IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 14 July 2023, accepted 30 July 2023, date of publication 3 August 2023, date of current version 9 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3301777

RESEARCH ARTICLE

PESAC, the Generalized Framework for RANSAC-Based Methods on SIMD Computing Platforms

EKATERINA O. RYBAKOVA[1,2], ANTON V. TRUSOV[2,3,4], (Member, IEEE), ELENA E. LIMONOVA[2,4], (Member, IEEE), NATALYA S. SKORYUKINA[2,4], KONSTANTIN B. BULATOV[2,4], (Member, IEEE), AND DMITRY P. NIKOLAEV[2,5], (Member, IEEE)

[1]Faculty of Mechanics and Mathematics, Department of Computational Mathematics, Lomonosov Moscow State University, 119992 Moscow, Russia
[2]Smart Engines Service LLC, 117312 Moscow, Russia
[3]Moscow Institute of Physics and Technology, 141701 Dolgoprudny, Russia
[4]Federal Research Center "Computer Science and Control," Russian Academy of Sciences, 119333 Moscow, Russia
[5]Institute for Information Transmission Problems, Russian Academy of Sciences (Kharkevich Institute), 127051 Moscow, Russia

Corresponding author: Ekaterina O. Rybakova (ekaterina.rybakova@math.msu.ru)

This work was supported by a grant from the Ministry of Science and Higher Education of the Russian Federation, internal number 00600/2020/51896, Agreement dated 21.04.2022 under Grant 075-15-2022-319.

ABSTRACT This paper focuses on the computational optimization of RANSAC. We describe the Parallel Efficient Sample Consensus (PESAC) framework that allows efficient utilization of SIMD extensions and provides memory locality due to a special way of storing the input sequence of correspondences and generating a batch of samples per one main loop iteration. It is inspired by the USAC framework and has a block structure capable of implementing most modern RANSAC-based methods. We enhance it with individual blocks of sample and model restrictors that are aimed at the rejection of "bad" samples and model hypothesis before time-consuming model computation and verification blocks. We also provide a detailed description implementing 2D homography estimation problem in PESAC and benchmark the running time on the MIDV-2020 dataset of identity documents. Comparing to naive implementation, we accelerated our framework by 122 times for the document classification task (with a 6% increase in accuracy) and by 18 times for document tracking (with a 46% decrease in tracking failure rate) by using both restrictors and vector processing. This version also outperformed a number of USAC implementations from OpenCV-4.6.0 in runtime and accuracy of estimation (3 times faster, 6% greater accuracy for the classification task, and 2 times faster, 33% lower failure rate for tracking if comparing with USAC_MAGSAC).

INDEX TERMS Homography estimation, identity document, image matching, localization, RANSAC, restrictors, SIMD, tracking, USAC, vectorization.

I. INTRODUCTION

THE problem of model estimation based on data contaminated by noise and outliers often occurs in computer vision. In 1981 Fischler and Bolles proposed a method [1] called RANdom SAmple Consensus (RANSAC), which became one of the most popular techniques for such kind of problems at present time due to its noise tolerance. It is a non-deterministic iterative method intended for estimating with given confidence the parameters of a mathematical model. Algorithms based on RANSAC have been successfully applied to a wide range of computer vision tasks: scene reconstruction from a stereo pair [2], [3], motion detection, and tracking [4], [5], [6], object detection [7], classification [8], segmentation [9], [10], [11], etc.

RANSAC proceeds in the manner of models hypothesizing and verifying. The working principle is as follows: a sample, i.e. a subset of input data with the minimum size

The associate editor coordinating the review of this manuscript and approving it for publication was Abdel-Hamid Soliman.

required for a model computation, is randomly selected, the model parameters are computed from the sample and the model hypothesis is evaluated over the entire input data for computing the support, i.e. a subset of input data consistent with the model. These steps are repeated until the confidence in finding the model with larger support falls below a given threshold. The model with the largest support is returned as the answer. The advantage of RANSAC is the ability to build the correct model, even for data with a high outlier ratio [1]. At the same time, it requires the number of sampling trials polynomially increasing with the outlier ratio, which leads to high computational costs.

Since the publication of the original RANSAC, researchers have suggested various modifications aimed at improving the accuracy [12], [13], [14], [15], [16], [17], speed [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], and robustness [30], [31], [32] of the algorithm.

Since various modifications are of interest for individual practical problems, we consider all of them as a family of RANSAC-based methods.

RANSAC-based methods can be used on various computing platforms. Despite the fact that RANSAC is most often fast enough, in some tasks there is a need to provide a large number of model estimations in a minimum time. The requirement for real-time processing in conditions of low computational capacity arises on the unmanned ground or aerial vehicles (UGVs, UAVs) [34], [35], [36]. For ground robots widely used in farming industry, navigation with Global Positioning System (GPS) can be unreliable because GPS signals alone are weak and unusable in complicated environments like urban canyons [37], so fast visual navigation is required. Air robots are always in motion, so they also need real-time scene analysing. As one more example, video stabilization tasks striving for computationally efficient solutions require RANSAC as the most widely used frame-to-frame analysis method for outlier removal [38]. Moreover, recognition of private and personal data tends to be implemented on edge devices for security reasons and also needs for fast model estimation. For instance, RANSAC is used when choosing the correct document type while extracting data from identification documents [33].

An important platform for such tasks is the Central Processing Units (CPUs) with Single Instruction, Multiple Data (SIMD) architecture. It can process multiple data items in the same way simultaneously using SIMD extensions. Data are placed in vectors of 128-, 256- or 512-bits wide and processed by specialized commands called intrinsics (x86 SSE, ARM NEON, etc.), which makes it possible to accelerate the computations several times. The majority of modern processors have capabilities for vector processing. Another important feature of CPUs is the design of the memory subsystem. Accessing and loading data from RAM can take hundreds of cycles, whereas arithmetic operations are performed tens and hundreds of times faster. To solve this problem, a hierarchical caching system is used. Small data blocks are loaded into caches, which are much faster to access. Therefore, to increase the algorithm efficiency, memory locality should be taken into account. However, RANSAC-based methods do not allow efficient utilization of SIMD extensions generally and do not provide data locality, because of performing rather complex, data-dependent processing of each sample sequentially.

In this paper, we propose the generalized framework for the family of RANSAC-based methods that is suitable for SIMD CPUs and preserves memory locality with no limitations to a specific method. Our Parallel Efficient Sample Consensus (PESAC) framework has a block structure based on the Universal RANSAC (USAC) presented in [39] as the universal modular framework being a synthesis of various RANSAC techniques. USAC accommodates a number of important practical and computational considerations, nonetheless, it does not exploit SIMD vectorization and provides only limited memory locality. We also enhance the PESAC framework with restrictor blocks, which significantly reduce the number of samples passed for model computation and model hypotheses to be verified thus considerably accelerating the runtime. Some checks for samples and models are included in the USAC framework inside modules of sampling minimal subset (Stage 1b) and model generation (Stage 2b) as well. However, we consider such checks to be an important tool for improving algorithm accuracy and time, so include them in PESAC as individual blocks and demonstrate their profit experimentally.

The paper is organized as follows. Section II is an overview of some previous works on improving the computational efficiency of RANSAC. Section III briefly describes the original RANSAC approach. The proposed PESAC framework is discussed in Section IV and Section V is the example-description of how one can use this framework in a 2D homography estimation problem. The results of computational experiments are presented in Section VI. The paper is concluded in Section VII.

II. RELATED WORK

The computational efficiency of RANSAC had been a key topic in many published papers. With the progress in the scientific field and the development of hardware capabilities, the algorithm has acquired an extensive arsenal of optimized variants.

A great number of works had ideas related to high-level algorithm analysis. To improve computational performance researchers have suggested various modifications aimed to optimize single components of RANSAC. The idea of adaptive early termination of sample generation was mentioned in [20] and is regarded as a standard stopping criterion nowadays. Different ideas were proposed with efficient sampling strategies [21], [22], [25], [27], [28], [29] for building more favorable model hypotheses earlier so reducing the number of sample selections and saving the runtime. Optimizations of the model verification, the most time-consuming step of RANSAC, were discussed in [19], [23], [24], and [26]. Their idea was to first perform

verification on a small portion of the data and continue only if some condition was met [23], [24], [26], or to early reject correspondences that are guaranteed to be outliers of the model being considered [19]. Other papers suggested speeding up RANSAC by reducing the number of models to be estimated or verified by holding preliminary tests that check the minimum samples [40], [41], [42], [43], models [44], [45], [46], or both [47], [48], [49] for predefined properties. These methods will be mentioned further in relation to the blocks of the generalized algorithm scheme where they can be implemented.

The integration of the optimized individual components of RANSAC into a unified efficient architecture was presented by Raguram et al. [39] as the USAC framework. It consists of 5 modules: prefiltering; sampling minimal subset; generating minimal sample model(s); model verification, including for nondegeneracy; generating non-minimal sample model. The sampling and minimal sample model generation are strengthened with checks for samples and models by simple tests. Modules can be configured with various approaches proposed in the literature over the years to best address the specific task.

A rather common method for accelerating algorithms is parallelization. Parallel implementations of RANSAC were proposed in [50], [51], [52], [53], [54], and [55]. A comparative review [54] of OpenMP, POSIX Threads, and CUDA variants for fitting a plane in a 3D point cloud established that GPU implementation won in terms of acceleration and acceptable accuracy. A parallel implementation using CUDA for point cloud registration was described in [55]. As a result, the running time was reduced by about 30 times. If there is no GPU, one should alternately use POSIX Threads.

Instruction level optimization is noticeably less common in the literature, nevertheless, it was presented for some image processing tasks. Processor SIMD extensions were used to vectorize the Canny operator [56], homography computation [57] or morphological operations [58]. For RANSAC, some papers combined parallel implementations at level of model generation and evaluation with vectorization of sequential scalar operations [51], [52]. In [51], authors were able to increase the overall performance by a factor of almost 3 on the Cell processor.

A fairly large number of papers considered RANSAC implementations for Field Programmable Gate Arrays (FPGAs), e.g. [3], [4], [5], [59], [60], [61], [62]. In [4] and [5], authors presented UAVs hardware/software co-design implementations to estimate affine transformation for tracking. They notice that most noise introduced into image processing occurs during transmission to the ground station computer, so implementing the onboard solution for micro UAV applications is obviously important [4]. In [5], the hardware implementation was described for model verification, the most time-consuming step. The proposed architecture was capable of processing a video stream at 30 frames per second. The FPGA implementation of RANSAC had been also used for real-time ellipse estimation for circular road sign detection [59] and eye tracking [60]. In [3], a multiprocessor system based on Microblaze microprocessors was proposed using 8-point and 5-point algorithms for the essential matrix computation.

Other papers described the hardware implementation of RANSAC for estimation models of affine and projective transformations [63], [64]. In [64], the authors showed that a projective transformation can be decomposed into four others: two shifts, an affine transformation, and a computationally simpler projective one. Such an approach enables hardware implementation and greatly reduces the required number of bits for fixed-point representation of the transformation coefficients and intermediate variables.

Thus, in a large number of problems reduced to image processing, the use of RANSAC emerges at some point, which indicates its relevance and justifies the need for further optimization.

III. ORIGINAL RANSAC

RANSAC was originally proposed as a robust method for fitting a model that satisfies the original data in the presence of outliers. There are no limitations on the model kind. The method is performed by iterative examining the space of model parameters to maximize (or minimize) some score functional.

Data items can be considered as correspondences between *reference* and *query* points. Each iteration starts with the generation a sample, i.e. data random subset with the minimum size required for a model computation. After the sample is generated, the model parameters θ are computed from it. To answer the question "Is the model consistent with the rest of the input data?", the model hypothesis is verified against the remaining subset. For i-th correspondence, the consistency with the given model is checked using an error functional ρ and some threshold $\mathcal{T}$:

$$\rho\left(e_i^2(\theta)\right) = \begin{cases} 1, & \text{if } e_i^2(\theta) \le \mathcal{T}^2, \quad (\textit{inlier}) \\ 0, & \text{otherwise}, \quad (\textit{outlier}) \end{cases} \tag{1}$$

where $e_i(\theta)$ is the i-th correspondence error for the model with parameters θ. The value of functional $\mathcal{C}$, or score, is obtained by summarizing the values of the error functional:

$$\mathcal{C}(\theta) = \sum_i \rho\left(e_i^2(\theta)\right). \tag{2}$$

These three "generate, compute, and verify" steps form the main loop of RANSAC. After the main loop has iterated, the model hypothesis with the largest $\mathcal{C}(\theta)$ is returned as a solution. Within the above ρ definition from Eq. (1), score $\mathcal{C}(\theta)$ corresponds to the number of inliers for a given model with parameters θ.

To complete the description of the original formulation, it remains to determine the number of main loop iterations. Let $\mathcal{P}$ be the probability of drawing an outlier-free sample. Denoting by ε the probability of selecting an inlier from the input data and by $\mathcal{M}$ the minimum sample size, with no limitations on the sample and assuming it can be degenerate,

we obtain the probability of independently selecting all-inlier sample, $\varepsilon^{\mathcal{M}}$. The probability of selecting no samples clear from outliers in $\mathcal{I}$ iterations is $(1-\varepsilon^{\mathcal{M}})^{\mathcal{I}}$, or $1-\mathcal{P}$. This yields a theoretical lower estimate for the number $\mathcal{I}$ of iterations required to construct with a probability $\mathcal{P}$ the all-inlier sample:

$$\mathcal{I}^* \geq \frac{\log\left(1-\mathcal{P}\right)}{\log\left(1-\varepsilon^{\mathcal{M}}\right)}. \tag{3}$$

For numerous applications, a real-time implementation of RANSAC is desirable. However, the algorithm's computational complexity depending on the number of iterations required is an obstacle to achieving such performance.

IV. A PARALLEL EFFICIENT SAMPLE CONSENSUS FRAMEWORK

This section provides a detailed discussion of the proposed framework for the family of RANSAC-based methods. The PESAC framework has a block structure and consists of *Sampling* and *Sample Restrictor* blocks, *Model Computation* and *Model Restrictor* blocks, blocks of *Model Verification*, *Early Termination*, and *Model Refinement*. The flowchart of the framework is shown in Fig. 1. Blocks highlighted with light-blue background are optional and not included in the original RANSAC approach, as could be seen from Section III. Below, we discuss each component in detail.

A feature of the PESAC is a special way of storing the input data, which is suitable for convenient SIMD vectorization. To maintain memory locality, we process $\mathcal{K} \gg 1$ samples per one main loop iteration. We should note that it can be a disadvantage in the case when the theoretically required number $\mathcal{I}$ of iterations determined by Eq. (3) is comparable to $\mathcal{K}$ and is not a multiple of $\mathcal{K}$. In this case, we can process more samples and hold extra iterations. However, in practice, since the inliers are noisy, processing large enough amount of samples is required in fact.

A. INPUT AND OUTPUT

1) INPUT

In general, RANSAC-based methods take as input a sequence $\mathcal{S}$ of correspondences from reference and query sets of points and some numerical values characterizing these correspondences. Such values can be the probabilities of selecting a given correspondence in a sample [21], [25], [32], [65], [66], or spatial characteristics [27], [67].

The sequence $\mathcal{S}$ can be stored in the memory in different ways. Here, we propose a method that provides locality in memory and the possibility of kindly vector data processing:

$$\mathcal{S} = \{\mathcal{R},\ \mathcal{Q},\ \mathcal{SP},\ \mathcal{EW}\}, \tag{4}$$

where $\mathcal{R} = \{r_1, \ldots, r_{\mathcal{N}}\}$, $\mathcal{Q} = \{q_1, \ldots, q_{\mathcal{N}}\}$, $\mathcal{N}$ is the sequence length and points from the pair (r_i, q_i) must be transformed into each other and form an edge in the matches graph. Arrays $\mathcal{SP}$ and $\mathcal{EW}$ of size $\mathcal{N}$ store probabilities and spatial characteristics of the correspondences. Its usage is optional.

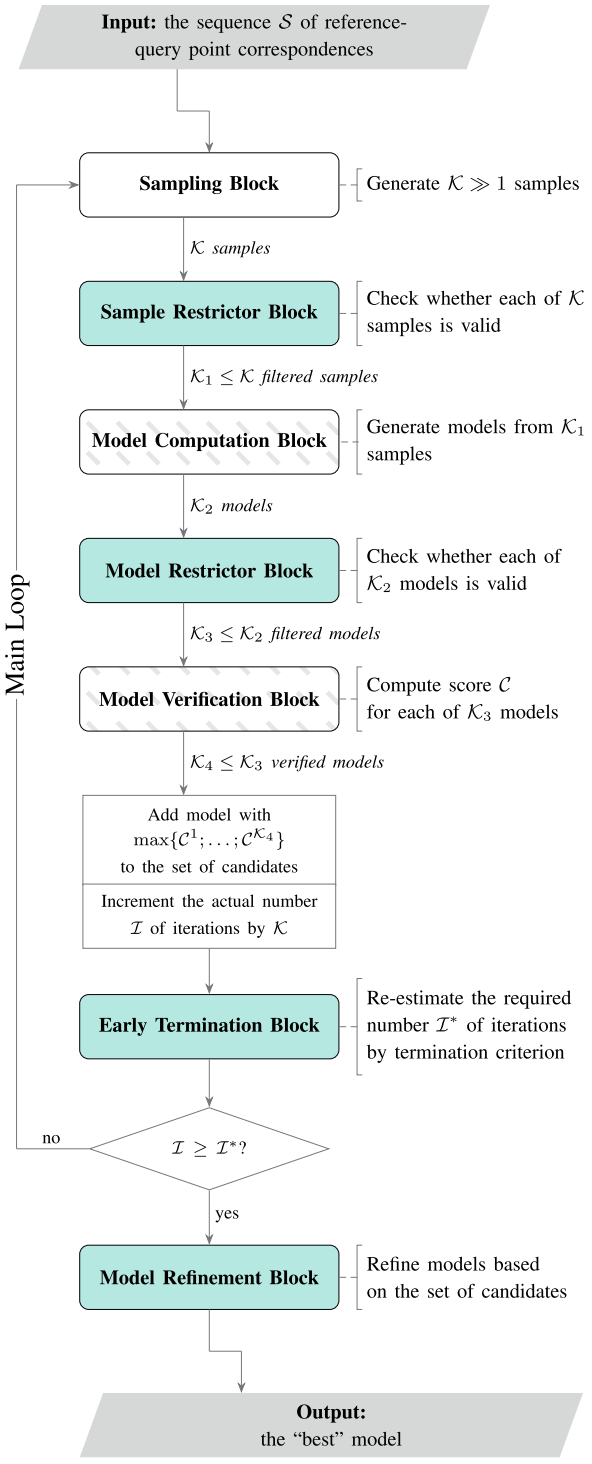


FIGURE 1. Flowchart of the PESAC framework for the family of RANSAC-based methods. Optional blocks not included in the original method formulation (Section III) are colored. In the shaded blocks we use vectorization.

Note that this storage method allows for the points in $\mathcal{R}$ and $\mathcal{Q}$ to be repeated. In addition, points can be of different types (e.g., for the task of camera pose estimation from 2D–3D point correspondences).

2) OUTPUT

The output of RANSAC-based methods is one or more models and their characteristics that can be useful for further applications. Most frequently, these will be the model $\mathcal{H}$ coefficients and score $\mathcal{C}$, but there may also be an inlier mask, points indices used to build the model, etc.

B. SAMPLING BLOCK

The block of *Sampling* takes the sequence $\mathcal{S}$ of correspondences as input and is intended for generating random minimal subsets of matches that will be used for model computation further.

In our notation, the sample is a finite set $\{i_1, \ldots, i_{\mathcal{M}}\}$, $i_j \in \overline{1, \mathcal{N}}$, $j \in \overline{1, \mathcal{M}}$, of indices from the sequence $\mathcal{S}$, where $\mathcal{M}$ is the minimum number of matches required for computing the model hypothesis. Sampling can be of different kinds:

- Exhaustive, when a complete enumeration of all the minimum subsets of matches in a predetermined order is performed; this kind is relevant for a small number of correspondences when one can iterate over all subsets;
- Uniform, when the matches in the sample are obtained by a uniform generating of $\mathcal{M}$ random numbers, i.e. indices of correspondences, from 1 to $\mathcal{N}$; this kind is used when no additional information about the correspondences is available and was also used in the original RANSAC approach [1];
- Non-uniform, when additional characteristics of correspondences are considered for the matches selection to generate more preferred samples earlier; it is the most modern sampling strategy followed by modifications of NAPSAC [22], PROSAC [25], DL-RANSAC [28], P-NAPSAC [29], and other [21], [27].

We notice that the block generates $\mathcal{K}$ samples per one main loop iteration. Using $\mathcal{K} \gg 1$ allows vectorizing random number generation directly in this block and involving SIMD extensions further.

C. SAMPLE RESTRICTOR BLOCK

Sometimes the model estimating can be a costly computation. If the minimum sample is unlikely to be meaningful in the context of the problem being solved, it will hardly produce an accurate model. We can avoid the unnecessary calculation of model parameters by adding some preliminary tests for samples. In order to do this, we include *Sample Restrictor* block in the framework.

The collection of sample restrictors is aimed at rejecting ''bad'' samples with a minimal computational effort from $\mathcal{K}$ given ones, returning $\mathcal{K}_1 < \mathcal{K}$ valid samples. The effective test must examine the relatively simple criterion compared to the model computation and further local/global optimization if used. In the literature, there have been proposed geometrical restrictions imposed on

- Relative order (orientation) of points in a sample; it was discussed for the affine transformation [40], 2D homography [41], [42], and epipolar geometry [47] estimation;
- Area of the shapes formed by the points in a sample: triangles in affine transformation [68], quadrilaterals in 2D homography [43] estimation;
- Distances between points in a sample: abscissa difference in the problem of line fitting [69], Euclidean distances in 2D homography estimation [48], [49]; etc.

D. MODEL COMPUTATION BLOCK

Given a set of $\mathcal{K}_1$ samples, this block builds a set of $\mathcal{K}_2$ models. We notice that $\mathcal{K}_2$ may not be equal to $\mathcal{K}_1$, if some samples did not originate models or, on the contrary, generated more than one (e.g., in a task of computation the fundamental or essential matrices by 7-point or 5-point algorithms [70], [71]). For $\mathcal{K}_1 \geq 1$, we can use SIMD vectorization for the given algorithm of calculation model parameters. The ease of using SIMD is highly dependent on the particular transformation. For example, in Section V we focus on the case of 2D homography since it represents the widest class of linear 2D transformations and is quite computationally intensive. Due to coinciding indexing in $\mathcal{R}$ and $\mathcal{Q}$, it is possible to process t elements of these sequences simultaneously. The value t is the number of values stored in a SIMD register depending on the register size and the type of data used and usually varies from 2 to 16.

E. MODEL RESTRICTOR BLOCK

Model Restrictor allows skipping the costly verification completely for some physically non-meaningful transformations. This block rejects ''bad'', in terms of the task being solved, models from $\mathcal{K}_2$ transformations obtained after model computation and retain $\mathcal{K}_3$ ''good'' ones before the model verification step.

Geometric constraints depend on the specific task. Restrictions have been introduced for the following models:

- Epipolar geometry: oriented version of the epipolar constraint [44], [45], consistency of orientations and distances from epipole to sample points in image pair [47];
- 2D homography: convexity preservation by checking that the intersection point of the diagonals of the unit square is inside the resulting quadrilateral [48], [49] or by checking that an ellipse inside a convex quadrilateral is still an ellipse under a homography [72], cheiral inequality [73]; etc.

In [46], authors proposed Latent RANSAC, where after a hypothesis is generated it is mapped to a latent space as λ-dimensional vector, undergoes a hash procedure, and is only verified if it is similar to any of the previously stored — that might also be regarded as a model check. The paper suggested parameterizations for the space of 2D homographies and 3D rigid motions. The method is particularly interesting because of the ability to evaluate hypotheses in constant time.

Pseudocode 1 Vectorized *Model Verification*

Input: sequence $\mathcal{R} = \{r_1, \ldots, r_{\mathcal{N}}\}$
of reference points from Eq. (4);
sequence $\mathcal{Q} = \{q_1, \ldots, q_{\mathcal{N}}\}$
of query points from Eq. (4);
matrices of transformations
$\{\mathcal{H}^1, \mathcal{H}^2, \ldots, \mathcal{H}^{\mathcal{K}_3}\}$, where $\mathcal{H}^j : \mathcal{R}_4^j \to \mathcal{Q}_4^j$;
integer value t – the amount of correspondences
processed simultaneously.
Output: score values $\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^{\mathcal{K}_3}$,
where $\mathcal{C}^j = \mathcal{C}(\mathcal{H}^j)$.
for k from 1 to $\mathcal{K}_3$ **do**
  $\mathcal{C}^k \leftarrow 0$
  $\mathcal{N}' \leftarrow \lfloor \mathcal{N} / t \rfloor \cdot t$;
  for n from 1 to $\mathcal{N}'$ with the step t **do**
    $e^2_{n+0} \leftarrow \|\mathcal{H}^k(r_{n+0}) - q_{n+0}\|^2$;
    $\vdots$
    $e^2_{n+t-1} \leftarrow \|\mathcal{H}^k(r_{n+t-1}) - q_{n+t-1}\|^2$;
    $\mathcal{C}^k \leftarrow \mathcal{C}^k + \rho(e^2_{n+0}) + \ldots + \rho(e^2_{n+t-1})$;
  end for
  for n from $\mathcal{N}' + 1$ to $\mathcal{N}$ with step 1 **do**
    $e^2_n \leftarrow \|\mathcal{H}^k(r_n) - q_n\|^2$;
    $\mathcal{C}^k \leftarrow \mathcal{C}^k + \rho(e^2_n)$;
  end for
end for

F. MODEL VERIFICATION BLOCK

After we have received $\mathcal{K}_3$ models from the model restrictor, we need to compute the scores and other statistics for them. For each transformation, the block computes a sequence of $\mathcal{N}$ residuals, the model score $\mathcal{C}(\theta)$, and an inlier mask vector, if needed. The score and residuals computation also allow vectorization with the proposed way of organizing the sequences $\mathcal{R}$ and $\mathcal{Q}$.

In the original RANSAC formulation, error functional ρ is taken as in Eq. (1). Different variations of ρ were proposed to increase the contribution of a single correspondence to the overall score (MSAC [12]); probabilistic approaches (MLESAC [12], AMLESAC [30], MAPSAC [13]), modeling the distribution of the error probability over inliers and outliers to evaluate the model hypothesis, and methods of maximum likelihood or estimation using a posteriori maximum are applied; to eliminate the need for user-defined an inlier-outlier threshold from Eq. (1) (MAGSAC [17]); etc.

Since the amount of input data can be quite large, and the evaluation must be performed for each transformation, *Model Verification* is usually a more expensive and time-consuming block than *Sampling* and *Model Computation*. Therefore, the use of SIMD extensions at the verification stage is especially advantageous. In the global case, the model score $\mathcal{C}$ is accumulated from values of some functional in each correspondence (e.g. error functional ρ from Eq. (1) and score from Eq. (2)). Using SIMD parallelism due to coinciding indexing in $\mathcal{R}$ and $\mathcal{Q}$, it is possible to process t correspondences and update the score by t values simultaneously. This idea is outlined in Pseudocode 1.

The model verification can be optimized using techniques aimed at the early rejection of the supposedly "bad" model hypotheses (retaining $\mathcal{K}_4 \leq \mathcal{K}_3$ ones) and, when verifying a certain model, discarding correspondences that are guaranteed to be inconsistent with it. In the literature, the following approaches were described:

- $T_{d,d}$ test [23], where each model is first verified by $d \ll \mathcal{N}$ randomly selected correspondences, and if all of them are inliers, then the remaining $\mathcal{N} - d$ correspondences are tested against the model, otherwise, the model is rejected;
- Sequential Probability Ratio Test (SPRT) test [24], where authors adapted A. Wald's sequential probability ratio test for the model verification strategy: for each correspondence, the likelihood ratio is calculated, and if it becomes larger than a certain threshold (the optimal value of which can be calculated), then the model is rejected;
- Bail-Out test [26], where the model is verified against a set of $d < \mathcal{N}$ randomly selected correspondences, and the model is rejected if the support for this set does not exceed so-far-the-largest number of inliers got from the entire sequence of correspondences;
- Space-Partitioning RANSAC [19], based on partitioning the joint correspondence space into a pair of regular grids. The grid cells are mapped by the verifying model, and for a certain cell on reference image the correspondences with query points falling outside the cell image under transformation are rejected; etc.

G. EARLY TERMINATION BLOCK

Typically, the number of RANSAC iterations is determined experimentally and supplemented by early termination techniques. Knowledge of the outlier ratio in the input data is necessary for an accurate estimation of the iterations number, but, in practice, it is rarely known reliably. Hence, a large enough number of iterations is specified with the early termination criterion in the main loop, which allows faster completion of the sample generation. The stopping condition depends on the used RANSAC modification and may be

- The adaptive number of iterations [20], where the number $\mathcal{I}$ of iterations is re-estimated using the Eq.(3) with the inlier ratio ϵ in so-far-the-best model;
- Criterion corresponding to $T_{d,d}$ [23] or SPRT [24] tests performed in *Model Verification* block;
- Criterion from PROSAC [25], MAGSAC [17], Latent RANSAC [46], other modifications with custom termination criterion; etc.

To implement the early termination, we suggest collecting some statistics that the early termination criterion operates with (e.g., the number of samples totally generated and rejected by the sample restrictor, the score of so-far-the-best model, its support, etc.) while running the main loop.

H. MODEL REFINEMENT BLOCK

As a result of the RANSAC main loop, a sequence of candidate models has been collected. This block is intended to refine them. We select a certain number of ''best'' models in terms of score and apply a non-minimal optimizer to them. Thus, we optimize the model from a larger subset of input data than a minimum sample and reduce the influence of random noise on the final hypothesis. Various strategies may be employed here. If refining is performed after the main loop, we may use, for example:

- Linearization of the least squares method following by singular value decomposition (SVD) solution; if approximations are required for linearization (e.g., in the case of a plane homography), then the answer will be inaccurate;
- Strategies for multidimensional nonlinear minimization, e.g. Gauss-Newton or Levenberg-Marquardt methods [74];
- Principal component analysis (PCA) for non-minimal estimation.

Alternatively, we may perform refinement during the main loop. This strategy is followed by some RANSAC modifications:

- Locally Optimized RANSAC (LO-RANSAC) [14], [15], where the optimization is performed not after the main loop, but during it: the main loop is supplemented by a local optimization step with the least squares on inliers performed after model verification when so-far-the-best model was updated;
- Graph-Cut RANSAC [16], that is similar to LO-RANSAC. The local optimization step uses the graph-cut algorithm and constructs new models until the support remains larger than its initial value. The result of the graph-cut algorithm is an optimal 1-0 (inlier-outlier) labeling for the correspondences, considering their spatial coherence and residuals; the resulting model is derived from $7\mathcal{M}$ [15] correspondences selected among ones with label 1; etc.

V. IMPLEMENTING PESAC FOR 2D HOMOGRAPHY ESTIMATION

This section provides a detailed description of how the PESAC framework can be configured for 2D homography estimation. In particular, we demonstrate possible restrictors for such a problem and the way of computing the homography matrix. Here we do not use the $\mathcal{EW}$ array in the input.

In *Sampling* block, we hold the sampling based on the probabilities specified in the $\mathcal{SP}$ array and employ techniques for generating numbers from an arbitrary discrete distribution. We use the Vose alias method [75] because of its low generation complexity, $O(1)$. That is a significant advantage since we need to generate quite a high amount of random numbers. Commonly, pseudo-random numbers are used in RANSAC, because of no need for reliable (and usually slow) algorithms for random numbers generating.

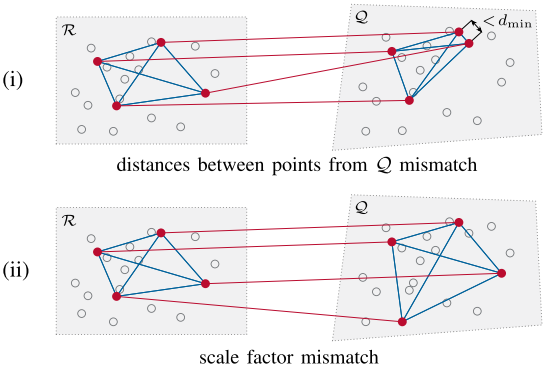


FIGURE 2. Example of samples that would be rejected by criterion with a specified index in *Sample Restrictor* block. Corresponding points from the sample are connected with segments.

We use the xoshiro128+ pseudo-random numbers generator for its speed [76] and giving easily vectorization.

For *Sample Restrictor* block, let us consider the sample restrictors proposed in [49] by Skoryukina et al. Introducing the notation $d_{\min}, D_{\max}$ for distance threshold values and calculating pair-wise distances

$$d_{ij}^r = |r_i - r_j|, \quad d_{ij}^q = |q_i - q_j|, \quad i, j = \overline{1, \mathcal{M}} \tag{5}$$

between points from $\mathcal{R}$ and $\mathcal{Q}$ respectively, we decide a sample is valid if it satisfies the following criteria:

(i) Distances between points from $\mathcal{R}$ and $\mathcal{Q}$ in the sample are large enough:

$$d_{ij}^r > d_{\min}, \quad d_{ij}^q > d_{\min}, \quad i, j = \overline{1, \mathcal{M}}. \tag{6}$$

(ii) Scale factor is similar among correspondences from the sample. Scale factor s_{ij} is determined for a pair of correspondences and is equal to the ratio of distances between two query points and two reference points:

$$s_{ij} = \frac{d_{ij}^q}{d_{ij}^r}, \quad i, j = \overline{1, \mathcal{M}}. \tag{7}$$

For each pair of correspondences, scale factor s_{ij} must not differ much from average value $\bar{s}$ of scale factor:

$$\left| \frac{s_{ij}}{\bar{s}} - 1 \right| < s_{\max}, \quad i, j = \overline{1, \mathcal{M}}, \tag{8}$$

where $\bar{s} = \binom{\mathcal{M}}{2}^{-1} \sum s_{ij}$.

Criterion (ii) might be used in tasks with the sample size $\mathcal{M} > 2$ only. For the case of 2D homography, we have $\mathcal{M} = 4$. Illustrations for the effect of these sample restrictors are given in Fig. 2. For each restrictor, there is a set of empty reference points on the left and a set of empty query points on the right. Points included in the sample are filled, and correspondences are connected with segments.

In *Model Computation* block, we implement the calculation of homography matrix coefficients by $\mathcal{M} = 4$ correspondences according to the method described in [77]. The idea of the method is as follows. Let us set the Cartesian coordinate system (x, y) on plane and consider some sample $\{i_1, \ldots, i_4\}$. We denote the subsets of reference and query

points conforming this sample by $\mathcal{R}_4 = \{r_{i_1}, \ldots, r_{i_4}\}$ and $\mathcal{Q}_4 = \{q_{i_1}, \ldots, q_{i_4}\}$. Consider transformations $\mathcal{H}_\mathcal{R} : \mathcal{U} \to \mathcal{R}_4$, $\mathcal{H}_\mathcal{Q} : \mathcal{U} \to \mathcal{Q}_4$, where $\mathcal{U}$ is the unit square with its vertices at points (0, 0), (1, 0), (1, 1), (0, 1) in Cartesian coordinates. Transformation $\mathcal{H} : \mathcal{R}_4 \to \mathcal{Q}_4$ can be found as a composition $\mathcal{H} = \mathcal{H}_\mathcal{Q} \times \mathcal{H}_\mathcal{R}^{-1}$. This is shown schematically in the commutative diagram:

$$\begin{array}{ccccc} & & \mathcal{U} & & \\ & \swarrow^{\mathcal{H}_\mathcal{R}} & & \searrow^{\mathcal{H}_\mathcal{Q}} & \\ \mathcal{R}_4 & & \xrightarrow{\quad\mathcal{H}\quad} & & \mathcal{Q}_4 \end{array} \tag{9}$$

Introducing the notation

$$\begin{aligned} \alpha &:= (r_{i_3} - r_{i_2}) \times (r_{i_3} - r_{i_4}), \\ \beta &:= (r_{i_3} - r_{i_2}) \times (r_{i_3} - r_{i_1}), \\ \gamma &:= (r_{i_3} - r_{i_1}) \times (r_{i_3} - r_{i_4}), \end{aligned} \tag{10}$$

where $u \times v = u^x v^y - u^y v^x$, one can obtain analytical expressions for the coefficients of matrix $(h_{ij})_\mathcal{R}$ of transformation $\mathcal{H}_\mathcal{R}$ in affine coordinates:

$$(h_{ij})_\mathcal{R} = \begin{pmatrix} r_{i_2}^x \frac{\gamma}{\alpha} - r_{i_1}^x & r_{i_4}^x \frac{\beta}{\alpha} - r_{i_1}^x & r_{i_1}^x \\ r_{i_2}^y \frac{\gamma}{\alpha} - r_{i_1}^y & r_{i_4}^y \frac{\beta}{\alpha} - r_{i_1}^y & r_{i_1}^y \\ \frac{\gamma}{\alpha} - 1 & \frac{\beta}{\alpha} - 1 & 1 \end{pmatrix}. \tag{11}$$

Replacing r_{i_j} with q_{i_j}, the corresponding expressions are obtained for calculating the matrix $(h_{ij})_\mathcal{Q}$ with coefficients of transformation $\mathcal{H}_\mathcal{Q}$. To obtain the elements of matrix (h_{ij}) of transformation $\mathcal{H}$ we are interested in, it remains to invert matrix $(h_{ij})_\mathcal{R}$ and multiply matrix $(h_{ij})_\mathcal{Q}$ by the result:

$$(h_{ij}) = (h_{ij})_\mathcal{Q} \cdot (h_{ij})_\mathcal{R}^{-1}. \tag{12}$$

Pseudocode 2 demonstrates the vectorization of the homography computation algorithm according to the method described above, for the general case of storing t variables in the register.

In block of *Model Restrictor*, we impose the following four criteria on the homography $\mathcal{H}$ with matrix (h_{ij}):

(i) Convex quadrilateral must remain convex after transformation.

(ii) Transformation must be reasonably projective: ratio of the matrix (h_{ij}) last row elements is not too large:

$$\frac{|h_{31}| + |h_{32}|}{|h_{33}|} < \tau_{\text{proj}}. \tag{13}$$

(iii) Area of the transformed quadrilateral must be larger than the threshold τ_{area}.

(iv) Aspect ratios of reference quadrilateral and its image after transformation, $q^{(\text{r})}$ and $q^{(\text{tr})}$ respectively, must be similar:

$$\left| \frac{\text{aspect ratio}\,(q^{(\text{tr})})}{\text{aspect ratio}\,(q^{(\text{r})})} - 1 \right| < \tau_{\text{asp}}. \tag{14}$$

The effect of these restrictors is shown in Fig. 3. The original rectangles are painted in gray, and their red-colored images are after some "bad" homography. Models producing illustrated results would be rejected by *Model Restrictor* with the corresponding index.

Finally, in *Model Verification* block we implement the algorithm outlined in Pseudocode 1 with MSAC score

$$\rho\left(e_i^2(\theta)\right) = \begin{cases} e_i^2(\theta), & \text{if } e_i^2(\theta) \le \mathcal{T}^2, \quad \text{(inlier)} \\ \mathcal{T}^2, & \text{otherwise,} \quad \text{(outlier)} \end{cases} \tag{15}$$

Pseudocode 2 Vectorized *Model Computation* for 2D Homography [77]

Input: samples $\{i_1^1, \ldots, i_4^1, i_1^2, \ldots, i_4^2, \ldots, i_1^{\mathcal{K}_1}, \ldots, i_4^{\mathcal{K}_1}\}$;
  reference points $\{\mathcal{R}_4^1, \mathcal{R}_4^2, \ldots, \mathcal{R}_4^{\mathcal{K}_1}\}$,
    where $\mathcal{R}_4^j = \{r_{i_1^j}, \ldots, r_{i_4^j}\}$;
  query points $\{\mathcal{Q}_4^1, \mathcal{Q}_4^2, \ldots, \mathcal{Q}_4^{\mathcal{K}_1}\}$,
    where $\mathcal{Q}_4^j = \{q_{i_1^j}, \ldots, q_{i_4^j}\}$;
  integer value t – the amount of samples
    processed simultaneously.
Output: matrices of homographies $\{\mathcal{H}^1, \ldots, \mathcal{H}^{\mathcal{K}_1}\}$,
    where $\mathcal{H}^j : \mathcal{R}_4^j \to \mathcal{Q}_4^j$.
$\mathcal{K}' \leftarrow \lfloor \mathcal{K}_1 / t \rfloor \cdot t$
for k from 1 to $\mathcal{K}'$ with step t **do**
  compute transformations
    $\{\mathcal{H}_\mathcal{R}^{k+0}, \ldots, \mathcal{H}_\mathcal{R}^{k+t-1}\}$ coefficients by Eq. (11);
  compute inverse transformations
    $\{\left(\mathcal{H}_\mathcal{R}^{k+0}\right)^{-1}, \ldots, \left(\mathcal{H}_\mathcal{R}^{k+t-1}\right)^{-1}\}$;
  compute transformations
    $\{\mathcal{H}_\mathcal{Q}^{k+0}, \ldots, \mathcal{H}_\mathcal{Q}^{k+t-1}\}$ coefficients by Eq. (11);
  compute compositions
    $\{\mathcal{H}_\mathcal{Q}^{k+0} \times \left(\mathcal{H}_\mathcal{R}^{k+0}\right)^{-1}, \ldots, \mathcal{H}_\mathcal{Q}^{k+t-1} \times \left(\mathcal{H}_\mathcal{R}^{k+t-1}\right)^{-1}\}$;
end for
for k from $\mathcal{K}' + 1$ to $\mathcal{K}_1$ with step 1 **do**
  compute transformation $\mathcal{H}_\mathcal{R}^k$ coefficients by Eq. (11);
  compute inverse transformation $\left(\mathcal{H}_\mathcal{R}^k\right)^{-1}$;
  compute transformation $\mathcal{H}_\mathcal{Q}^k$ coefficients by Eq. (11);
  compute composition $\mathcal{H}_\mathcal{Q}^k \times \left(\mathcal{H}_\mathcal{R}^k\right)^{-1}$;
end for

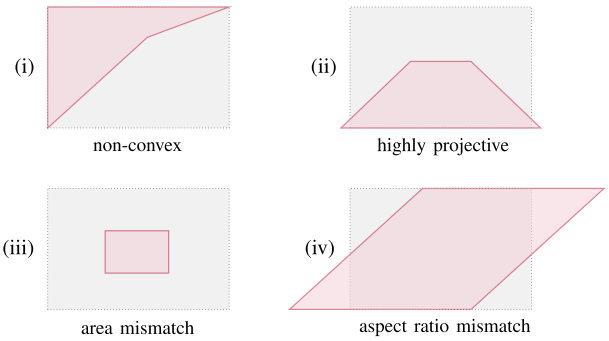


FIGURE 3. Examples of the rectangle and its image (densely dotted gray and solid red respectively) after some "bad" homography that would be rejected by criterion with specified index in ***Model Restrictor*** block.

and use the criterion of adaptive number of iterations in *Early Termination* block. *Model Refinement* is executed after the main loop, and this block implements PCA with a non-minimal solver.

VI. COMPUTATIONAL EXPERIMENTS

The rising number of mobile applications based on image registration and recognition (e.g. augmented reality, computed tomography, autonomous navigation on UAVs, facial recognition systems) leads to the development of image registration techniques. A common approach for image registration consists of steps shown schematically in Fig. 4. The first step is to compute so-called keypoints in the image and a description of the neighborhood of each keypoint, which is invariant to certain transformations. As a result, for image I we get a feature set

$$\left\{(p_i, f_i),\ i = \overline{1, N(I)}\right\}, \tag{16}$$

where $p_i = (x_i, y_i)$ are coordinates of i-th keypoint, f_i is the descriptor of i-th keypoint neighborhood, $N(I)$ is the total amount of keypoints on image I. Once the features have been extracted, they need to be matched: descriptors from the reference image are then compared to descriptors extracted from the query image to build a sequence $\mathcal{S}$ of reference-query points correspondences. Then one can employ RANSAC to estimate the underlying geometric transformation between images.

In our experiments, we used RANSAC for tasks that arise in the document processing field: classification, localization, and tracking. A document is a 2D object, so we can describe the change in its location between a pair of images with a 2D homography.

All tested variants of the PESAC framework were specified by the components described in Section V. In addition, we experimented with disabling optional blocks of *Sample Restrictor* and *Model Restrictor* and SIMD vectorization in algorithms outlined in Pseudocodes 1, 2. Per one iteration of the main loop, 200 samples were generated. In *Sample Restrictor* block, $d_{\min}$ and $s_{\max}$ were taken equal to 50 and 0.5 respectively, and in *Model Restrictor* block, τ_{proj}, τ_{area}, τ_{asp} were set to 5, 0.05, 0.5 respectively.

We compared PESAC with the classical version of RANSAC, five variations of USAC from the OpenCV-4.6.0 library, Latent RANSAC, and Space-Partitioning RANSAC. Classical OpenCV RANSAC implements score given by Eq. (1) and early termination by an adaptive number of iterations. As it is mentioned in OpenCV documentation [78], methods USAC_DEFAULT, USAC_FAST involve local optimization from LO-RANSAC [14] for model refinement, but USAC_FAST uses fewer iterations. Latent RANSAC implementation also had LO-RANSAC [14] for model refinement, and Space-Partitioning RANSAC had least squares fitting after the main loop. USAC_MAGSAC is the implementation of MAGSAC++ [18]. USAC_PROSAC had PROSAC [25] sampling, and other four variations of USAC used uniform sampling to generate samples. Latent RANSAC implementation included PROSAC [25] sampling, and the implementation of Space-Partitioning RANSAC – P-NAPSAC [29] sampling. All but USAC_MAGSAC compute MSAC score, while USAC_MAGSAC uses its own score type. All OpenCV USAC implementations and Space-Partitioning RANSAC execute SPRT in model verification and use early termination criterion associated with SPRT. As a sample check, in the module of sampling minimal subset all these implementations check if no 3 points lie on the same line and hold the orientation test [41], [42]. With Latent RANSAC we set maximum number of collisions to 2000 for document classification and 200 for tracking.

For all compared methods, common input parameters were taken to be identical. For document classification and localization task, inlier-outlier threshold $\mathcal{T}$ from Eq. (1) was taken equal to 10, maximum number $\mathcal{I}$ of iterations was 10^6, confidence $\mathcal{P}$ in solution was 0.9999. For document tracking task, these values were set to 5, 10^4, and 0.99 correspondingly.

We ran the experiments on AMD FX-8350 (with x86_64 architecture) and ARM Cortex-A73 (with ARMv8 architecture) CPUs in single-threaded mode. SIMD vectorization was carried out using SSE intrinsics for x86_64, NEON intrinsics for ARMv8, and 128-bit registers, storing 4 single-precision or 2 double-precision floating-point values.

A. DATASET

We experimented on images from the open dataset MIDV-2020 [79] containing identity documents of 10 types. The following subsets of this dataset were used:

- **MIDV-2020 / templates**
  For each of 10 document types, this subset contains 100 ideal images, i.e. templates with different personal data. Rectangle bounding the template is considered having vertices (0, 0), (w, 0), (w, h), (0, h), where w and h are template linear dimensions.
- **MIDV-2020 / photo**
  For each of 10 document types, this subset contains 100 photos with a resolution of 2268 × 4032 pixels. The subset contains pictures taken at different levels of illumination, with challenging background variations, with strong projective distortions, etc. For each photo, there is a JSON annotation, that contains the correct coordinates of the document quadrilateral presented on a certain picture.
- **MIDV-2020 / clips**
  The subset contains 100 video clips for each of 10 document types. The smallest clip has 38 frames, whereas the largest one has 129 frames. The total number of frames in 100 clips is 68409. Frames have a resolution of 2160 × 3840 pixels. In clip annotation, one can find the correct document quadrilateral coordinates for each frame of that clip.

All images in these subsets are stored in JPEG format. For the task of document classification and localization, we used

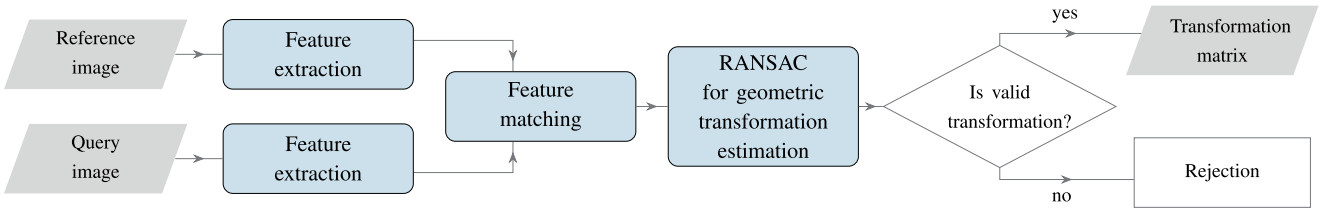


FIGURE 4. Flowchart of the common approach for geometric transformation estimation between two images using RANSAC.

subsets of photos and templates. Clips subset was used for the document tracking. Pictures in the dataset have complex background resulting in the detection of a great number of keypoints, and that is another reason why we specified such a huge number of RANSAC iterations.

B. FORMATION OF RANSAC INPUT

Before feature extraction each RGB image was preprocessed by scaling from (w, h) linear size to $(1000\,w/h, 1000)$ and converting to gray-scale. Then an image pyramid with 3 levels of $(1, 2/3, 1/2)$ resolution was made for computing multi-scale image features, and the algorithm for keypoints extraction was run on each pyramid level with a maximum limit of $5 \cdot 10^4$ points. We used YAPE [80], [81] keypoint detector with a radius of 4 due to its computation performance. From the keypoint neighborhood of size 32×32, RFDoc [82] descriptors were computed with a 128-bit feature vector.

Feature matching was implemented by comparing the Hamming distance d_H between descriptors. Keypoint pair with $d_H < \tau_H$ was considered as the correspondence with sampling probability of $\max\{0;\ (\tau_H - d_H)/\tau_H\}$, where τ_H was taken equal to 32. As a prefiltering, a fixed number of "best" correspondences in terms of sampling probability was inserted in sequence $\mathcal{S}$. It was 10^3 for document identification problem and $5 \cdot 10^2$ for document tracking.

C. DOCUMENT CLASSIFICATION AND LOCALIZATION

Identity documents can be considered semi-structured objects because they contain static elements of the document template and variable elements – personal data. For a document template of a certain type, an approximate location of personal data information is known. Given a predefined set of templates, the identification problem is to classify the document image as one of the classes represented by document templates of different types, or as a null class that corresponds to the absence of a known document type. Current approaches for solving such a problem are based on the matching of features such as singular points and descriptors, but the feature space can be extended with line segments, quadrilaterals, etc. [8].

Given image I and document templates $\{t^{(j)}\}$ of different types, the task is to identify the type $t^{(i)}$ of the document on image I and to determine its quadrilateral coordinates. Here, we do not follow the approach proposed in [8] for localization problem with preliminary extraction of lines and quadrilaterals in the input image and filtering features

FIGURE 5. Example of correctly identified document type for **MIDV-2020 / photo** grc_passport/28: matching with a) correct document type (grc_passport) and b) incorrect document type (fin_id). Inliers drawn. Localization result is illustrated by green quadrilateral (**D = 24.52, IoU = 0.98**). For clarity, in case of incorrect document type here we also presented quadrilateral obtained by corresponding homography (**D = 1762.84, IoU = 0.32**).

on template images before matching, since the purpose is to test RANSAC itself. The following approach was used as a solution. Each of the 10 templates was matched with the input image. For the template, keypoints were extracted and their descriptors were computed only inside pre-selected regions containing static information, i.e. appearing on any instance of the certain document type. After RANSAC, a set of $\leq$ 10 homographies was obtained (for some templates, transformation may not be found). We selected the document type corresponding to the homography with the largest inlier ratio as the answer.

We have denoted the ratio of images with correctly identified document type among **MIDV-2020 / photo** via Accuracy. Given the correctly identified type for a certain image, we were to determine whether the corresponding homography had localized the document quadrilateral. Introducing the notation $\mathcal{H}$ for estimated homography transforming template quadrilateral q^t with vertices $\{q_i^t\}_{i=1}^4$ into image

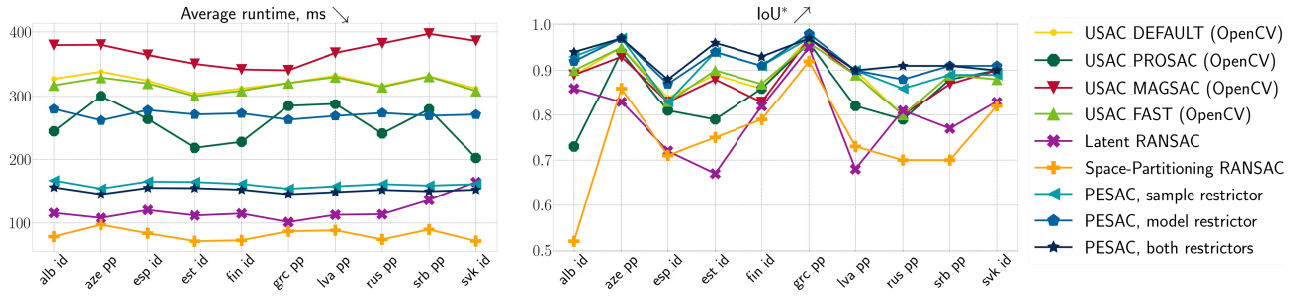


FIGURE 6. Plots of the average runtime and IoU* criterion by document types in classification and localization task for MIDV-2020 / photo.

TABLE 1. Average runtime and metric values of the PESAC framework implementations in document classification and localization task for MIDV-2020 / photo.

Implementation	Average time per one non-degenerate RANSAC run, ms		Metric values		
	CPU: AMD FX-8350	CPU: ARM Cortex-A73	Accuracy ↗	D* ↗	IoU* ↗
	without SIMD ⟶ with SIMD	without SIMD ⟶ with SIMD			
PESAC, no restrictors	f32 13999.19 → 4631.29 (-66.91%) f64 12270.14 → 6577.60 (-46.39%)	f32 21592.45 → 11039.61 (-48.87%) f64 21592.45 → 19463.88 (-9.85%)	0.907	0.882	0.883
PESAC, sample restrictor	f32 151.11 → 118.35 (-21.68%) f64 167.29 → 160.31 (-4.17%)	f32 251.14 → 231.27 (-7.91%) f64 300.00 → 291.17 (-2.94%)	0.933	0.907	0.909
PESAC, model restrictor	f32 311.67 → 232.17 (-25.50%) f64 316.28 → 270.31 (-14.53%)	f32 535.78 → 445.45 (-16.85%) f64 688.45 → 638.33 (-7.28%)	0.943	0.916	0.919
PESAC, both restrictors	f32 136.08 → 114.31 (-15.99%) f64 151.03 → 148.38 (-1.75%)	f32 217.88 → 212.54 (-2.45%) f64 265.45 → 259.91 (-2.08%)	0.962	0.924	0.927

TABLE 2. Average runtime and metric values of existing methods' implementations in document classification and localization task for MIDV-2020 / photo.

Implementation	Average time per one non-degenerate RANSAC run, ms		Metric values		
	CPU: AMD FX-8350	CPU: ARM Cortex-A73	Accuracy ↗	D* ↗	IoU* ↗
RANSAC (OpenCV)	24584.61	48374.03	0.903	0.878	0.878
USAC_DEFAULT (OpenCV)	321.96	485.31	0.905	0.883	0.887
USAC_PROSAC (OpenCV)	254.65	373.09	0.895	0.845	0.848
USAC_FAST (OpenCV)	318.10	452.33	0.908	0.885	0.887
USAC_MAGSAC (OpenCV)	369.09	593.07	0.909	0.878	0.879
Latent RANSAC	120.03	241.78	0.894	0.791	0.794
Space-Partitioning RANSAC	81.60	138.61	0.865	0.738	0.750

I, q with vertices $\{q_i\}_{i=1}^4$ for document quadrilateral localized on image I (so that $\mathcal{H}\, q_i^t = q_i$), m with vertices $\{m_i\}_{i=1}^4$ for quadrilateral with correct coordinates on image I, we defined metric D called corner residual in the following way:

$$\mathrm{D}(q, m) = \max_i \frac{\|m_i - q_i\|_2}{\text{perimeter}\,(m)}. \tag{17}$$

Additionally, we calculated the region overlap criterion Intersection over Union (IoU) given by formula

$$\mathrm{IoU}(q, m) = \frac{\text{area}\,(q \cap m)}{\text{area}\,(q \cup m)}. \tag{18}$$

See that $\mathrm{D} \in [0; +\infty)$ and $\mathrm{IoU} \in [0; 1]$, so smaller D and higher IoU values mean more similar coordinates of quadrilaterals q and m. Below there are given values of ratios of photos with $\mathrm{D} < 0.02$ and with $\mathrm{IoU} > 0.9$, defined by D* and IoU* correspondingly.

An example document type identification is shown in Fig. 5 with templates of a Greek passport and Finland ID card and a region of the Greek passport as an input image. Fig. 5a) demonstrates correctly identified type and localized document quadrilateral, and Fig. 5b) demonstrates transformation for the wrong document type. Connected points are inliers of the given homography.

Computational results are presented in Tables 1–3. Average time in Tables 1, 2 stands for one non-degenerate RANSAC run, when it managed to return some homography estimation (each implementation had more than 99.6% non-degenerate runs). Tables 1, 2 also review metric values for PESAC and OpenCV implementations correspondingly. In PESAC, metric values for 32-bit floating points have not changed significantly from using 64-bit data type, and Table 1 gives values for 32-bit type only. Statistics about the average number of generated samples ($\overline{\mathcal{K}}$), non-rejected samples ($\overline{\mathcal{K}}_1$), computed models ($\overline{\mathcal{K}}_2$), and non-rejected models ($\overline{\mathcal{K}}_3$) is presented in Table 3.

The average runtime demonstrated up to 83 (122) times speedup with 64-bit (32-bit) floating point data type between "no restrictors, without SIMD" and "both restrictors, with SIMD" PESAC versions on x86_64 CPU (and up to 83 (101) times on ARMv8 CPU). Sample and model restrictors rejected a huge part of the samples and models: values of $\overline{\mathcal{K}}_1$ and $\overline{\mathcal{K}}_3$ in Table 3 are much smaller than $\overline{\mathcal{K}}$ and $\overline{\mathcal{K}}_2$, respectively, if using corresponding restrictor. The use of SIMD extensions itself gives up to 16% speedup with SSE

TABLE 3. Statistics on restrictor blocks of the PESAC framework implementations in document classification and localization task for MIDV-2020 / photo. For 32-bit floating point data type.

Implementation	Statistics on restrictor blocks			
	$\overline{\mathcal{K}}$	$\overline{\mathcal{K}}_1$	$\overline{\mathcal{K}}_2$	$\overline{\mathcal{K}}_3$
PESAC, no restrictors	903398	903398	759982	759982
PESAC, sample restrictor	918471	1342	1335	1335
PESAC, model restrictor	918881	918881	773025	2081
PESAC, both restrictors	920113	1348	1341	91

intrinsics, and up to 2.5% with NEON intrinsics in fastest version ''both restrictors''. Operating with 128-bit SIMD registers storing 4 (2) single (double)-precision floating-point values, the maximum possible acceleration for vectorized parts of framework could be 75% (50%). So, depending on the runtime of ''no restrictors'' version we can conclude that speedup with SIMD in *Model Verification* and *Model Computation* blocks is quite good, and using vectorization in them is very relevant.

Among other implementations, Latent RANSAC and Space-Partitioning RANSAC were the best in terms of time but gave the lowest values of Accuracy and localization metrics. PESAC version with ''both restrictors'' ran up to 1.5 times slower than Space-Partitioning RANSAC but with 11.2% better Accuracy and more than 20% better localization metrics. Fig. 6 illustrates that for most document types versions of proposed framework showed higher values for criterion IoU*. At the same time, one can see that ''both restrictors'' and ''sample restrictor'' versions were much faster than others. It reveals the success of using the restrictors. Note also that ''no restrictors'' version was much faster than OpenCV RANSAC, which is due in part to good memory locality.

D. DOCUMENT TRACKING

Tracking is a superposition of coordinate systems of one object on different frames of a video stream. In our experiment, the object is a document. Document tracking finds application in many practical usages. It is performed for the alignment of coordinate systems in the task of assessing the presence or absence of optically variable devices (OVDs) (e.g. holograms) changing in the video stream on the document. If the analyzed OVDs are small-sized, a high alignment accuracy of two adjacent frames is required. In addition, tracking accuracy directly affects the determining of document rotation angle relative to the camera coordinate system – rotation is needed to assess whether the document had moved within the frame.

Formally, the input data of tracking are sequence $\{I_n\}$ of frames, each containing the same document instance. The frame-to-frame document movement must be followed by a sequential matching of neighboring frames. When matching frames I_j, I_{j+1}, for the frame I_j keypoints filtering is performed: only points located inside the document quadrilateral are used. For the first clip frame, the quadrilateral coordinates

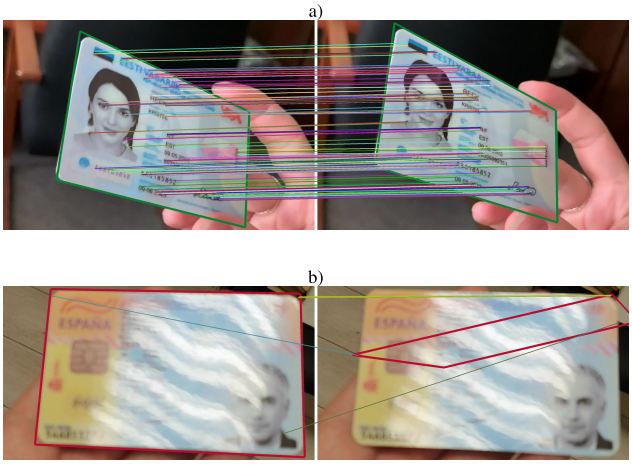


FIGURE 7. Example of tracked frames from MIDV-2020 / clips: a) correctly tracked (clip est_id/87, frames 000445, 000451), inlier ratio = 0.402, and b) incorrectly tracked (clip esp_id/90, frames 000115, 000121), inlier ratio = 0.079. Inliers drawn. quadrilaterals illustrate tracking result: right quadrilateral for current pair of frames and left for previous.

were considered known a priori and extracted from the clip annotation.

Among other things, document tracking is caused by the fact that it can be difficult to obtain high accuracy with frame-by-frame localization, as such methods analyze only areas with static information. Tracking uses keypoints from the whole document. In particular, personal data is often more resolvable when captured with a mobile phone camera than areas with static information under the same conditions.

To evaluate tracking quality, we calculated region overlap, tracking length, and failure rate metrics [83]. The overlap criterion IoU was given by Eq. (18). The corresponding binary criterion was obtained by thresholding: if we had $\mathrm{IoU}(q^{(j)}, m^{(j)}) < \tau$ for some fixed τ, the quadrilateral $q^{(j)}$ on frame I_j was considered incorrectly localized and a decision about tracker failure was issued; otherwise, the quadrilateral was considered correct and the tracking process continued. Hereinafter, we took τ equal to 0.5. An example of correctly and incorrectly tracked frames is shown in Fig. 7, where the ''bad'' tracking result was caused by glares.

If tracker failure was obtained as the result of matching frames I_{j-1}, I_j, we re-initialized the tracker: for the frame I_j, the coordinates of document quadrilateral were extracted from the clip annotation, and the next pair I_j, I_{j+1} was matched, and so on, until either another tracker failure or the end of the clip. Therefore, to assess the tracking quality, we used two more metrics based on IoU. The P_τ criterion corresponded to the number of successfully tracked frames from the initial frame to the first tracking failure:

$$P_\tau = \min\left\{j : \mathrm{IoU}\left(q^{(j)}, m^{(j)}\right) \geq \tau, \\ \mathrm{IoU}\left(q^{(j+1)}, m^{(j+1)}\right) < \tau\right\}. \quad (19)$$

TABLE 4. Average runtime and metric values of the PESAC framework implementations in document tracking task for MIDV-2020 / clips.

Implementation	Average time per one non-degenerate RANSAC run, ms				Metric values	
	CPU: AMD FX-8350		CPU: ARM Cortex-A73		$\overline{P}_{0.5}\nearrow$	$\overline{\lambda}_{0.5}\searrow$
	without SIMD $\longrightarrow$ with SIMD		without SIMD $\longrightarrow$ with SIMD			
PESAC, no restrictors	f32	19.08 $\rightarrow$ 7.52 (-60.58%)	f32	27.60 $\rightarrow$ 16.67 (-39.60%)	0.6733	0.0937
	f64	13.48 $\rightarrow$ 7.84 (-41.83%)	f64	31.34 $\rightarrow$ 24.81 (-20.83%)	0.6698	0.0966
PESAC, sample restrictor	f32	1.51 $\rightarrow$ 1.12 (-25.82%)	f32	1.91 $\rightarrow$ 1.55 (-18.84%)	0.7796	0.0513
	f64	1.38 $\rightarrow$ 1.06 (-23.18%)	f64	2.28 $\rightarrow$ 2.21 (-3.07%)	0.7786	0.0515
PESAC, model restrictor	f32	2.16 $\rightarrow$ 1.48 (-31.48%)	f32	2.77 $\rightarrow$ 2.20 (-20.57%)	0.7748	0.0528
	f64	1.88 $\rightarrow$ 1.43 (-23.93%)	f64	3.50 $\rightarrow$ 3.31 (-5.42%)	0.7755	0.0528
PESAC, both restrictors	f32	1.44 $\rightarrow$ 1.10 (-23.61%)	f32	1.77 $\rightarrow$ 1.50 (-15.25%)	0.7913	0.0498
	f64	1.19 $\rightarrow$ 0.92 (-22.68%)	f64	2.21 $\rightarrow$ 2.15 (-2.71%)	0.7907	0.0499

TABLE 5. Average runtime and metric values of existing methods' implementations in document tracking task for MIDV-2020 / clips.

Implementation	Average time per one non-degenerate RANSAC run, ms		Metric values	
	CPU: AMD FX-8350	CPU: ARM Cortex-A73	$\overline{P}_{0.5}\nearrow$	$\overline{\lambda}_{0.5}\searrow$
RANSAC (OpenCV)	51.34	104.30	0.7477	0.0672
USAC_DEFAULT (OpenCV)	4.63	8.65	0.7073	0.0805
USAC_PROSAC (OpenCV)	0.90	1.57	0.7374	0.0678
USAC_FAST (OpenCV)	1.50	2.58	0.7074	0.0811
USAC_MAGSAC (OpenCV)	1.82	3.53	0.7245	0.0750
Latent RANSAC	0.82	2.13	0.6630	0.1000
Space-Partitioning RANSAC	2.06	4.08	0.7582	0.0582

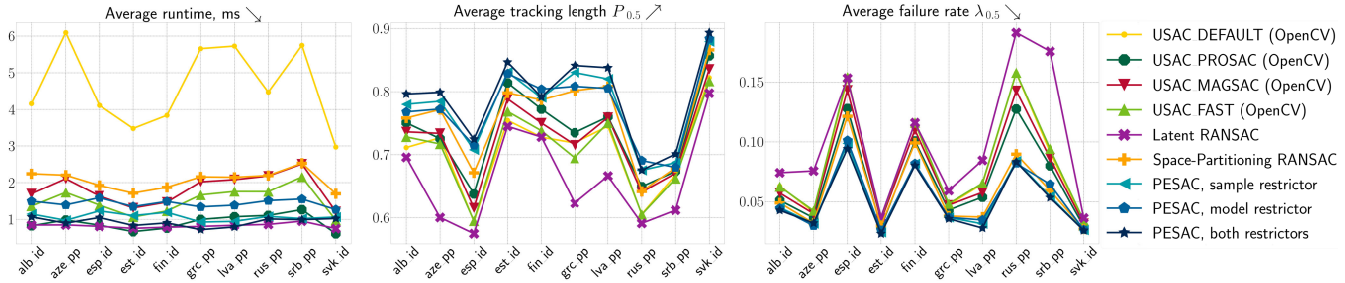


FIGURE 8. Average runtime, average tracking length $\overline{P}_{0.5}$, and average failure rate $\overline{\lambda}_{0.5}$ by document types in tracking task for MIDV-2020 / clips.

For the whole dataset the average tracking length $\overline{P}_\tau$ was considered:

$$\overline{P}_\tau = \frac{1}{|\text{ clips}|}\sum_{\text{clips}}\frac{P_\tau}{\text{clip length} - 1}. \tag{20}$$

Yet another criterion we calculated is λ_τ, which is associated with tracking failure rate, and its average value $\overline{\lambda}_\tau$:

$$\lambda_\tau = \left|\left\{j:\ \text{IoU}\left(q^{(j)}, m^{(j)}\right) < \tau\right\}\right|, \tag{21}$$

$$\overline{\lambda}_\tau = \frac{1}{|\text{ clips}|}\sum_{\text{clips}}\frac{\lambda_\tau}{\text{clip length} - 1}. \tag{22}$$

So we have $\overline{P}_\tau = 1$, $\overline{\lambda}_\tau = 0$ for an ideal tracker, and $\overline{P}_\tau = 0$, $\overline{\lambda}_\tau = 1$ for a totally unusable tracker.

The average runtime of the considered PESAC and OpenCV implementations is given in Tables 4 and 5 correspondingly. Values here again stand for one non-degenerate RANSAC run. All implementations had more than 95% non-degenerate runs. Metric values when using the PESAC framework (for both cases with 32-bit and 64-bit floating points) are presented in Table 4, and Table 5 contains the metric values of OpenCV implementations. The average numbers of generated and non-rejected samples ($\overline{\mathcal{K}}$ and $\overline{\mathcal{K}}_1$), computed and non-rejected models ($\overline{\mathcal{K}}_2$ and $\overline{\mathcal{K}}_3$) are shown in Table 6.

TABLE 6. Statistics on restrictors of the PESAC framework implementations in document tracking task for MIDV-2020 / clips. For 32-bit floating point data type.

Implementation	Statistics on restrictor blocks			
	$\overline{\mathcal{K}}$	$\overline{\mathcal{K}}_1$	$\overline{\mathcal{K}}_2$	$\overline{\mathcal{K}}_3$
PESAC, no restrictors	1949	1949	1668	1668
PESAC, sample restrictor	1897	42	42	42
PESAC, model restrictor	1984	1984	1692	61
PESAC, both restrictors	1800	43	42	36

Acceleration between "no restrictors, without SIMD" and "both restrictors, with SIMD" PESAC versions was 14.6 times on both x86_64 and ARMv8 CPUs when using 64-bit data type; 17.3 times on x86_64 and 18.4 times on ARMv8 CPUs when using 32-bit data type. We see that "both restrictors" had the lowest average runtime among versions of the PESAC framework and produced the most accurate models demonstrating the highest average tracking length and lowest failure rate (Table 4). Using the 32-bit floating point data type instead of the 64-bit type resulted in improved metric values and a little longer runtime. Note that in tracking, the current estimation is based on the result of the previous

one, which is not the case in localization task, so changing the data type affects the computations more significantly. The use of SIMD extensions in ''both restrictors'' version accelerated the method by 23.61% on x86_64 and by 15.25% on ARMv8 when using 32-bit data type (22.68% on x86_64, 2.71% on ARMv8 when using 64-bit data type).

Among other methods, USAC_PROSAC and Latent RANSAC had the fastest average runtime and were slightly faster than our version with ''both restrictors'' in this experiment (on average over all clips). At the same time, PESAC resulted in better metric values (see Table 5, Fig. 8). This effect can be reasoned by the fact that unlike document localization, where PROSAC sampling was not faster, in tracking the whole region inside the reference frame quadrilateral is analyzed, which leads to a greater number of correct matches among the top-ranked ones after prefiltering.

VII. CONCLUSION

In this paper, we presented PESAC, the Parallel Efficient Sample Consensus framework for the family of RANSAC-based methods that is adapted for SIMD CPUs and possesses memory locality. The PESAC consists of the following blocks: *Sampling* and *Sample Restrictor*, *Model Computation* and *Model Restrictor*, *Model Verification*, *Early Termination*, and *Model Refinement*. Since *Model Computation* and *Model Verification* are the most time-consuming, we aimed the framework to reduce the time taken for their execution by the preliminary rejection ''bad'' samples and model hypothesis from consideration. Within *Sample Restrictor* and *Model Restrictor* blocks, the collection of sample and model validity criteria is listed.

The PESAC framework uses a special way of storing data in the memory that is suitable for vectorization. Generating $K \gg 1$ samples per one iteration of the main loop allows efficient vectorization of pseudo-random numbers generation in *Sampling* block, ensures memory locality, and provides a field for using vectorization within the following blocks. *Model Computation* and *Model Verification* blocks were speeded up by using SIMD extensions. We demonstrate such vectorized algorithms for 2D homography computation and model verification. A complete example of configuring the framework for a 2D homography estimation problem is also presented.

Experiments included the comparison of the PESAC framework with several USAC implementations from OpenCV-4.6.0, Latent RANSAC and RANSAC with space partitioning. Homography estimation in tasks of document classification and tracking on the MIDV-2020 dataset demonstrated that all optimizations resulted in significant speedup on x86_64 and ARMv8 CPUs with a concomitant increase in accuracy.

Limitations of the study: The first limitation grows from the reasons that generally limit the benefits of SIMD vectorization. If particular method has a lot of dependencies on previous calculations and conditional branching, vectorization may not be useful. The second one is related to the fact that we only focused on the case of homography estimation in the field of identity document recognition. The work can be extended to address other applications and more datasets can be investigated. Also our study does not include testing the proposed framework in computer vision tasks such as PnP or epipolar geometry estimation, etc.

Future work: The direction of the future work is to probe the PESAC implementation for homography estimation problem on other datasets. Moreover, the PESAC framework might be implemented and tested in other problems, such as estimation of fundamental or essential matrices that involve a significant amount of matrix and vector operations, which are well suited for SIMD vectorization.

REFERENCES

[1] M. A. Fischler and R. Bolles, ''Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,'' *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[2] P. H. S. Torr and D. W. Murray, ''The development and comparison of robust methods for estimating the fundamental matrix,'' *Int. J. Comput. Vis.*, vol. 24, no. 3, pp. 271–300, 1997.
[3] M. Fularz, M. Kraft, A. Schmidt, and A. Kasinski, ''FPGA implementation of the robust essential matrix estimation with RANSAC and the 8-point and the 5-point method,'' in *Facing the Multicore—Challenge II*. Berlin, Germany: Springer, 2012, pp. 60–71.
[4] B. Tippetts, S. Fowers, K. Lillywhite, D.-J. Lee, and J. Archibald, ''FPGA implementation of a feature detection and tracking algorithm for real-time applications,'' in *Proc. Int. Symp. Vis. Comput. (ISVC)*, Lake Tahoe, NV, USA, Nov. 2007, pp. 682–691.
[5] J. W. Tang, N. Shaikh-Husin, and U. U. Sheikh, ''FPGA implementation of RANSAC algorithm for real-time image geometry estimation,'' in *Proc. IEEE Student Conf. Res. Developement*, Dec. 2013, pp. 290–294.
[6] G. H. Lee, F. Faundorfer, and M. Pollefeys, ''Motion estimation for self-driving cars with a generalized camera,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2746–2753.
[7] A. Lopez, C. Canero, J. Serrat, J. Saludes, F. Lumbreras, and T. Graf, ''Detection of lane markings based on ridgeness and RANSAC,'' in *Proc. IEEE Intell. Transp. Syst.*, Sep. 2005, pp. 254–259.
[8] N. Skoryukina, V. Arlazarov, and D. Nikolaev, ''Fast method of ID documents location and type identification for mobile and server application,'' in *Proc. Int. Conf. Document Anal. Recognit. (ICDAR)*, Sep. 2019, pp. 850–857.
[9] P. F. U. Gotardo, O. R. P. Bellon, K. L. Boyer, and L. Silva, ''Range image segmentation into planar and quadric surfaces using an improved robust estimator and genetic algorithm,'' *IEEE Trans. Syst. Man Cybern., B, Cybern.*, vol. 34, no. 6, pp. 2303–2316, Dec. 2004.
[10] K. Wang and Y. Qian, ''Fast and accurate iris segmentation based on linear basis function and RANSAC,'' in *Proc. 18th IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 3205–3208.
[11] X. Qian and C. Ye, ''NCC-RANSAC: A fast plane extraction method for 3-D range data segmentation,'' *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2771–2783, Dec. 2014.
[12] P. H. S. Torr and A. Zisserman, ''MLESAC: A new robust estimator with application to estimating image geometry,'' *Comput. Vis. Image Understand.*, vol. 78, no. 1, pp. 138–156, Apr. 2000.
[13] P. H. Torr, ''Bayesian model estimation and selection for epipolar geometry and generic manifold fitting,'' *Int. J. Comput. Vis.*, vol. 50, no. 1, pp. 35–61, Oct. 2002.
[14] O. Chum, J. Matas, and J. Kittler, ''Locally optimized RANSAC,'' in *Pattern Recognition* (Lecture Notes in Computer Science), vol. 2781. Berlin, Germany: Springer, 2003, pp. 236–243.
[15] K. Lebeda, J. Matas, and O. Chum, ''Fixing the locally optimized RANSAC—Full experimental evaluation,'' in *Proc. BMVC*, vol. 2, 2012, pp. 1–55.
[16] D. Barath and J. Matas, ''Graph-cut RANSAC,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6733–6741.
[17] D. Barath, J. Matas, and J. Noskova, ''MAGSAC: Marginalizing sample consensus,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10189–10197.

[18] D. Baráth, J. Noskova, M. Ivashechkin, and J. Matas, "MAGSAC++, a fast, reliable and accurate robust estimator," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1301–1309.

[19] D. Barath and G. Valasek, "Space-partitioning RANSAC," in *Proc. ECCV*, 2022, pp. 721–737.

[20] P. H. S. Torr, A. Zisserman, and S. J. Maybank, "Robust detection of degenerate configurations while estimating the fundamental matrix," *Comput. Vis. Image Understand.*, vol. 71, no. 3, pp. 312–333, Sep. 1998.

[21] B. Tordoff and D. W. Murray, "Guided sampling and consensus for motion estimation," in *Proc. ECCV*, Jan. 2002, pp. 82–96.

[22] D. R. Myatt, P. H. Torr, S. J. Nasuto, J. M. Bishop, and R. Craddock, "NAPSAC: High noise, high dimensional robust estimation—It's in the bag," in *Proc. Brit. Mach. Vis. Conf.*, vol. 2, Jan. 2002, pp. 1–11.

[23] J. Matas and O. Chum, "Randomized RANSAC with $T_{d,d}$ test," *Image Vis. Comput.*, vol. 22, no. 10, pp. 837–842, Sep. 2004.

[24] J. Matas and O. Chum, "Randomized RANSAC with sequential probability ratio test," in *Proc. 10th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 2, Oct. 2005, pp. 1727–1732.

[25] O. Chum and J. Matas, "Matching with PROSAC-progressive sample consensus," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 220–226.

[26] D. Capel, "An effective bail-out test for RANSAC consensus scoring," in *Proc. Brit. Mach. Vis. Conf.*, vol. 1, Sep. 2005, pp. 1–10.

[27] T. Sattler, B. Leibe, and L. Kobbelt, "SCRAMSAC: Improving RANSAC's efficiency with a spatial consistency filter," in *Proc. IEEE 12th Int. Conf. Comput. Vis. (ICCV)*, Sep. 2009, pp. 2090–2097.

[28] M. Rahman, X. Li, and X. Yin, "DL-RANSAC: An improved RANSAC with modified sampling strategy based on the likelihood," in *Proc. IEEE 4th Int. Conf. Image, Vis. Comput. (ICIVC)*, Xiamen, China, Jul. 2019, pp. 463–468.

[29] D. Barath, M. Ivashechkin, and J. Matas, "Progressive NAPSAC: Sampling from gradually growing neighborhoods," 2019, *arXiv:1906.02295*.

[30] A. Konouchine, V. A. Gaganov, and V. Veznevets, "AMLESAC: A new maximum likelihood robust estimator," in *Proc. Graphicon*, Novosibirsk, Russia, Jun. 2005, pp. 93–100.

[31] S. Choi and J.-H. Kim, "Robust regression to varying data distribution and its application to landmark-based localization," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2008, pp. 3465–3470.

[32] G. Wang, X. Sun, Y. Shang, Z. Wang, Z. Shi, and Q. Yu, "Two-view geometry estimation using RANSAC with locality preserving constraint," *IEEE Access*, vol. 8, pp. 7267–7279, 2020.

[33] K. Bulatov, V. V. Arlazarov, T. Chernov, O. Slavin, and D. Nikolaev, "Smart IDReader: Document recognition in video stream," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 6, Nov. 2017, pp. 39–44.

[34] D. K. MacArthur and C. D. Crane, "Unmanned ground vehicle state estimation using an unmanned air vehicle," in *Proc. Int. Symp. Comput. Intell. Robot. Autom.*, Jun. 2007, pp. 473–478.

[35] G. Christie, G. Warnell, and K. Kochersberger, "Semantics for UGV registration in GPS-denied environments," 2016, *arXiv:1609.04794*.

[36] A. Tamjidi and C. Ye, "A pose estimation method for unmanned ground vehicles in GPS denied environments," *Proc. SPIE*, vol. 8387, pp. 571–582, May 2012.

[37] S. Çaşka and A. Gayretli, "A survey of UAV/UGV collaborative systems," in *Proc. CIE44&IMSS*, vol. 14, 2014, pp. 453–463.

[38] W. Guilluy, L. Oudre, and A. Beghdadi, "Video stabilization: Overview, challenges and perspectives," *Signal Process., Image Commun.*, vol. 90, Jan. 2021, Art. no. 116015.

[39] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm, "USAC: A universal framework for random sample consensus," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 2022–2038, Aug. 2013.

[40] P. Marquez-Neila, J. G. Miro, J. M. Buenaposada, and L. Baumela, "Improving RANSAC for fast landmark recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2008, pp. 1–8.

[41] D. Monnin, E. Bieber, G. Schmitt, and A. Schneider, "An effective rigidity constraint for improving RANSAC in homography estimation," in *Proc. ACIVS*, Sydney, NSW, Australia, Dec. 2010, pp. 203–214.

[42] P. Márquez-Neila, J. López-Alberca, J. M. Buenaposada, and L. Baumela, "Speeding-up homography estimation in mobile devices," *J. Real-Time Image Process.*, vol. 11, no. 1, pp. 141–154, Jan. 2016.

[43] K.-L. Chung, Y.-C. Tseng, and H.-Y. Chen, "A novel and effective cooperative RANSAC image matching method using geometry histogram-based constructed reduced correspondence set," *Remote Sens.*, vol. 14, no. 14, p. 3256, Jul. 2022.

[44] O. Chum, T. Werner, and J. Matas, "Epipolar geometry estimation via RANSAC benefits from the oriented epipolar constraint," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, vol. 1, Aug. 2004, pp. 112–115.

[45] W. Xu and J. Mulligan, "Robust relative pose estimation with integrated cheirality constraint," in *Proc. 19th Int. Conf. Pattern Recognit.*, Dec. 2008, pp. 1–4.

[46] S. Korman and R. Litman, "Latent RANSAC," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6693–6702.

[47] K. Matsuzaki, Y. Uchida, S. Sakazawa, and S. Sato, "Geometric verification using semi-2D constraints for 3D object retrieval," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 2338–2343.

[48] N. Skoryukina, I. Faradjev, V. L. Arlazarov, and J. Shemiakina, "Document localization algorithms based on feature points and straight lines," in *Proc. 10th Int. Conf. Mach. Vis. (ICMV)*, Apr. 2018, pp. 385–392.

[49] N. S. Skoryukina, I. A. Faradjev, K. B. Bulatov, and V. V. Arlazarov, "Impact of geometrical restrictions in RANSAC sampling on the ID document classification," in *Proc. 12th Int. Conf. Mach. Vis. (ICMV)*, Jan. 2020, pp. 35–41.

[50] R. Iser, D. Kubus, and F. M. Wahl, "An efficient parallel approach to random sample matching (pRANSAM)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 1199–1206.

[51] A. Khalili, A. Fijany, F. Hosseini, S. Safari, and J.-G. Fontaine, "Fast parallel model estimation on the cell broadband engine," in *Proc. ISVC*, Las Vegas, NV, USA, Nov./Dec. 2010, pp. 469–480.

[52] A. Fijany and F. Hosseini, "Image processing applications on a low power highly parallel SIMD architecture," in *Proc. Aerosp. Conf.*, Mar. 2011, pp. 1–12.

[53] A. Fijany and F. Diotalevi, "A cooperative search algorithm for highly parallel implementation of RANSAC for model estimation on Tilera MIMD architecture," in *Proc. IEEE Aerosp. Conf.*, Mar. 2012, pp. 1–14.

[54] A. Hidalgo-Paniagua, M. A. Vega-Rodríguez, N. Pavón, and J. Ferruz, "A comparative study of parallel RANSAC implementations in 3D space," *Int. J. Parallel Program.*, vol. 43, pp. 703–720, Oct. 2015.

[55] D. Koguciuk, "Parallel RANSAC for point cloud registration," *Found. Comput. Decis. Sci.*, vol. 42, no. 3, pp. 203–217, Sep. 2017.

[56] M. Qi, G. Sun, and G. Chen, "Parallel and SIMD optimization of image feature extraction," *Proc. Comput. Sci.*, vol. 4, pp. 489–498, Jan. 2011.

[57] *Optimizing RANSAC with SSE*. Accessed: Aug. 4, 2023. [Online]. Available: https://opticode.ch/blog/ransac-sse/

[58] E. Limonova, A. Terekhin, D. Nikolaev, and V. V. Arlazarov, "Fast implementation of morphological filtering using ARM NEON extension," *Int. J. Appl. Eng. Res.*, vol. 11, no. 24, pp. 11675–11680, 2016.

[59] S. Martelli, R. Marzotto, A. Colombari, and V. Murino, "FPGA-based robust ellipse estimation for circular road sign detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2010, pp. 53–60.

[60] K. Dohi, Y. Hatanaka, K. Negi, Y. Shibata, and K. Oguri, "Deep-pipelined FPGA implementation of ellipse estimation for eye tracking," in *Proc. 22nd Int. Conf. Field Program. Log. Appl. (FPL)*, Aug. 2012, pp. 458–463.

[61] J. Vourvoulakis, J. Kalomiros, and J. Lygouras, "FPGA-based architecture of a real-time SIFT matcher and RANSAC algorithm for robotic vision applications," *Multimedia Tools Appl.*, vol. 77, pp. 9393–9415, Apr. 2018.

[62] Z. Zhao, F. Wang, and Q. Ni, "An FPGA-based hardware accelerator of RANSAC algorithm for matching of images feature points," in *Proc. IEEE 13th Int. Conf. ASIC (ASICON)*, Oct. 2019, pp. 1–4.

[63] L. Dung, C. Huang, and Y. Wu, "Implementation of RANSAC algorithm for feature-based image registration," *J. Comput. Commun.*, vol. 1, no. 6, pp. 46–50, Nov. 2013.

[64] E. Azimi, A. Behrad, M. B. Ghaznavi-Ghoushchi, and J. Shanbehzadeh, "Hardware architecture for projective model calculation and false match refining using random sample consensus algorithm," *J. Electron. Imag.*, vol. 25, no. 6, Dec. 2016, Art. no. 063014.

[65] L. Goshen and I. Shimshoni, "Balanced exploration and exploitation model search for efficient epipolar geometry estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1230–1242, Jun. 2008.

[66] A. S. Brahmachari and S. Sarkar, "BLOGS: Balanced local and global search for non-degenerate two view epipolar geometry," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 1685–1692.

[67] V. Fragoso, P. Sen, S. Rodriguez, and M. Turk, "EVSAC: Accelerating hypotheses generation by modeling matching scores with extreme value theory," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2472–2479.
[68] J. Zheng, W. Peng, Y. Wang, and B. Zhai, "Accelerated RANSAC for accurate image registration in aerial video surveillance," *IEEE Access*, vol. 9, pp. 36775–36790, 2021.
[69] M. Cao, F. Tang, P. Ji, and F. Ma, "Improved real-time semantic segmentation network model for crop vision navigation line detection," *Front. Plant Sci.*, vol. 13, Jun. 2022, Art. no. 898131.
[70] R. I. Hartley, "Projective reconstruction and invariants from multiple images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 10, pp. 1036–1041, Oct. 1994.
[71] J. Philip, "A non-iterative algorithm for determining all essential matrices corresponding to five point pairs," *Photogramm. Rec.*, vol. 15, no. 88, pp. 589–599, Oct. 1996.
[72] G. Nakano, "Algebraic constraint for preserving convexity of planar homography," in *Proc. Int. Conf. 3D Vis. (3DV)*, Dec. 2021, pp. 126–135.
[73] L. Moisan, P. Moulon, and P. Monasse, "Automatic homographic registration of a pair of images, with a contrario elimination of outliers," *Image Process. Line*, vol. 2, pp. 56–73, May 2012.
[74] J. Moré, "The Levenberg–Marquardt algorithm: Implementation and theory," *Numer. Anal.*, vol. 630, pp. 105–116, Dec. 1978.
[75] M. D. Vose, "A linear algorithm for generating random numbers with a given distribution," *IEEE Trans. Softw. Eng.*, vol. 17, no. 9, pp. 972–975, Sep. 1991.
[76] D. Blackman and S. Vigna, "Scrambled linear pseudorandom number generators," *ACM Trans. Math. Softw.*, vol. 47, no. 4, pp. 1–32, Dec. 2021.
[77] J. Shemiakina, A. Zhukovsky, and I. Faradjev, "The calculation of a projective transformation in the problem of planar object targeting by feature points," *Proc. SPIE*, vol. 10341, pp. 324–329, Mar. 2017.
[78] *OpenCV: USAC*. Accessed: Aug. 4, 2023. [Online]. Available: https://docs.opencv.org/4.6.0/d1/df1/md__build_master-contrib_docs-lin64_opencv_doc_tutorials_calib3d_usac.html
[79] K. B. Bulatov, E. V. Emelyanova, D. V. Tropin, N. S. Skoryukina, Y. S. Chernyshova, A. V. Sheshkus, S. A. Usilin, Z. Ming, J.-C. Burie, M. M. Luqman, and V. V. Arlazarov, "MIDV-2020: A comprehensive benchmark dataset for identity document analysis," *Comput. Opt.*, vol. 46, no. 2, pp. 252–270, 2022.
[80] V. Lepetit and P. Fua, "Towards recognizing feature points using classification trees," Swiss Federal Inst. Technol. (EPFL), Tech. Rep. IC/2004/74, 2004.
[81] J. Pilet. *YAPE Realization*. Accessed: Aug. 4, 2023. [Online]. Available: https://github.com/jpilet/polyora/blob/master/polyora/yape.cpp
[82] D. Matalov, E. Limonova, N. Skoryukina, and V. Arlazarov, "RFDoc: Memory efficient local descriptors for ID documents localization and classification," in *Proc. ICDAR*, Sep. 2021, pp. 209–224.
[83] L. Cehovin, A. Leonardis, and M. Kristan, "Visual object tracking performance measures revisited," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1261–1274, Mar. 2016.

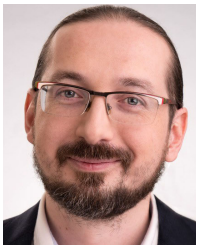
EKATERINA O. RYBAKOVA was born in Togliatti, Russia, in 1999. She received the Specialist degree from the Faculty of Mechanics and Mathematics, Lomonosov Moscow State University, in 2023. Since 2021, she has been a Programmer with Smart Engines Service LLC.

ANTON V. TRUSOV (Member, IEEE) was born in Moscow, Russia, in 1997. He received the master's degree from the Moscow Institute of Physics and Technology, in 2021. He is currently pursuing the Ph.D. degree with the Moscow Institute of Physics and Technology, National Research University. Since 2017, he has been a Programmer and a Technician with Smart Engines Service LLC, and also with FRC CSC RAS, since 2021. His research interests include device-efficient neural network implementations and computer vision algorithms.

ELENA E. LIMONOVA (Member, IEEE) was born in Dolgoprudny, Moscow, Russia, in 1993. She received the master's degree in physics, mathematics, and computer science from the Moscow Institute of Physics and Technology, in 2017, and the Ph.D. degree in computer science, in 2023. Since 2016, she has been a Programmer and a Technician with Smart Engines Service LLC. Since 2022, she has also been a Researcher with FRC CSC RAS. Her research interests include neural network compression and image recognition on mobile devices.

NATALYA S. SKORYUKINA was born in 1991. She received the Specialist degree in applied mathematics from the National University of Science and Technology "MISiS," Moscow, Russia, in 2013. Since 2014, she has been with the Federal Research Center "Computer Science and Control," Russian Academy of Sciences. Since 2016, she has also been with Smart Engines Service LLC, Moscow.

KONSTANTIN B. BULATOV (Member, IEEE) was born in Petrozavodsk, Russia, in 1991. He received the Specialist degree in applied mathematics from the National University of Science and Technology "MISiS," Moscow, Russia, in 2013, and the Ph.D. degree in computer science from the Federal Research Center "Computer Science and Control," Russian Academy of Sciences, Moscow, in 2020. Since 2014, he has been with the Federal Research Center "Computer Science and Control," Russian Academy of Sciences. Since 2016, he has been with Smart Engines Service LLC, Moscow. He has been teaching a Combinatorial Optimization Course with the Moscow Institute of Physics and Technology, State University, Moscow, since 2014. He has authored more than 20 scientific publications. His fields of study are computer vision, image processing, and document recognition systems.



DMITRY P. NIKOLAEV (Member, IEEE) was born in Moscow, Russia, in 1978. He received the master's degree in physics and the Ph.D. degree in computer science from Moscow State University, Moscow, in 2000 and 2004, respectively. Since 2007, he has been the Head of the Vision Systems Laboratory at the Institute for Information Transmission Problems, Russian Academy of Sciences, Moscow. Since 2016, he has also been a CTO of Smart Engines Service LLC, Moscow. Since 2016, he has also been an Associate Professor with the Moscow Institute of Physics and Technology, National Research University, Moscow, teaching the "Image Processing and Analysis" course. He is the author of over 220 articles and six patents. His research activities are in the area of computer vision with primary application to color image understanding.