquad (1)$$

where $\mathrm{ROI_t}$ and $\mathrm{ROI_{gt}}$ denote the tracking region of interest and ground truth region of interest respectively.

The frame rate per second (FPS) is a measure of how many still frames are processed in a single second.

Table 3

Init hashed template visualization with different perceptual hashing algorithms and different size of hash for OccludedFace2 dataset

Method	Hash size			
	8	16	32	64
aHash				
dHash horizontal				
dHash vertical				
LdHash				
LHash				
mHash				
pHash				

Table 4

Init hashed template visualization with different perceptual hashing algorithms and different size of hash for Sylvester dataset

Method	Hash size			
	8	16	32	64
aHash				
dHash horizontal				
dHash vertical				
LdHash				
LHash				
mHash				
pHash				

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

5. Results and discussion

In visual object tracking, multiple search strategies are employed to locate the target in successive frames. These strategies vary in updating the target's position and their trade-offs between computational complexity and tracking accuracy.

The prev_frame search strategy updates the template at each step of the process. Initially, a predefined region is used to locate the object, and at the next step, the best result from the previous step is adopted as the new template H_0, continuing in this manner. This approach results in the accumulation of errors and exhibits the worst performance, as demonstrated by the data in Table 5 and the accompanying graph.

In contrast, the init_template search strategy consistently uses the initial template from the first frame and updates only the scanning region r_s based on the previous best result. This strategy produces better tracking results, as shown in the graph, and aligns with the method in which each H_w is compared only with H_0 from the first frame, while $r_w(x,y)$ is used to update $r_0(x,y)$ without modifying H_0.

The weighted strategy combines elements of both the prev_frame and init_template approaches, with 80% of the result based on the comparison with H_0 from the first frame and 20% based on the comparison with H_0 from the previous frame. This approach performs slightly better, with the optimal percentage ratio determined experimentally.

Figure 3 shows a graphical representation of the comparison between these search strategies, plotting the average Tracking Success Rate (TSR) for each method. Each line on the graph corresponds to a particular search strategy, and TSR determined its color on the experimental datasets.

Ultimately, the best results are obtained using the init_template strategy; thus, only this search strategy is used for further analysis throughout this paper.

Table 5

Hash algorithms with FPS and TSR depending on the size of the hash window and the dataset

Hash name	Hash size	Dataset name						Average value	
		David		OccludedFace2		Sylvester			
		FPS	TSR	FPS	TSR	FPS	TSR	FPS	TSR
aHash	8x8	37.465	0.633	39.91	0.752	95.594	0.605	57.656	0.663
	16x16	41.043	0.608	40.001	0.767	97.86	0.496	59.634	0.624
	32x32	38.385	0.485	38.559	0.767	97.035	0.464	57.993	0.572
	64x64	33.939	0.628	35.097	0.767	89.029	0.456	52.688	0.617
dHash horizontal	8x8	46.164	0.547	47.333	0.325	133.623	0.151	75.707	0.341
	16x16	45.103	0.498	44.233	0.491	103.474	0.299	64.270	0.429
	32x32	43.98	0.587	42.612	0.362	115.588	0.169	67.393	0.297
	64x64	55.261	0.372	39.409	0.653	105.261	0.160	66.644	0.395
dHash vertical	8x8	46.138	0.230	42.869	0.684	108.665	0.503	65.891	0.472
	16x16	43.838	0.359	42.081	0.698	107.757	0.196	64.559	0.418
	32x32	50.029	0.312	41.639	0.725	104.467	0.215	65.378	0.417
	64x64	50.444	0.333	38.821	0.722	97.978	0.180	62.414	0.412
LdHash	8x8	35.884	0.321	36.597	0.424	100.121	0.192	57.534	0.312
	16x16	33.28	0.617	40.746	0.402	98.552	0.248	57.526	0.422
	32x32	31.732	0.391	30.637	0.567	76.352	0.295	46.240	0.418
	64x64	24.574	0.377	23.878	0.506	61.282	0.281	36.578	0.388
LHash	8x8	38.001	0.386	46.625	0.269	95.987	0.104	60.204	0.253
	16x16	40.398	0.653	39.807	0.452	92.829	0.350	57.678	0.485
	32x32	37.108	0.237	35.869	0.414	88.046	0.308	53.674	0.320
	64x64	29.057	0.352	31.073	0.387	83.298	0.338	47.809	0.359
mHash	8x8	32.222	0.640	34.112	0.753	85.356	0.468	50.563	0.620
	16x16	30.872	0.669	33.262	0.746	82.322	0.282	48.819	0.566
	32x32	28.899	0.719	30.364	0.753	86.012	0.339	48.425	0.604
	64x64	24.046	0.756	25.848	0.767	67.41	0.285	39.101	0.603
pHash	8x8	28.823	0.515	30.88	0.769	75.484	0.438	45.062	0.574
	16x16	23.723	0.324	30.79	0.524	77.228	0.279	43.914	0.376
	32x32	17.921	0.353	24.334	0.236	65.576	0.182	35.944	0.257
	64x64	8.571	0.495	8.852	0.574	28.654	0.394	15.359	0.488

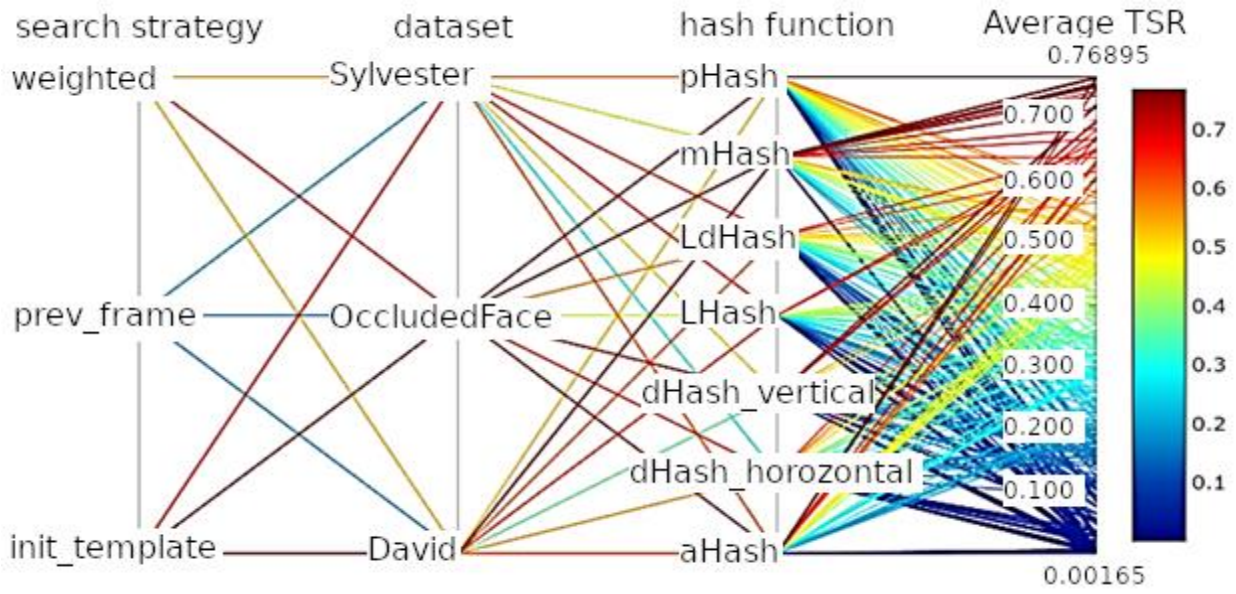

Fig. 3. Search strategy analysis

Table 5 presents the analysis results of the hash algorithms depending on the size of the hash window and the dataset, focusing on tracking accuracy (TSR) and processing speed (FPS) as the primary indicators. This analysis directly confirms the achievement of the research objective and demonstrates how each hashing method balances tracking success rate with computational efficiency.

As shown in Table 5, the best accuracy (average tracking success rate) was achieved with aHash and mHash across all search window sizes, with aHash consistently being faster than mHash. This is due to the higher complexity of finding the median value compared to calculating the average value, although mHash demonstrated higher accuracy on the David dataset, where lighting and object positioning gradually change. dHash_vertical, commonly referred to simply as dHash, generally performs less accurately than aHash and mHash, but is better than dHash_horizontal, except on the David dataset.

The LHash algorithm is faster and more accurate than LdHash and shows comparable speed to aHash and mHash; however, in most cases, it falls short of accuracy. pHash can only work in real-time with a hash window size of 8x8, achieving approximately 30 FPS. However, with each subsequent increase in search window size, the proposed method loses speed significantly while gradually improving accuracy.

Overall, the Table data demonstrate that using a hash window size larger than 64x64 is ineffective. Suppose the image fragment input to the hash algorithm is smaller than the hash window size. In that case, it provides no additional information for comparison while significantly increasing the computation time per step. This is especially true for pHash algorithm, which uses DCT result 4 times larger than the required output hash size.

It is also worth noting that due to the uniformity of the background and the high contrast between the search object and the background, processing the Sylvester video sequence is much faster for all algorithms compared to the David and OccludedFace2 sequences, which are processed at approximately the same speed.

When comparing the accuracy and speed results, aHash exhibited the highest frame processing speed (26–97 FPS) and an accuracy (TSR) range of 0.456–0.767 for a hash matrix size of 64x64, and 0.605–0.752 for a hash matrix size of 8x8. mHash, with a speed range of 17–65 FPS, is somewhat slower than aHash but demonstrates more consistent accuracy (TSR) across different datasets.

When comparing dHash_vertical and dHash_horizontal, the dHash_vertical hashing algorithm demonstrates better performance on the Occluded Face2 and Sylvester datasets in terms of tracking accuracy (TSR = 0.684–0.725 and TSR = 0.180–0.503, respectively) compared to dHash_horizontal (TSR = 0.325–0.653 and TSR = 0.151–0.299, respectively). However, on the David dataset, dHash_horizontal performs better, with a TSR = 0.372–0.587 versus TSR = 0.230–0.359 for dHash_vertical. This suggests that dHash_vertical is generally more suitable for typical tracking tasks, while dHash_horizontal is more effective in scenarios with significant lighting changes.

Conclusions

The comparative analysis of image hashing algorithms for visual object tracking has allowed us to

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

identify the hashing methods that deliver the best tracking success rates (TSR) and efficient processing speed (FPS). Experimental evaluation on the "OccludedFace2," "David," and "Sylvester" datasets demonstrated that the aHash and mHash algorithms consistently provided the highest tracking success rates across various conditions and parameter settings. Specifically, the aHash algorithm achieves excellent processing speed (up to 97 FPS) with robust tracking performance (TSR up to 0.767), whereas the mHash algorithm exhibits slightly lower speed but improved stability under changing lighting and positional variations (TSR up to 0.767).

The results also demonstrate the significant influence of the selected search strategy on the tracking results. The init_template strategy, which uses a fixed initial template for similarity comparisons, outperformed the weighted and prev_frame strategies, achieving the highest TSR values by avoiding cumulative errors. The weighted approach, combining comparisons with both initial and previous frames, demonstrated intermediate effectiveness, whereas the prev_frame strategy demonstrated the lowest accuracy due to error accumulation.

In addition, this study provides insights into the impact of the hash window size on algorithm performance. Smaller hash sizes (8x8, 16x16) yielded faster processing, while larger hash sizes (32x32, 64x64) could improve accuracy under certain conditions, at the expense of reduced speed.

Thus, the primary goal of identifying the best hashing methods and search strategies for visual object tracking was successfully confirmed by the experimental results.

Future research directions. In future work, it may be useful to optimize the search strategy for the best window position by incorporating rotation and scaling. In addition, the automatic selection of the optimal hashing algorithm based on the input video parameters appears to be a promising direction for further improvements.

Contributions of authors: conceptualization, methodology – **Volodymyr Lukin**; formulation of tasks, analysis – **Volodymyr Lukin**; development of model, software, verification, visualization – **Vitalii Naumenko**; analysis of results – **Sergiy Abramov**; writing – original draft preparation, writing – review and editing – **Vitalii Naumenko**.

Conflict of Interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

Financing

This study was conducted without financial support.

Data Availability

The work has no associated data.

Use of Artificial Intelligence

The authors confirm that they did not use artificial intelligence methods while creating the presented work.

All the authors have read and agreed to the published version of this manuscript.

References

1. Bao, C., Wu, Y., Ling, H., & Ji, H. Real time robust L1 tracker using accelerated proximal gradient approach. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1830-1837. DOI: 10.1109/CVPR.2012.6247881.

2. Bai, S., Liu R.,, Su, Z., Zhang, C., & Jin, W. Incremental robust local dictionary learning for visual tracking. *Proc (IEEE Int Conf Multimed Expo)*, 2014, vol. 2014, pp. 1-6. DOI: 10.1109/ICME.2014.6890262.

3. Jia, C., & et al. A Tracking-Learning-Detection (TLD) method with local binary pattern improved. *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2015, pp. 1625-1630. DOI: 10.1109/ROBIO.2015.7419004.

4. Babenko, B., Yang, M.-H., & Belongie, S. Visual tracking with online Multiple Instance Learning. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 983-990. DOI: 10.1109/CVPR.2009.5206737.

5. Cho, J., Jin, S., Pham, X., Jeon, J., Byun, J., & Kang, H. A Real-Time Object Tracking System Using a Particle Filter. *IEEE International Conference on Intelligent Robots and Systems*, 2006, pp. 2822-2827. DOI: 10.1109/IROS.2006.282066.

6. Li, Y., & Zhu, J. A Scale Adaptive Kernel Correlation Filter Tracker with Feature Integration. *Computer Vision - ECCV 2014 Workshops. ECCV 2014. Lecture Notes in Computer Science*, Springer, Cham, 2015, vol. 8926, pp. 254-265. DOI: 10.1007/978-3-319-16181-5_18.

7. Danelljan, M., Häger, G., Khan, F., & Felsberg, M. Learning Spatially Regularized Correlation Filters for Visual Tracking. *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 4310-4318. DOI: 10.1109/ICCV.2015.490.

8. Henriques, J., Caseiro, R., Martins, P., & Batista, J. High-Speed Tracking with Kernelized Correlation Filters. *IEEE Trans Pattern Anal Mach Intell*, 2014, vol. 37. DOI: 10.1109/TPAMI.2014.2345390.

9. Bolme, D., Beveridge, J., Draper, B., & Lui, Y. Visual object tracking using adaptive correlation filters. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp.

2544-2550. DOI: 10.1109/CVPR.2010.5539960.

10. Zhang, Z., & Peng, H. Deeper and Wider Siamese Networks for Real-Time Visual Tracking. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 4586-4595. DOI: 10.1109/CVPR.2019.00472.

11. Xu, Y., Wang, Z., Li, Z., Yuan, Y., & Yu, G. SiamFC++: Towards Robust and Accurate Visual Tracking with Target Estimation Guidelines. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 12549-12556. DOI: 10.1609/aaai.v34i07.6944.

12. Bertinetto, L., Valmadre, J., Henriques, J., Vedaldi, A., & Torr, P. Fully-Convolutional Siamese Networks for Object Tracking. *Computer Vision – ECCV 2016 Workshops. ECCV 2016. Lecture Notes in Computer Science,* Springer, Cham, 2016, vol. 9914, pp. 850-865. DOI: 10.1007/978-3-319-48881-3_56.

13. Li, B., Yan, J., Wu, W., Zheng, Z., & Hu, X. High Performance Visual Tracking with Siamese Region Proposal Network. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition,* Salt Lake City, UT, USA, 2018, pp. 8971-8980. DOI: 10.1109/CVPR.2018.00935.

14. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., & Yan, J. SiamRPN++: Evolution of Siamese Visual Tracking with Very Deep Networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* Long Beach, CA, USA, 2019. DOI: 10.1109/CVPR.2019.00441.

15. Yan, B., Peng, H., Wu, K., Wang, D., Fu, J., & Lu, H. LightTrack: Finding Lightweight Neural Networks for Object Tracking via One-Shot Architecture Search. *arXiv:2104.14545*, 2021. DOI: 10.48550/arXiv.2104.14545.

16. Zhao, M., Okada, K., & Inaba, M. TrTr: Visual Tracking with Transformer. *arXiv:2105.03817*, 2021, DOI: 10.48550/arXiv.2105.03817.

17. Yan, B., Peng, H., Fu, J., Wang, D., & Lu, H. Learning Spatio-Temporal Transformer for Visual Tracking. *arXiv:2103.17154*, 2021. DOI: 10.48550/arXiv.2103.17154.

18. Evgeniou, T., & Pontil, M. Support Vector Machines: Theory and Applications. *Machine Learning and Its Applications. ACAI 1999. Lecture Notes in Computer Science,* Springer, Berlin, Heidelberg, 2001, vol. 2049, pp. 249-257. DOI: 10.1007/3-540-44673-7_12.

19. Yi, C. Target Tracking Feature Selection Algorithm Based on Adaboost. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 2014, vol. 12. Available at: https://ijeecs.iaescore.com/index.php/IJEECS/article/view/3056. (accessed Aug. 8 2024).

20. Hare, S., Saffari, A., & Torr, P. H. S. Struck: Structured output tracking with kernels. *2011 International Conference on Computer Vision,* Barcelona, Spain, 2011, pp. 263-270. DOI: 10.1109/ICCV.2011.6126251.

21. Kirillov, A., & et al. Segment Anything. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, Paris, France, 2023, pp. 3992-4003. DOI: 10.1109/ICCV51070.2023.00371.

22. Ravi, N., & et al. SAM 2: Segment Anything in Images and Videos. *arXiv.2408.00714*, 2024. DOI: 10.48550/arXiv.2408.00714.

23. Fei, M., Li, J., & Liu, H. Visual tracking based on improved foreground detection and perceptual hashing. *Neurocomputing*, 2015, vol. 152, pp. 413-428. DOI: 10.1016/j.neucom.2014.09.060.

24. Fei, M., Ju, Z., Zhen, X., & Li, J. Real-time visual tracking based on improved perceptual hashing. *Multimed Tools Appl*, 2017, vol. 76, pp. 4617-4634. DOI: 10.1007/s11042-016-3723-5.

25. Chen, N., Xiao, H.-D., & Wan, W. Audio hash function based on non-negative matrix factorisation of mel-frequency cepstral coefficients. *IET Information Security*, 2011, vol. 5, iss. 1, pp. 19-25. DOI: 10.1049/iet-ifs.2010.0097.

26. Chen, N., & Xiao, H. Perceptual audio hashing algorithm based on Zernike moment and maximum-likelihood watermark detection. *Digit Signal Process*, 2013, vol. 23, iss. 4, pp. 1216-1227. DOI: 10.1016/j.dsp.2013.01.012.

27. Yang, B., Gu, F., & Niu, X. Block Mean Value Based Image Perceptual Hashing. *2006 International Conference on Intelligent Information Hiding and Multimedia,* Pasadena, CA, USA, 2006, pp. 167-172. DOI: 10.1109/IIH-MSP.2006.265125.

28. Deng, Z., Xiao, H., Lang, Y., Feng, H., & Zhang, J. Multi-scale hash encoding based neural geometry representation. *Comput Vis Media (Beijing)*, 2024, vol. 10, iss. 3, pp. 453-470. DOI: 10.1007/s41095-023-0340-x.

29. Xuan, Z., Wu, D., Zhang, W., Su, Q., Li, B., & Wang, W. Central similarity consistency hashing for asymmetric image retrieval. *Comput Vis Media (Beijing)*, 2024, vol. 10, no. 4, pp. 725-740. DOI: 10.1007/s41095-024-0428-y.

30. Watson, A. Image Compression Using the Discrete Cosine Transform. *Mathematica Journal*, 1994, vol. 4, iss. 1, pp. 81-88. Available at: http://sites.apam.columbia.edu/courses/ap1601y/Watson_MathJour_94.pdf. (accessed Aug. 8 2024).

31. Fei, M., Li, J., Shao, L., Ju, Z., & Ouyang, G. Robust Visual Tracking Based on Improved Perceptual Hashing for Robot Vision. *Intelligent Robotics and Applications. Lecture Notes in Computer Science,* Springer, Cham, 2015, vol. 9246, pp. 331-340. DOI: 10.1007/978-3-319-22873-0_29.

32. Babenko, B., Yang, M.-H., & Belongie, S. Robust Object Tracking with Online Multiple Instance Learning. *EEE Transactions on Pattern Analysis and Machine Intelligence,* 2011, vol. 33, no. 8, pp. 1619-1632. DOI: 10.1109/TPAMI.2010.226.

Received 25.09.2024, Accepted 17.02.2025

ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ ХЕШУВАННЯ ЗОБРАЖЕНЬ ДЛЯ ВІЗУАЛЬНОГО ВІДСТЕЖЕННЯ ОБ'ЄКТІВ

В. М. Науменко, С. К. Абрамов, В. В. Лукін

Предмет дослідження – візуальне відстеження об'єктів з використанням різних алгоритмів хешування зображень для задачі трекінгу в реальному часі. Мета полягає в успішності відстеження та швидкості обробки існуючих та нових хеш-алгоритмів для трекінгу об'єктів та виявленні найбільш підходящих алгоритмів для використання в умовах обмежених обчислювальних ресурсів. Завдання дослідження включають: розробку та реалізацію трекінгу об'єктів на основі алгоритмів aHash, dHash, pHash, mHash, LHash та LdHash; порівняння швидкості роботи та точності цих методів на відеопослідовностях "OccludedFace2", "David", і "Sylvester"; визначення показників точності відстеження (TSR) та швидкості обробки кадрів (FPS) для кожного з алгоритмів; аналіз впливу розміру вікна пошуку, стратегії пошуку та типу хешування на якість трекінгу та надання рекомендацій щодо їх використання. Дослідження також вивчає компроміси між точністю і швидкістю обробки для кожного алгоритму, беручи до уваги обмеження обмежених обчислювальних ресурсів. Методи дослідження включають тестування та оцінку точності та швидкості роботи алгоритмів хешування зображень на різних тестових відеопослідовностях, а також використання метрик для визначення подібності об'єктів за допомогою відстані Хеммінга. Отримані результати показують, що алгоритми aHash та mHash демонструють найкращі показники точності для всіх розмірів хеш вікна, в той час як aHash має вищу швидкість роботи, а mHash - кращу стійкість до змін освітлення та положення об'єкта. Алгоритми dHash та pHash виявилися менш ефективними у порівнянні з aHash та mHash через їхню чутливість до змін масштабу та поворотів. Проте, методи на основі перцептивного хешування, такі як pHash, продемонстрували кращу стійкість до змін контрасту та розмиття. Висновки. Найкращими алгоритмами хешування для задач відстеження об'єктів у реальному часі є aHash та mHash. Дослідження підкреслює важливість вибору відповідних алгоритмів хешування і стратегій пошуку, адаптованих до конкретних сценаріїв застосування, і пропонує можливості для подальшої оптимізації.

Ключові слова: візуальне відстеження об'єктів; відстеження одного об’єкту; хешування зображень; перцептивне хешування.

Науменко Віталій Миколайович – асп. каф. інформаційно-комунікаційних технологій ім. О. О. Зеленського, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Абрамов Сергій Клавдійович – канд. техн. наук, доц. каф. інформаційно-комунікаційних технологій ім. О. О. Зеленського, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Лукін Володимир Васильович – д-р техн. наук, проф., зав. каф. інформаційно-комунікаційних технологій ім. О. О. Зеленського, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Vitalii Naumenko – PhD Student of the Department of Information-Communication Technologies named after O. O. Zelensky, National Aerospace University “Kharkiv Aviation Institute”, Kharkiv, Ukraine,
e-mail: v.m.naumenko@khai.edu, ORCID: 0009-0005-8426-6635.

Sergiy Abramov – Candidate of Technical Sciences, Associate Professor at the Department of Information-Communication Technologies named after O. O. Zelensky, National Aerospace University "Kharkiv Aviation Institute", Kharkiv, Ukraine,
e-mail: s.abramov@khai.edu, ORCID: 0000-0002-8295-9439.

Vladimir Lukin – Doctor of Technical Science, Professor, Head of the Department of Information-Communication Technologies named after O. O. Zelensky, National Aerospace University “Kharkiv Aviation Institute”, Kharkiv, Ukraine,
e-mail: v.lukin@khai.edu, ORCID: 0000-0002-1443-9685.