Multi-Directional Image Projections with Fixed Resolution for Object Matching

Gábor Kertész, Sándor Szénási, Zoltán Vámossy

John von Neumann Faculty of Informatics
Óbuda University
H-1034, Budapest, Bécsi str. 96/b
{kertesz.gabor, szenasi.sandor, vamossy.zoltan}@nik.uni-obuda.hu

Abstract: The matching of the visual representations of two objects is a very important task in most computer vision applications. In special cases, when all objects look alike and only small differences occur, the difficulty of the task increases. In this paper, a novel method for matching low-quality images of rear-viewed vehicles is proposed, using multi-directional image projection functions. For GPU-accelerated implementations, a data-parallel algorithm is introduced. It is concluded, that the use of multiple directions with a small fixed resolution number is the most efficient, and more precise than similar techniques with smaller projection dimensions.

Keywords: Computer Vision; Image Projections; Radon Transform; Object Matching; Vehicle Matching

1 Introduction

Closed-circuit television cameras – also known as surveillance cameras – are often applied to monitor traffic. Based on the use-cases, several applications of these traffic cameras are known: congestion-detection and accident-detection systems are popular, speed cameras, safety and various enforcement solutions as well [1]. Most of these solutions require the device to be able to identify or track the vehicle, which could be challenging depending on the brightness and weather conditions. Advanced devices have high resolution cameras with infrared LEDs for night vision [2], also PTZ (pan-tilt-zoom) cameras can be used to track object movement.

On distant locations like public roads, highways, bridges and tunnels simple static cameras are placed with non-overlapping fields of view. These camera-networks are mostly used to measure traffic after crossroads, calculate the average speed of vehicles based on the distance between the cameras and the time of observations [3].

Recognizing a vehicle by using the plate number is not always feasible. In tunnels, where natural light is rare and colors are hard to detect, the usage of such high-level devices is not cost-efficient, giving similar low-quality images as other, less expensive cameras.

1.1 Problem definition

The changes of lighting and vehicle movement can cause difficulties when matching the image representations, as well as different camera settings could reflect in error.

In computer vision, there are simple methods to find objects with specific color or shape [4] [5]. Most methods are based on lines or corners, or other keypoint-based descriptors extracted from template images. In the case of noisy and low-quality pictures, low-level techniques can be applied. For example, template matching is a method which is based on the pixel-level comparison of the reference and the template. The matching process calculates the correspondence of the template image with the reference image [6]. If the sizes of images differ, a sliding window containing the template is moved over the reference. Most methods are able to handle the scaling or the rotation of objects, however template matching is very sensitive to these manipulations, although several additions exist to handle these.

A method introduced by Viola and Jones [7] is able to summarize pixel values into image integrals, in order to accelerate processing of template data for training [8]. A similar, pixel-level approach available is to compare image projections of the reference and the template images.

In this paper we introduce a multi-directional projection calculation method, which is then used to match low-quality images of vehicles (Fig. 1). In Section 2 a brief overview of the related works are given, Subsection 2.1 formally defines image projections and signatures formed from them and in Subsection 2.2 the paradigm of multi-directional projection is presented. Subsection 3.1 contains the declaration of our novel method of projecting images to a fixed number of bins, the parallel implementation of the suggestion is in Subsection 3.2. The matching technique of the signatures are analyzed and the comparison of our results is shown in Section 4.

Figure 1
Sample images from our dataset. The vehicles are viewed from behind on these low quality images. Resolution goes from 50×50 to 150×150.

2 Related work

Tracking the color information and transforming these data between camera-based models [9] is an appropriate method when objects are tracked in a system of multiple cameras with non-overlapping fields of view. In other relevant work [10], color correlograms are used to match vehicles in different poses. However, these methods are mainly based on color information and assume that the vehicles could show up in different poses.

A matching based on Support Vector Machine (SVM) classification is presented in [11], where the detected edges of the vehicle pairs are used to train the classifier to detect which are the same and the different objects.

Jelača et al. [12] presented a solution for vehicle matching, where object signatures are calculated from projection profiles, and to tolerate potential faults caused by the changing environment and the movement of the object. These projections are joined together in an appearance model. Our work is motivated by this matching technique. Our goal is to increase the precision by using multiple projection directions, and a fixed vector length for all directions.

2.1 Image projection signature

The detection of the vehicles on the image plane could be done several ways: a pre-trained detector based on Haar-features could be applied [13], or even convolutional neural networks seem to perform well on detecting multiple objects on images [14].

After the region of interest is selected, the area is completed to a square and cropped. Since color data are irrelevant, the objects images are grayscaled, meaning that the information is simplified from a RGB structure to a single intensity value. In the case of 8-bit grayscale images the intensity information of one pixel is stored in one single byte.

Each image can be handled as matrix $\mathbf{I} \in \mathbb{N}^{N \times N}$ where $I_{i,j} = [0 \cdots 255]$ denotes the element of the matrix. The horizontal ($\boldsymbol{\pi}_H$) and vertical projections ($\boldsymbol{\pi}_V$) for a squared $N \times N$ matrix results in vectors with the same length:

$$|\boldsymbol{\pi}_H| = |\boldsymbol{\pi}_V| = N. \tag{1}$$

These projections are the averaged sums of the rows and columns of the matrix, normalized to $[0,1]$ by the value of maximal intensity 255:

$$\begin{aligned} \boldsymbol{\pi}_H(i) &= \frac{1}{N \cdot 255} \sum_{j=1}^{N} I_{i,j}, \\ \boldsymbol{\pi}_V(j) &= \frac{1}{N \cdot 255} \sum_{i=1}^{N} I_{i,j}. \end{aligned} \tag{2}$$

The diagonal and antidiagonal projections can be calculated likewise, but it is important to point out that the number of elements for each projected value is not constant. While the length of the diagonal projection vectors are:

$$|\boldsymbol{\pi}_D| = |\boldsymbol{\pi}_A| = 2 \times N - 1, \tag{3}$$

the number of elements in each summarization is based on the distance from the main diagonal:

$$ElemNum(i) = \begin{cases} i & \text{if } i \leq N \\ N - i & \text{otherwise} \end{cases} \tag{4}$$

where i is the index of an element in a diagonal projection, having $i \leq 2 \times N - 1$. The calculation of the diagonal projections $\boldsymbol{\pi}_D$, $\boldsymbol{\pi}_A$ is formalized as:

$$\boldsymbol{\pi}_D(i) = \begin{cases} \frac{1}{ElemNum(i) \cdot 255} \sum_{j=1}^{i} I_{j,N-(i-j)} & \text{if } i \leq N \\ \frac{1}{ElemNum(i) \cdot 255} \sum_{j=1}^{i-N} I_{j+(i-N),j} & \text{otherwise} \end{cases}$$

$$\boldsymbol{\pi}_A(i) = \begin{cases} \frac{1}{ElemNum(i) \cdot 255} \sum_{j=1}^{i} I_{j,(i-j)+1} & \text{if } i \leq N \\ \frac{1}{ElemNum(i) \cdot 255} \sum_{j=1}^{i-N} I_{j+(i-N),N-(j-1)} & \text{otherwise} \end{cases} \tag{5}$$

These vectors together provide a so-called signature of the object.

$$\mathbf{S_4} = (\boldsymbol{\pi}_H, \boldsymbol{\pi}_V, \boldsymbol{\pi}_D, \boldsymbol{\pi}_A), \tag{6}$$

as a 4 dimensional signature, while

$$\mathbf{S_2} = (\boldsymbol{\pi}_H, \boldsymbol{\pi}_V), \tag{7}$$

can also be used as a way simpler 2D signature.

2.2 Multi-directional projections

A logical addition to the signatures above is to calculate the projections of the object from more than four directions. There are few methods that provide a mapping from 2D data to its 1D projections. One of them is the Radon transform [15] [16], a formula which is mostly used with the Computer Tomography (CT) CT, Positron Emission Tomography (PET) or Magneto Resonance (MR) scanners to reconstruct images from the obtained data.

It is interesting to point out, that the Hough transform results in a very similar projection vector. The connection between the two is well-known and discussed thoroughly [17] [18].

The common point of both transforms is that if the input is not circular, the length of the projection vectors differ: each projection length depends on the angle.

The visual representations of the results of the transforms are referred as sinograms, where the projection sums are presented for each direction. A sample of a sinogram representation is on Fig. 2. The denomination sinogram comes from the sinusoid representation of the points.

3 Methodology

Our novel method is based on the idea that adding more dimensions to the signature model could provide more accurate results. Another goal is to create a technique with fixed number of bins, where the number of significant elements of the projection vector does not depend on the angle of projection.

3.1 Image Projections with Fixed Resolution

As demonstrated on Fig. 3a, the projection line is placed to the left side of the image (Fig. 3b), and it is rotated by α degrees around point P, which is in the top left corner. In this case, $\alpha \in [0, \frac{\pi}{2}]$.

While rotating the line, the lowest and highest points of the orthogonal projection divide it into two segments around point P. The length of these can be easily given as

$$\begin{aligned} LL &= \cos(\alpha) \times N \\ HL &= \sin(\alpha) \times N \end{aligned} \tag{8}$$

To create a fixed number of subsegments for all angles, segment $LL+HL$ is divided into S equal parts, where S stands for the number of bins, resulting in $\frac{LL+HL}{S}$ in the resolution. Finally the $(x;y)$ projection position of each pixel is given as

$$\begin{aligned} start &= \sqrt{(x+1)^2+y^2} \times \cos(\arctan(y,x+1)+\alpha) \\ end &= \sqrt{x^2+(y+1)^2} \times \cos(\arctan(y+1,x)+\alpha) \end{aligned} \tag{9}$$

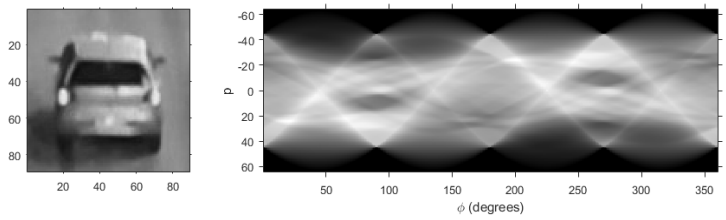
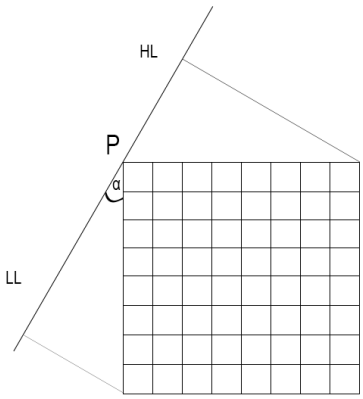
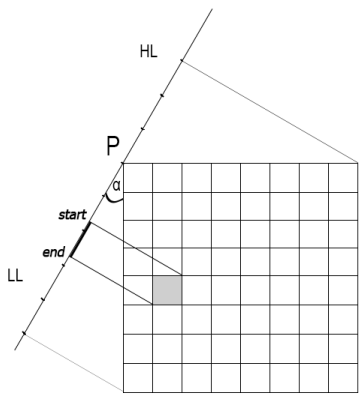


Figure 2
A sample from the dataset and the sinogram of the Radon transform for the same image for $[0;2\pi]$ degrees



(a)
After the calculation of the projection lengths *HL* and *LL* for rotation angle α is done, the projection line is divided into different number of pieces, according to the projection resolution.



(b)
The value of one pixel could affect two (or more) projection bins, according to the resolution.

Figure 3

as seen on Fig. 3b.

Based on S, each pixel is projected into one or more subsegments of the projection line, each pixel should increase the value of all affected bins, proportionately. Algorithm 1 defines the technique in pseudo language, for the better understanding.

The results for each direction are calculated for angles between 0 and $\frac{\pi}{2}$ for an $N \times N$ sized squared matrix I, and collected into R resulting matrix. In practice a step size is used at the iteration of α. After *LL* and *HL* is specified, according to Eq. 8, the resolution of the segment *res* is given by dividing the projection line into S pieces. Each pixel p in image $\mathbf{I}$ is processed by calculating the position of the projection

Algorithm 1 Method to calculate Multi-Directional Projections of an Image

```
procedure MULTIDIRECTIONALPROJECTIONS(I, N, R)
    for α := 0 → π/2 step StepSize do
        LL ← COS(α) * N
        HL ← SIN(α) * N
        res ← (LL + HL)/S
        for all p ∈ I do
            start, end ← POSITIONOF(p.X, p.Y, α)
            RATIONALACCUMULATION(R, res, start, end, p)
        end for
    end for
end procedure
```

using POSITIONOF, which is based on Equation 9, and the correct values of R are increased proportionately, represented as function RATIONALACCUMULATION.

The behavior of RATIONALACCUMULATION procedure can be described as:

- If the number of affected bins is 1, the pixel value is added to the bin entirely
- If the number of affected bins is 2, a ration based on the projection segments is added to each affected bin
- If the number of affected bins is more than 2, fully covered bins are increased by the whole value, and the value of partially affected bins are raised according to the portion of the projection.

The extension of this method to $[0;\pi]$ is done by moving point P to the upper right corner, and rotating HL and LL with it respectively – or the same result could be achieved by rotating the matrix counter-clockwise. Notable, that the method does not need to be extended to a full circle, since the projections are equal on the $[0;\pi]$ and $[2\pi;\pi]$ sections [19].

The resulting matrix of projected values is visualized on Fig. 4. The difference compared to the Radon transform is remarkable: the sinusoids of the picture edges are eliminated.

Although the method provides the necessary results, the performance is questionable. The calculation complexity of the 4D signature is

$$T_1(N) = \mathcal{O}(4 \times N^2) = \mathcal{O}(N^2) \tag{10}$$

which means that the performance is directly proportional to the projection count. The runtime of the proposed method is

$$T_2(N) = \mathcal{O}(StepNumber \times N^2) = \mathcal{O}(N^2) \tag{11}$$

where $StepNumber >> 4$, meaning that the performance of both methods depend on the projection count of the signature. By analyzing the memory cost of the method, we can declare that the original algorithm uses

$$2 \times N + 2 \times (2 \times N - 1) = 6 \times N - 2$$

double-precision floating point numbers to store the results, while our method uses $StepNumber \times S$ doubles, where $StepNumber >> 4$, $N \leq S \leq 2 \times N - 1$.

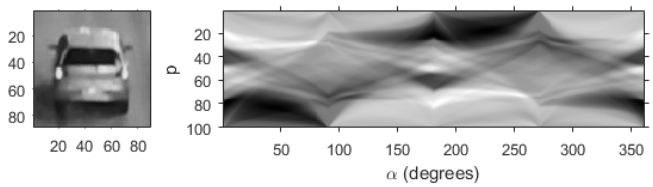


Figure 4
The results of the proposed method, displayed on a sinogram, similar used on Fig. 2.

The value of the total bin number could be defined empirically as N, or constant values could be used. A value less than N causes the compression of the data, resulting in information loss, while greater values result in redundancy.

When using 4D signatures defined in [12], the longest projections are the diagonals, as seen in Eq. 3, while the horizontal and vertical sums are almost half (Eq. 1). The difference of the vector length could cause difficulties storing and handling the data: instead of a matrix an array of different length array should be used.

Software implementations of the Radon transform and Hough transform both use a 2D matrix with the dimension $StepNumber \times (2 \times N - 1)$, meaning that the unused cells of the matrix are filled with empty data. The main advantage of the method presented in this paper is that it does not store any empty values [19], which is useful in data parallel implementations. Since the input matrix is not modified in the iterations, and there is no dependency between calculation steps, multi-level parallelization of the algorithm can be achieved.

3.2 Data parallel solution

The idea of using the architecture of a massive number of processing units in graphical accelerators to solve computationally intense cases created General-Purpose computing on Graphical Processing Units (GPGPU). In practice, these devices perform best on multi-dimensional matrix operations, such as this problem.

When running a calculation on a GPU, the first step must be the transferring of the input data from the memory of the so-called host computer to the device memory. This is the memory transfer time of initializing, which is raised with the time necessary to move the results back from the graphical processor to the memory of the hosting computer. These transfer times should be taken into account when designing the application [20]; it would be wrong to try to access the memory of the computer during the calculation, as it would significantly increase the runtime of the whole procedure.

The code implemented on the GPU is referenced as a compute kernel. The design of the kernel procedures determines the performance of the solution. To achieve the best performance, optimal breakdown of the task is necessary. The aim is to use all multiprocessing units, keeping in mind that access to common variables could cause faults.

The correct usage of the memory architecture [20] of the device is crucial: transfer and access times are present and could have remarkable effects on runtime if designed badly. The main memory of the device – the global memory – could be accessed by the threads, however the access times are better if the less accessible storages, which are assigned to the blocks (shared memory) or the even less accessible registers belonging to the threads themselves (local memory) are used.

A possible solution to achieve a data-parallel solution is to assign singular threads to pixels, and calculate results for every angle, individually (Alg. 2). First, the input matrix is divided into several smaller pieces. These image parts are copied into

Algorithm 2 Kernel procedure to calculate the image projection for multiple directions

procedure MULTIDIRECTION_KERNEL(blk, I_G, N, S, R_G)
  $I_S \leftarrow$ GETBLOCK($I_G, blk.X, blk.Y$)
  $R_S \leftarrow$ *new array*[]
  $disp_S \leftarrow$ *new array*[]
  for $\alpha := 0 \rightarrow \frac{\pi}{2}$ **step** *StepSize* **do**
    $LL \leftarrow \cos(\alpha) * N$
    $HL \leftarrow \sin(\alpha) * N$
    $res \leftarrow (LL + HL)/S$
    $start, end \leftarrow$ POSITIONOF($blk.X, blk.Y + 1, \alpha$)
    $disp_S[\alpha] \leftarrow \lfloor start/res \rfloor$
  end for
  for all $t \in threads$ **do**
    for $\alpha_L := 0 \rightarrow \frac{\pi}{2}$ **step** *StepSize* **do**
      $LL_L \leftarrow \cos(\alpha_L) * N$
      $HL_L \leftarrow \sin(\alpha_L) * N$
      $res_L \leftarrow (LL_L + HL_L)/S$
      $start_L, end_L \leftarrow$ GPOS($blk.X, blk.Y, t.X, t.Y, \alpha_L$)
      RATIONALACCUMULATION($R_S, res_L, start_L, end_L, I_S[t.X]$)
    end for
  end for
  SUMMARIZATION($R_S, disp_S, R_G$)
end procedure

the shared memory. After the transfer, each thread of the block is assigned to each element of the image section, and after the calculation is done the outcomes are positioned and summarized for each angle. The results are first summarized thread safely in the shared memory, then the results of blocks are accumulated in the global memory, from where the final results are transferred back to the host.

The proposed method in Alg. 2 uses all three mentioned levels from the memory architecture: indexes G, S and L indicate that the variables are stored in the global, shared and local memories, respectively. The following list contains comments and explanations for each member of the procedure:

- *blk*: image block identifier
- I_G: image in global memory
- N: image size (width & height)
- S: number of bins
- R_G: result container in global memory
- $blk.X, blk.Y$: coordinates of the block
- GETBLOCK($I_G :, x, y$): returns the block starting at x, y from A matrix
- I_S: image in shared memory
- R_S: results in shared memory
- $disp_S$: precalculated dispositions in block memory

- α: rotation angle
- LL, HL: projection line segment lengths
- *res*: resolution: length of each bin
- POSITIONOF(x, y, α): position of a block with the coordinates of x, y on the projection line for α rotation
- $start, end$: starting and ending of the projection on the projection line
- *threads*: container representing every thread on a block
- t: a single thread
- α_L: rotation of a single pixel, iterated locally
- LL_L, HL_L: projection line segment lengths used by a thread for a specific rotation
- $t.X, t.Y$: position of a pixel
- GPOS(bx, by, x, y, α): returns the projection position of a pixel referred at x, y relatively to the block bx, by, for α rotation
- $start_L, end_L$: starting and ending position of projection, handled locally
- RATIONALACCUMULATION$(R_S, r_L, start, end, v)$: accumulates R_S with v, having r_L resolution from *start* to *end* using thread safe atomic increment
- SUMMARIZATION$(R_S, disp, R_G)$: R_S values are summarized atomically into R_G based on the dispositions $disp_S$

As earlier measurements [21] indicated (Fig. 5), the method performed with promising numbers in point of time and memory efficiency: while the processing times on CPU increase exponentially, the runtime of the GPU-accelerated solution shows linear behaviour.

4 Results

The dataset used for evaluating the method consists of 253 images of 21 different vehicles, labeled manually. The point of view of the detected vehicles are the same, the width and height of the squared images are in average 100 pixels, sizes vary from 48×48 to 150×150, sparsely with a few larger (200×200, 290×290) instances.

On Fig. 6 the results of the original method are visualized: the horizontal, vertical, diagonal and antidiagonal projections are calculated, divided by the number of elements, and normalized to fit to the $[0; 1]$ interval, for both observations. The visualized projection functions show the same behaviour in cases of the horizontal, vertical, diagonal and antidiagonal angles.

4.1 Matching

To calculate the alignment of the functions, the method suggested by Jelača et al. [12] is to align the projection functions globally, and then fine-tune with a local alignment using a method similar to the Iterative Closest Point [22]. There are a number of other methods to measure similarities [23].

Instead of building up the two-step alignment technique, we chose to apply the Pearson correlation coefficient (PCC) with a shifting technique. Since the size of

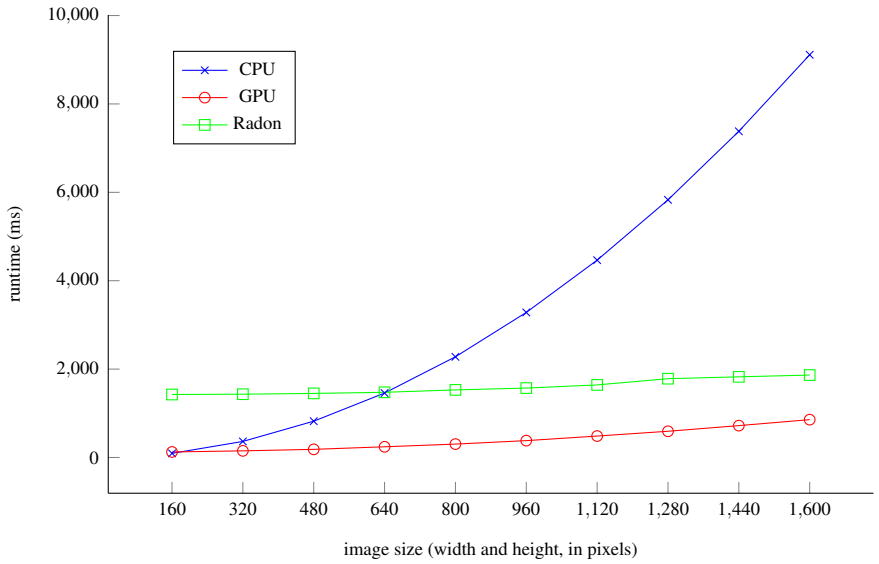


Figure 5
Comparison of runtimes for different image sizes [21]. The horizontal axis displays the width of the squared images in pixels, the vertical axis shows the runtimes in milliseconds. The CPU ad GPU implementation of the presented method was compared with the runtime of Matlab's GPU-accelerated Radon transform.

input images could differ, the projection functions are compared using a shifting window technique: the shorter function is moved over the longer function, and each correlation coefficient is calculated using the PCC formula.

$$\rho(s) = \frac{\mathrm{cov}(\mathbf{x}, \mathbf{y_s})}{\sigma(\mathbf{x})\sigma(\mathbf{y_s})} \tag{12}$$

Basically the $\rho(s)$ correlation coefficients are calculated for each step, where the number of steps is $|\mathbf{y}| - |\mathbf{x}|$, having $|\mathbf{y}| > |\mathbf{x}|$. $\mathbf{y_s}$ stands for the section of $\mathbf{y}$ compared with $\mathbf{x}$ in step s, $\mathrm{cov}()$ means the covariance between the two vectors, and σ indicates the standard deviation.

The range of the values are mapped to $[-1;1]$, which could be easily handled: the higher the coefficient, the better the match. The highest value $\max_s \rho(s)$ is selected as ρ, defining the similarity of $\mathbf{x}$ and $\mathbf{y}$. After all similarity values are calculated for the projections, the result values are filtered with a rectifier, setting all negative values to zero:

$$r(v) = \begin{cases} v & \text{if } v > 0 \\ 0 & \text{otherwise} \end{cases} = \max(0, v) = v^+ \tag{13}$$

The penalization of the negative correlation is necessary because the projection inverses should not be used at all. Negative correlation values mean that the changes

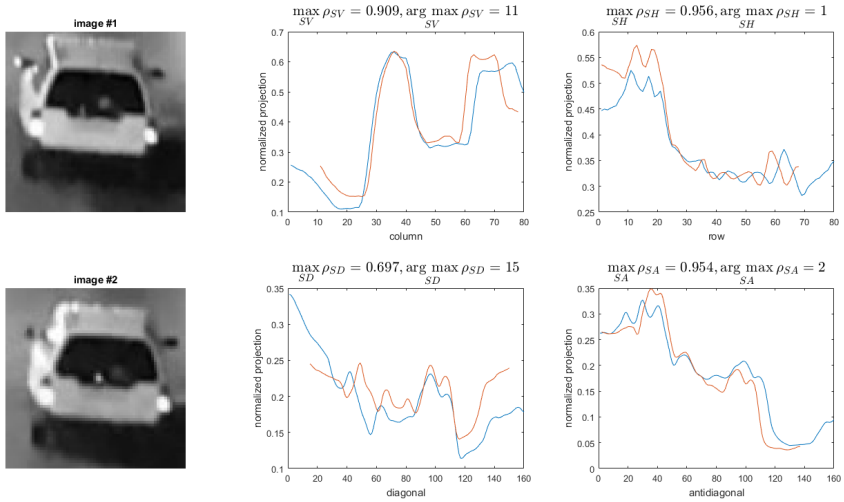


Figure 6
The behavior of the projections of the same vehicle on different observations. In the first column two images of the same object is shown: mind the different sizes and the different lightning conditions, the blink on the side. On the next four diagrams the vertical (top-left), horizontal (top-right), diagonal (bottom-left) and the antidiagonal (bottom-right) projections are visualized, and aligned with the highest calculated correlation.

of one function affects an opposite change on the other function, meaning that the relationship between the two is inverse.

Since each dimension of the data should be equally handled, the suggestion of [12] to use the Euclidean norm is applied here as well. A single value μ is calculated from the 4D signature as

$$\mu = \frac{\sqrt{r(\rho_H)^2 + r(\rho_V)^2 + r(\rho_D)^2 + r(\rho_A)^2}}{2} \tag{14}$$

where 2 is the square root of the dimension number. The measured similarities of the 4D signature are visualized on Fig. 7.

The same method could be applied to evaluate the method presented in this paper: the correlation of each fixed length projection function could be calculated, and the results of the shifted PCC could be united using the Euclidean norm, defined above.

4.2 Evaluation

As already seen on Fig. 7, for the same vehicles the lowest similarity μ is 0.6, while the largest value is 0.98, which is quite convincing. However, when comparing all different vehicles with each other, the results show great spread, represented on Fig. 8.

If a simple classification is done, where it is desired that 50% of the true matches should pass, the line should be drawn to 0.82. However, using this threshold,

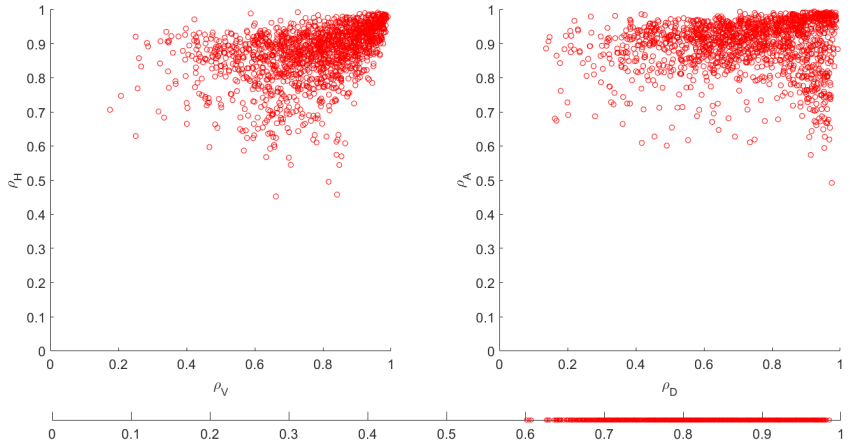


Figure 7
The measured correlations and the calculated similarities in the case of comparing the same instances with different observations. On the left a scatter diagram of the values of horizontal and vertical coefficients ρ_H-ρ_V, and on the right is the same with diagonal and antidiagonal values ρ_A-ρ_D. Below them is the μ similarity value calculated by the Euclidean norm of these, according to Eq. 14.

19.29% of the different vehicles would also pass as false positives, which is way too high.

This is caused by the high variance between the similarity values calculated for different vehicles: while the minimum value is 0.27, the highest calculated similarity is 97.69, with a 10.43% standard deviation. The application of the 2D signature show the same low results: the threshold should be set to $\mu \geq 0.833$, resulting in 22.79% false positives.

When applying the proposed method, several variables could be set: first, the *StepSize* between each projection angle should be set. Our experiments are done with *StepSize* = 5 degrees, $\frac{\pi}{36}$.

The resolution of the projection line is also a significant tradeoff variable. By setting it to N for all images, every projection will be set to N number of bins. If the setting is a constant, for example 100, as the average of the image sizes, will compress less of the data, also some redundancy will come up on smaller images. Notable, that the runtime for the calculation of correlation shortens significantly, as no sliding window is needed, since the vector sizes are equal.

The method with the least compression of the data is the application of $2 \times N - 1$ resolution, which is the exact length of the diagonal. The results for these three different settings are shown in Table 1.

When the proposed method is used with relative bin numbers, and results are matched with the technique described before, the pass-rate and the portion of false positives are closely the same.

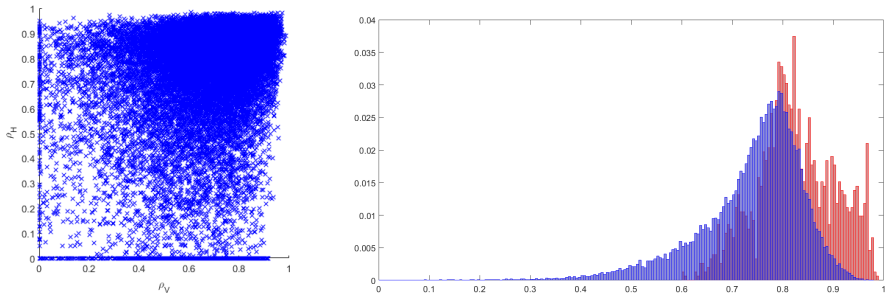


Figure 8
The high rate of false positives using a 4-dimensional signature. On the left side a scatter diagram of measured ρ_H horizontal and ρ_V vertical correlations. On the right is a histogram of the distribution of the similarities calculated for the same (red) and different (blue) objects.

	2D	**4D**	Multi-directional with bin number:					
			N	**2N-1**	**25**	**50**	**100**	**300**
Threshold to pass 50% of true matches	0.833	0.820	0.819	0.819	0.881	0.875	0.873	0.872
Portion of false matches above this	22.79%	19.29%	19.94%	19.84%	5.06%	5.22%	5.24%	5.25%
Threshold to pass 80% of true matches	0.740	0.763	0.769	0.768	0.804	0.795	0.793	0.792
Portion of false matches above this	56.75%	48.85%	48.40%	48.67%	21.26%	21.64%	21.82%	21.85%
Median of the similarity values of true matches	0.833	0.820	0.819	0.819	0.882	0.875	0.873	0.872
Median of the similarity values of false matches	0.760	0.761	0.766	0.766	0.697	0.691	0.689	0.688
Minimum of the similarity on true matches	0.479	0.601	0.573	0.571	0.566	0.557	0.554	0.553
Maximum of the similarity on false matches	0.978	0.976	0.970	0.970	0.968	0.964	0.962	0.962

Table 1
2D, 4D, fixed multi directional with bin number set as N, 2N-1, 25, 50, 100 and 300

However, when using a fixed number as the projection length for all input images, results show that the number of false positives reduces significantly. For example for bin number 25, the limit which passes through 50% of the true matches is drawn at $\mu \geq 0.881$, which is higher than the border set at the two and four dimensional signatures. The portion of false matches is only 5.06%, which is around four times better than the false positives counted using the 2D and 4D signatures.

The difference between the results of the 4D signatures and our method with bin number 25 is visualized on histograms (Fig. 9) generated from the similarity values measured for true and false matches. The high false match rate is caused by the moving window: the best similarity is handled as the final similarity, which leads to high values. A possible solution would be to stretch the different length vectors to the same size, and calculate the correlation correspondingly.

In our research, the optimal resolution number begins at a minimum of 10 bins for all projections (Fig. 10). We understand, that in case of low numbers, the small details are removed, and by the compression a small tolerance to changes is developed.

It might be interesting to present the top false positives and negatives of the method: on Fig. 11a the falsely excluded vehicle pairs with the lowest similarity rate are shown, while the couples of different vehicles with the highest calculated similarity are on Fig. 11b. As Fig. 11a shows, the low similarity values measured for the same vehicles are caused by different poses. By empirically evaluating the calcu-

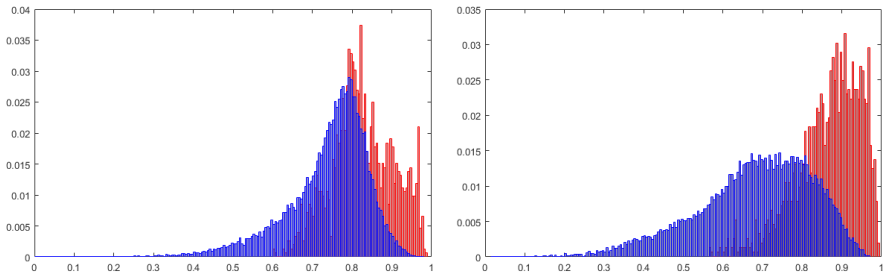


Figure 9
Histograms of the similarities measured using the 4D and the proposed method, red columns show the percentage for the comparison of the same, and blue columns present the calculated values for different objects. The diagram on the left presents the distribution of similarities using the 4D image projection signature, while on the right side, the results of the proposed method is shown, with $\pi/36$ step size, and the projection bin number set to 25.

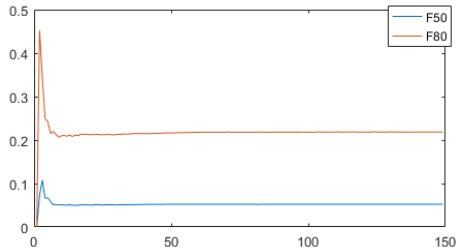


Figure 10
The rate of false positives if the threshold is adjusted to a limit where 50% (F50) or 80% (F80) of true matches should pass, for different number of projection bins.

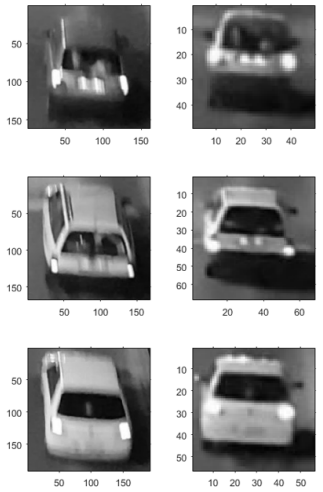
lated similarities, we learned that the correlation of each projection change as the projection angle diverts from the vertical direction. The highest false positives are caused by similar vehicles, blinks, or in few cases the same or similar type of a vehicle is falsely recognized as the same instance.

Conclusions

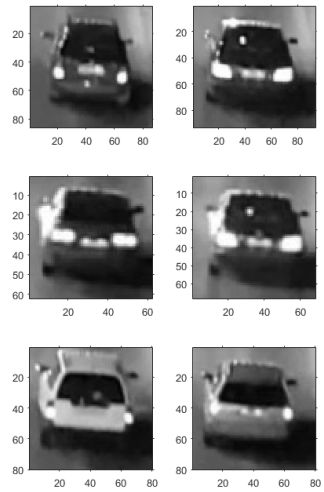
In this paper, we defined a novel method to calculate image projections, similar to the Radon transform. To increase the efficiency of the algorithm, we introduced a data-parallel solution, which could be applied on graphical processors. After evaluating the results, we concluded that in case of a simple Euclidean norm-based matching method the precision of the proposed method exceeds the rates given by previously studied techniques.

As an overall procedure, other possibilities should be examined in each different phase (Fig. 12): the image of the detected vehicle could be preprocessed (noise removal, background subtraction).

Our future plans include the redesign of the matching procedure: a method regarding the angle and the orientation of the vehicle, could lead to more precise predictions



(a)
Images of the same vehicles, detected falsely as negatives. Note the different poses and blur.



(b)
Images of different vehicles, detected as positives. Note the similar blinks on the side of the cars, and that few of them are very much alike, even the same type.

Figure 11

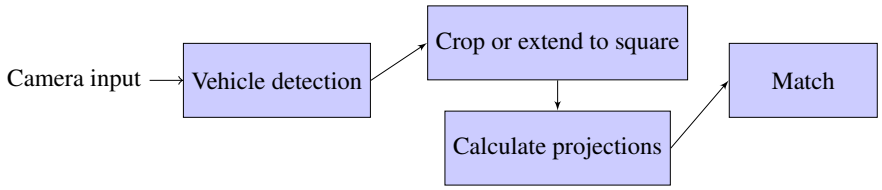


Figure 12
The overall approach as a block diagram: the method described in this paper discusses the last three phases.

for the same or different vehicles. If the current Euclidean norm is to be used, different weights should be rendered to each projections, based on their direction, or more sophisticated classification methods should be applied.

Acknowledgement

Gábor Kertész was supported by UNKP-17-3 New National Excellence Program of the Ministry of Human Capacities. The authors thankfully acknowledge the support of the Doctoral School of Applied Informatics and Applied Mathematics of Obuda University.

Notations and abbrevations

In this study we consequently used the following notations and abbreviations:

$N, start, end, LL, HL, S, \ldots$	scalars
$\mathbf{I}$	image matrix
$\mathbf{S_2}, \mathbf{S_4}$	object signatures
$\boldsymbol{\pi}_H$, $\boldsymbol{\pi}_V$, $\boldsymbol{\pi}_D$, $\boldsymbol{\pi}_A$	projection vectors
σ	standard deviation
$\mathrm{cov}()$	covariance between two vectors
ρ	correlation coefficient
μ	similarity

References

[1] B. Rinner and W. Wolf, "An Introduction to Distributed Smart Cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1565–1575, 2008.

[2] C.-C. Yu, H.-Y. Cheng, and Y.-F. Jian, "Raindrop-Tampered Scene Detection and Traffic Flow Estimation for Nighttime Traffic Surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 1518–1527, 2015.

[3] A. Sanchez, P. D. Suarez, A. Conci, and E. Nunes, "Video-Based Distance Traffic Analysis: Application to Vehicle Tracking and Counting," *Computing in Science and Engg.*, vol. 13, no. 3, pp. 38–45, 2011.

[4] A. E. Abdel-Hakim and A. A. Farag, "Color segmentation using an Eigen color representation," *2005 8th International Conference on Information Fusion*, vol. 2, 2005.

[5] M. Azodinia and A. Hajdu, "A Novel Combinational Relevance Feedback Based Method for Content-based Image Retrieval," *Acta Polytechnica Hungarica*, vol. 13, no. 5, pp. 121–134, 2016.

[6] G. Kertész, S. Szénási, and Z. Vámossy, "Parallelization Methods of the Template Matching Method on Graphics Accelerators," *CINTI 2015 - 16th IEEE International Symposium on Computational Intelligence and Informatics*, pp. 161–164, 2015.

[7] P. Viola and M. Jones, "Robust Real-time Object Detection," in *International Journal of Computer Vision*, 2001.

[8] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. 511–518, IEEE, 2001.

[9] O. Javed, K. Shafique, and M. Shah, "Appearance Modeling for Tracking in Multiple Non-Overlapping Cameras," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, (Washington, DC, USA), pp. 26–33, IEEE Computer Society, 2005.

[10] T. E. Choe, M. W. Lee, and N. Haering, "Traffic Analysis with Low Frame Rate Camera Networks," in *Computer Vision and Pattern Recognition Work-*

shops (CVPRW), 2010 IEEE Computer Society Conference on, pp. 9–16, IEEE, 2010.

[11] Y. Shan, H. S. Sawhney, and R. Kumar, "Vehicle Identification between Non-Overlapping Cameras without Direct Feature Matching," *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1, pp. 378–385 Vol. 1, 2005.

[12] V. Jelača, A. Pižurica, J. O. Niño-Castañeda, A. Frías-Velazquez, and W. Philips, "Vehicle matching in smart camera networks using image projection profiles at multiple instances," *Image and Vision Computing*, vol. 31, pp. 673–685, 2013.

[13] R. Rios-Cabrera, T. Tuytelaars, and L. Van Gool, "Efficient Multi-camera Vehicle Detection, Tracking, and Identification in a Tunnel Surveillance Application," *Comput. Vis. Image Underst.*, vol. 116, no. 6, pp. 742–753, 2012.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'12, (USA), pp. 1097–1105, Curran Associates Inc., 2012.

[15] J. Radon, "Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten," *Berichte über die Verhandlungen der Königlich-Sächsischen Akademie der Wissenschaften zu Leipzig, Mathematisch-Physische Klasse*, pp. 262–277, 1917.

[16] J. Radon, "On the Determination of Functions from Their Integral Values along Certain Manifolds," *Medical Imaging, IEEE Transactions on*, vol. 5, pp. 170–176, 1986.

[17] S. R. Deans, *The Radon Transform and Some of Its Applications*. New York: John Wiley and Sons, 1983.

[18] M. van Ginkel, C. L. Hendriks, and L. van Vliet, "A short introduction to the Radon and Hough transforms and how they relate to each other," 2004.

[19] G. Kertész, S. Szénási, and Z. Vámossy, "Application and properties of the Radon transform for object image matching," *SAMI 2017: IEEE 14th International Symposium on Applied Machine Intelligence and Informatics*, pp. 353–358, 2017.

[20] D. B. Kirk and W. W. Hwu, *Programming Massively Parallel Processors: A Hands-on approach*. Morgan Kaufmann, 2010.

[21] G. Kertész, S. Szénási, and Z. Vámossy, "A Novel Method for Robust Multi-Directional Image Projection Computation," *INES 2016 - 20th IEEE International Conference on Intelligent Engineering Systems*, pp. 239–243, 2016.

[22] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time Dense Surface Mapping and Tracking," in *Proceedings of the 2011 10th*

IEEE International Symposium on Mixed and Augmented Reality, ISMAR '11, (Washington, DC, USA), pp. 127–136, IEEE Computer Society, 2011.

[23] G. Németh, G. Kovács, A. Fazekas, and K. Palágyi, "A Method for Quantitative Comparison of 2D Skeletons," *Acta Polytechnica Hungarica*, vol. 13, no. 7, pp. 123–142, 2016.