Preliminary image registration for creation of virtual models of real objects

Dmitriy B. Volegov
Moscow Institute
of Physics and Technology
Moscow, Russia
Email: dvolegov@dgap.mipt.ru

Dmitriy V. Yurin
Moscow State University
Moscow, Russia
Email: yurin@list.ru

***Abstract*—New image processing algorithms are presented facilitating creation of virtual models of real objects: finding straight lines and rough image registration. Lines are found via Fast Hough Transform using Hartley Transform. Lines and color around them are used for rough image registration. Fuzzy logic is used to account for color.**

I. Introduction

Automatic creation of virtual models of real objects is a very challenging and actual problem. It is important to make this process cheap, fast and widely available. This means that the system should not use special equipment (laser scanners, special light sources, sensors etc), but only widely used: digital photo- or video- camera.

Currently the problem of creation of virtual models using many images is studied extensively (wide-base stereo). One of the key problems is seeking of corresponding points (point matching). Since the base is large corresponding points may be rather distant from each other and large size of search window should be used. The problem may be significantly simplified if images are registered to make corresponding points closer to each other. Besides, the fraction of false correspondences decreases.

The problem of rough image registration is addressed. In three cases there exists a planar homography (2d projective transform) which exactly maps one image to another [1]:

1) arbitrary camera position but the scene is a plane;
2) arbitrary scene but cameras have the same projection center;
3) very distant scene: when distance from the scene to the camera is much more then scene depth. This is often the case in aerospace photography.

In general case there is no homography which maps one image to another. However it is often possible to find a homography which registers images approximately.

In case of small perspective distortions methods [2], [3], [4] can be used. Method [2] uses properties of Fourier transform to deal with translation, scaling and rotation. The algorithm is able to deal with scaling coefficient up to 1.8 [2]. However the algorithm fails when perspective distortions are large. The method can be successfully applied to registration of aerospace photographs when perspective distortions are small or can be compensated using telemetry.

Method [3] is usually used for feature tracking. However it can be sometimes successfully applied to image registration problem. It is able to find an affine transform and achieves subpixel accuracy. However the algorithm works only when difference between images is small and the value of difference is defined by the half-width of Gaussian function used to calculate derivatives. Unlike [2] the algorithm does not generate hypotheses but instead falls to the nearest local minimum of the functional.

Method [4] can be applied wider. In particular, difference in scale can be up to 6 times [4]. The idea behind the method is the following: a modification of Harris detector [5] is used to deal with scale space [6]. The vector of differential invariants [7], [8] is calculated for each feature. The homography is sought via RANSAC [9]. Since RANSAC is used false correspondences (outliers) may constitute significant fraction of data. However, the results are illustrated on two images of a mountain, taken from the same point (In this case there exists a homography which exactly maps one image to another). It seems that in artificial scenes the percentage of outliers may be very large (for example, corners of windows in a multistorey building) and the algorithm may fail. Another problem is that RANSAC seeks a particular model (homography), however in general case the data do not obey such a model exactly.

Thus the problem of preliminary image registration seems to be rather actual especially if the images are taken by the photocamera and differ significantly. Proposed method allows to solve this problem in presence of straight lines on the scene. Straight lines are very stable features which survive under change of illumination conditions, camera position, projective transforms. The algorithm finds an arbitrary homography which approximately maps one image to another. Inherently the algorithm generates hypotheses about possible homographies and the best one can be chosen. Color distribution of image around lines is also taken into account.

Part II explains how straight lines on image are found. Part III deals with registration algorithm. In Part IV results are presented.

The research is done under financial support of RFFI, grants 06-01-00789, 05-07-90345, 05-07-90390

II. Finding straight lines

Widely used method for finding parametric lines on images is Hough Transform [10]. This transform is based on voting and requires extensive sampling of subsets of image points to build reliable histogram in parametric space. Due to this fact there are many ways to approximately calculate Hough transform, which are based on sampling not all subsets but only some of them [11], [12], [13]. The main inconvenience with these methods is the requirement of some heuristics to limit subsets to sample.

In [14] is shown that Hough transform for straight lines is equivalent to Radon transform. There are methods [15], [16] to calculate Radon transform with complexity $N \log N$, where N is the number of image pixels. Techniques described in [17] also can be used to find lines on images (however, they require more memory than [15], [16] and apparently time). Present works extends our previous work [16] to deal with color around lines.

Straight line is defined by two parameters: ρ - distance from origin to line, $\rho \geq 0$, ϕ - angle between line normal and abscissa axis, $\phi \in [0, 2\pi)$. Line equation in Cartesian coordinate system is:

$$\vec{x}^T\vec{k} - \rho = 0 \tag{1}$$

$$\vec{x} \equiv (x, y)^T \tag{2}$$

$$\vec{k} \equiv (\cos\phi, \sin\phi)^T \tag{3}$$

The idea behind finding lines on image (Fig.1b) is to integrate image along lines with various ρ, ϕ (Fig.1d):

$$R(\rho, \phi) = \int I(\vec{x})\delta(\vec{x}^T\vec{k} - \rho)d\vec{x} \tag{4}$$

$$\int_{-\infty}^{\infty} \delta(x)dx = 1 \tag{5}$$

$I(\vec{x})$ - image, $R(\rho, \phi)$ - Radon transform of image (called sinogram), $\delta(x)$ - Dirac delta-function. Maxima on sinogram correspond to lines on image.

In [15] is shown that sinogram of image can be calculated via Fourier transform of image. In [16] is shown that results of [15] can be extended to Hartley transform [18]. Advantage of Hartley transform over Fourier is that the former is real (whereas the latter is complex), requires less memory, data are stored more compactly and hence calculation time reduces in 3-5 times [16].

At Fig.1 example of proposed method is given. Original color image is at Fig.1a. Edge detector [19] is applied to Fig.1a. Result of edge detection is presented at Fig.1b. Sinogram of Fig.1b is at Fig.1d. Found lines are at Fig.1c. (Fig.1e is discussed in Part II-C).

A. Hartley transform

For brevity new function $\mathtt{cas}(x)$ is introduced [18]:

$$\mathtt{cas}(x) \equiv \cos(x) + \sin(x) = \sqrt{2}\sin(x + \frac{\pi}{4}) \tag{6}$$

Hartley transform $h(\vec{\zeta})$ of function $f(\vec{z})$, $\vec{z}, \vec{\zeta} \in R^n$:

$$h(\vec{\zeta}) = \int f(\vec{z})\mathtt{cas}(2\pi\vec{\zeta}^T\vec{z})d\vec{z} \tag{7}$$

Direct and inverse Hartley transform are identical:

$$f(\vec{z}) = \int h(\vec{\zeta})\mathtt{cas}(2\pi\vec{\zeta}^T\vec{z})d\vec{\zeta} \tag{8}$$

Discrete one-dimensional Hartley transform (DHT):

$$h_i = \sum_{j=0}^{L-1} f_j \mathtt{cas}\frac{2\pi ij}{L} \tag{9}$$

$$f_j = \frac{1}{L}\sum_{i=0}^{L-1} h_i \mathtt{cas}\frac{2\pi ij}{L} \tag{10}$$

$$i, j = 0 \dots L-1 \tag{11}$$

There is [18] an algorithm for calculation of Hartley transform with complexity $L \log L$, like for the Fourier transform.

Two-dimensional DHT:

$$h_{i,j} = \sum_{k,l=0}^{L-1} f_{k,l}\mathtt{cas}\frac{2\pi(ik + jl)}{L} \tag{12}$$

Unlike Fourier transform the kernel of Hartley transform is not separable:

$$\mathtt{cas}\frac{2\pi(ik + jl)}{L} \neq \mathtt{cas}\frac{2\pi ik}{L}\mathtt{cas}\frac{2\pi jl}{L}$$

Using (6):

$$\mathtt{cas}(a + b) = \frac{1}{2}\big(\mathtt{cas}a\mathtt{cas}b + \mathtt{cas}(-a)\mathtt{cas}b + \mathtt{cas}a\mathtt{cas}(-b) - \mathtt{cas}(-a)\mathtt{cas}(-b)\big) \tag{13}$$

Using (13) to calculate two-dimensional DHT one should calculate one-dimensional DHT of rows, than columns:

$$h'_{i,j} = \sum_{k,l=0}^{L-1} f_{k,l}\mathtt{cas}\frac{2\pi ik}{L}\mathtt{cas}\frac{2\pi jl}{L} \tag{14}$$

and transform elements according to the formula:

$$h_{i,j} = \frac{1}{4}\big(h'_{i,j} + h'_{p,j} + h'_{i,q} - h'_{p,q}\big) \tag{15}$$

$$p = (L - i) \mod L \tag{16}$$

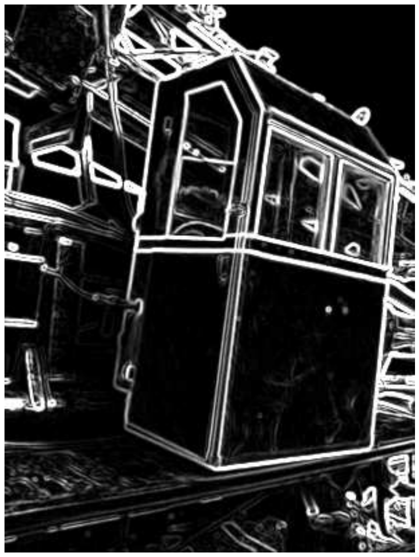
$$q = (L - j) \mod L \tag{17}$$

$a \mod b$ means the rest of integral division of a by b.

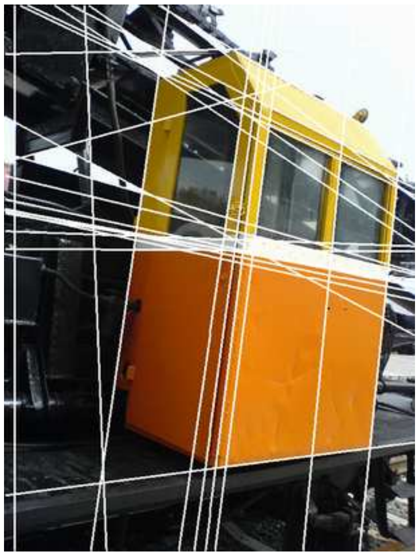
When applying (14), (15) straightforward low frequencies are concentrated at the image corners. Later on zero frequency should be moved to the center. To make this correctly one should swap opposite quadrants of image **before** (14) and swap again quadrants of Hartley transform **after** (15). This is analogous to well-known procedure of calculation Centered Discrete Fourier Transform (CDFT) but for Harltey transform.

Since implementation of Fast Hartley Transform [18] requires image size to be multiple of two one should pad image with zeroes before calculating its Hartley transform.

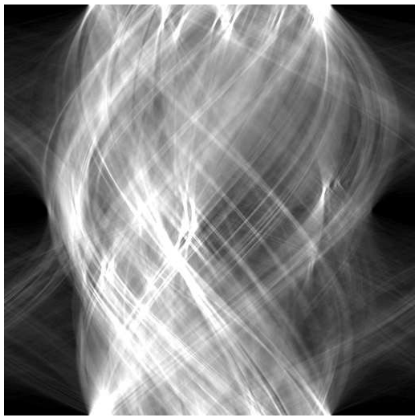
(a) Original image



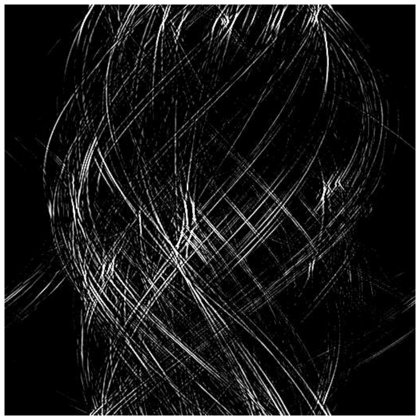
(b) Result of edge detector



(c) Found straight lines



(d) Sinogram. ρ-axis is directed downwards, ϕ-axis is directed to the right



(e) Sinogram after filtering (see Part II-C).

Fig. 1. Finding straight lines

B. Radon transform

A method for calculating Radon transform via Hartley transform (analog to [15] for Fourier) is presented below.

Theorem 1 (central section): Radon transform $R(\rho, \phi)$ of gray image $I(\vec{x})$ can be calculated as follows:

$$R(\rho, \phi) = \int P(r, \phi)\mathsf{cas}(2\pi r\rho)dr \tag{18}$$

$$P(r, \phi) = H(r\vec{k}) \tag{19}$$

$$H(\vec{\xi}) = \int I(\vec{x})\mathsf{cas}(2\pi\vec{x}^T\vec{\xi})d\vec{x} \tag{20}$$

$$\vec{x}, \vec{\xi} \in R^2 \tag{21}$$

Thus, to calculate Radon transform $R(\rho, \phi)$ of image $I(\vec{x})$ one should:

1) Calculate two-dimensional Hartley transform $H(\vec{\xi})$ of image $I(\vec{x})$ according to (20). Image $I(\vec{x})$ is a scalar image - result of edge detector [19] applied to original color image. Opposite quadrants of original image must be swaped before transform and swapped back after it.
2) Resample $H(\vec{\xi})$ to polar coordinates according to (19) (bilinear interpolation was used). Resulting image is denoted $P(r, \phi)$
3) Calculate one-dimensional Hartley transforms of rows of $P(r, \phi)$ according to (18).

Proof:
Using (7), (8) one gets:

$$R(\rho, \phi) \equiv \int S(\rho, \phi)\mathsf{cas}(2\pi r\rho)dr \tag{22}$$

$$S(r, \phi) \equiv \int R(\rho, \phi)\mathsf{cas}(2\pi r\rho)d\rho \tag{23}$$

Substituting (4) into (23) one gets:

$$S(r, \phi) = \int\int I(\vec{x})\delta(\vec{x}^T\vec{k} - \rho)\mathsf{cas}(2\pi r\rho)d\vec{x}d\rho =$$

$$= \int I(\vec{x})\mathsf{cas}(2\pi r \vec{x}^T \vec{k}) d\vec{x} = H(r\vec{k}) = P(r,\phi) \quad (24)$$

Substituting (24) into (22) on gets (18) ■

C. Finding maxima on sinogram

Finding maxima on sinogram Fig.1d is hard because of many bright regions (not points), which do not correspond to straight lines. Applying filter with large response to maxima one can make maxima more contrast (Fig.1e). Filtering is done by convolution of sinogram rows with second derivative $G_2(x)$ of Gauss function:

$$G_2(x) = \frac{d^2}{dx^2}\frac{1}{\sqrt{2\pi}\sigma}\exp\big(-\frac{x^2}{2\sigma^2}\big) \quad (25)$$

Half-width σ is the same as used to calculate derivatives in contour image.

There is the theorem [18] which says that Hartley transform of convolution of two functions, one of which is either odd or even is equal to production of their Hartley transforms (analogous to Fourier transform). Since $G_2(x)$ is even to filter sinogram one should multiply rows of $P(r,\phi)$ by Hartley transform $H_2(\xi)$ of $G_2(x)$ (theorem 1, p. 2,3):

$$H_2(\xi) = \int G_2(x)\mathsf{cas}(2\pi x\xi)dx = \xi^2 \exp\big(-\frac{\sigma^2\xi^2}{2}\big) \quad (26)$$

This result is analogous to [15] but for Harltey transform.

Filtered sinogram is presented at Fig.1e. Finding maxima after filtering is performed in two stages. At the first stage pixels of sinogram with small intensity are zeroed. Threshold is determined locally for small regions of sinogram. Nonzero pixels form the set of bright clusters. At the second stage center of mass of each cluster is calculated, point intensity is treated as its weight. Center mass of cluster determines parameters of line.

III. Image registration

In case the scene is planar (Fig.2) there is a planar homography (2d projective transform) $\mathbf{P}$ (3×3 matrix) which exactly maps one scene image to another:

$$\vec{h}_2 = \mathbf{P}\vec{h}_1 \quad (27)$$

$\vec{h}_1, \vec{h}_2$ - vectors of homogeneous coordinates of image points.

In case of general scene and camera position there is no such a homography. One can find only homography which registers images approximately. The problem is to find this homography using parameters of found lines.

A. Homography parametrisation

Planar scene (Fig.2) is viewed by two cameras with focuses in points F_1 and F_2. Normal to scene is denoted by $\vec{n}$ and directed from the first camera. Each camera has local coordinate system $\vec{i}_1$, $\vec{j}_1$, $\vec{k}_1$ and $\vec{i}_2$, $\vec{j}_2$, $\vec{k}_2$ with origins in focuses. Orts $\vec{i}_1$, $\vec{j}_1$ and $\vec{i}_2$, $\vec{j}_2$ are parallel to focal planes. Orts $\vec{k}_1$, $\vec{k}_2$ parallel to optical axes. Shift from F_1 to F_2 is denoted by $\vec{t}$. Orientation of the first camera turns into orientation of the second one by $\mathbf{R}$:

$$\mathbf{R}(\vec{i}_1 \quad \vec{j}_1 \quad \vec{k}_1) = (\vec{i}_2 \quad \vec{j}_2 \quad \vec{k}_2) \quad (28)$$

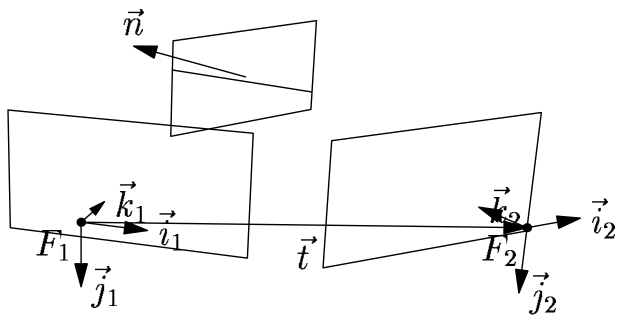


Fig. 2. Planar scene is viewed by two cameras

Distance from F_1 to scene is $d_1 \geq 0$. One can easily derive the formula, which expresses $\mathbf{P}$ through scene geometry:

$$\mathbf{P} = \mathbf{R}^T\big(\mathbf{I} - \frac{\vec{t}\vec{n}^T}{d_1}\big) \quad (29)$$

B. Lines under homography

Under homography straight line maps to another straight line. It is convenient to use another line parametrisation by unit vector $\vec{m}$:

$$\vec{m} = \frac{w}{\sqrt{w^2+\rho^2}}(\cos\phi \quad \sin\phi \quad -\frac{\rho}{w})^T, \quad |\vec{m}| = 1 \quad (30)$$

w is the camera angle resolution and has dimension pixel/radian. Geometrically $\vec{m}$ is a normal to plane passing through camera focus and line.

If the images are related by homography $\mathbf{P}$ (27) then $\vec{m}_1$, $\vec{m}_2$ (parameters of corresponding lines) are related:

$$\vec{m}_2 = \frac{\mathbf{\Lambda}\vec{m}_1}{|\mathbf{\Lambda}\vec{m}_1|} \quad (31)$$

$$\mathbf{\Lambda} \equiv (\mathbf{P}^{-1})^T \quad (32)$$

Taking into account special form (29) of matrix $\mathbf{P}$ one can use Shermon-Morris formula for matrix inversion [20]:

$$\mathbf{\Lambda} = \mathbf{R}^T\big(\mathbf{I} + \frac{\vec{n}\vec{t}^T}{d_1 - (\vec{t},\vec{n})}\big) \quad (33)$$

If only shifts are allowed which are some fraction κ of d_1:

$$|\vec{t}| \leq \kappa d_1 \quad (34)$$

then $\mathbf{\Lambda}$ spectrum is bounded:

$$\frac{1}{1+\kappa} \leq \lambda(\mathbf{\Lambda}) \leq \frac{1}{1-\kappa} \quad (35)$$

One can derive constraint (35) from (33). The constraint is used in Part III-C. The less the module of $\vec{t}$ the closer eigenvalue absolute values to the unit.

C. Finding homography

The problem is to find a homography which approximately maps set of lines on the first image $\rho_{1,i}$, $\phi_{1,i}$ to the set on the second image $\rho_{2,j}$, $\phi_{2,j}$. The problem is rather challenging because of:

- Correspondences between lines are not known.

- Numbers of found lines on images are different.
- Lines found on first image can be not found on the second one and vice versa.

1) Building the functional: Our approach is to build a functional F which depends on the parameters of lines and is defined in the parametric space (element of this space defines particular homography):

$$F = F(\mathbf{P}, \rho_{1,i}, \phi_{1,i}, \rho_{2,j}, \phi_{2,j})$$

The more lines on the first image (and the more accurate) are mapped to lines on the second image, the less is value of functional. Functional minimisation procedure is followed. Each local minimum defines hypothesis about possible homography. Each homography is applied to the first image and difference between transformed first image and second one is calculated. The best homography is the one for which integral over square of difference image is minimal.

The functional used:

$$F = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} c_{ij} F_{ij} \tag{36}$$

$$F_{ij} = -\exp\big(-\frac{|\vec{m}_{2,j} \times \Lambda \vec{m}_{1,i}|^2}{2\sigma_f^2}\big) \tag{37}$$

The functional consists of double sum. Each summand shows how line i on the first image is similar to the line j on the second one. Multiplier c_{ij} shows how similar are color distributions around lines. Its choice is described in Part III-D.

Multiplier F_{ij} shows how geometrically similar the lines are (how accurate first line is mapped to the second one). If line i on the first image is exactly mapped to line j on the second one then $F_{ij} = -1$.

Since there is generally no exact transform parameter σ_f defines boundary on line parameters when they are treated is equal. Due to (35) eigenvalue modules of $\mathbf{\Lambda}$ are approximately unit. Hence if the angle between $\mathbf{\Lambda}\vec{m}_{1,i}$ and $\vec{m}_{2,j}$ exceeds σ_f then

$$|\vec{m}_{2,j} \times \Lambda \vec{m}_{1,i}| \gg \sigma_f \quad \rightarrow \quad |F_{ij}| \approx 0 \tag{38}$$

2) Minimizing the functional: Functional (36) depends in general on eight variables (29): three angles of orthogonal matrix $\mathbf{R}$, two components of unit three-dimensional vector $\vec{n}$, three components of $\vec{t}$. This is what one expects because homography matrix is arbitrary matrix 3×3 defined up to scale.

Below the algorithm for finding functional local minima is presented:

1) A grid is defined in parametric space and functional values are calculated at the grid nodes.
2) The nodes are sorted in ascending order and nodes with minimal values are selected (some dozens).
3) Functional minimization follows using as starting points selected nodes. Minimization is done by conjugate gradient method [20].
4) Copies of local minima are removed since one can get to the same minimum from different starting points.

The subtle point in the algorithm above is to select grid step. If the step is too large there is a chance to ”slip by” the minimum. If the step is too small one has to make many unnecessary calculation due to large dimension of parametric space.

Supposing that the hypersurface in parametric space around local minimum has parabolic form, its shape (matrix of quadratic surface) was estimated. Hence reasonable grid step can be estimated. The formulas are too cumbersome to be placed here.

D. *Color around lines*

Two lines (on different images) and area around them is considered. The question is to say whether the lines could be the images of the same line in space. The answer is not a discrete value (yes/no) but a continuous one since, for example, when the boundary of some object is viewed it can have different background on different images and only colors on one side will coincide. If reflectivity depends on directions then intensities around boundaries will be different around lines. In this work fuzzy logic is used.

Advantages of fuzzy logic over empiric functionals is ”transparency” of ideas (rules). The rules are defined and formulas are derived from these rules using formal methods of fuzzy logic. It is easy to change rules and formulas change according to definitions of fuzzy logic operations.

Two color descriptors $\vec{d}_1$, $\vec{d}_2$ are associated with each line. Each color descriptor defines color distribution of image around line to the one side of it.

In the current implementation color descriptor is the average color around line (to the one side of line) and has three color components:

$$\vec{d} = (r, g, b)^T \tag{39}$$

The area to average color is defined by two conditions:

- Each point of area is distant from the line not more than 3σ, where σ is the half-width of Gauss function used to calculate derivatives.
- When moving from line along its normal $I(\vec{x})$ ($I(\vec{x})$ - result of edge detection) should not increase.

In this work fuzzy variable is defined to be a real number between zero and unit. The following definitions of fuzzy operations are used:

$$a \quad AND \quad b \quad \equiv \quad ab \tag{40}$$

$$NOT \quad a \quad \equiv \quad 1 - a \tag{41}$$

where a, b - fuzzy logic variables. Logic addition is defined through the well-known formula of set theory:

$$a \quad OR \quad b = NOT((NOT \quad a) AND (NOT \quad b)) \tag{42}$$

From (42) on gets:

$$a \quad OR \quad b = 1 - (1 - a)(1 - b) \tag{43}$$

Rules are presented below which were used to calculate how similar are the lines (similarity measure). Input is two pairs of color descriptors for two lines. Output is the fuzzy variable denoting similarity measure.

1) If descriptor brightness is small and for each descriptor in first pair there is a descriptor in the second pair with similar brightness, then the lines are similar.
2) If for each descriptor in the first pair there is a descriptor in the second pair with similar hue and saturation, then the lines are similar.
3) If at least for one descriptor in the first pair there is a descriptor in the second pair with similar hue and saturation, then the lines are similar (account for different background).

Fuzzy functions are defined corresponding to each statements in rules (like 'intensity is small', 'color is similar' etc). Than the result of each rule is calculated, their results are joined (in terms of fuzzy logic - 'aggregation') and similarity of lines is calculated ('defuzzyfication').

Coefficient c_{ij} in (36) is set to similarity between line i on the first image and line j on the second image (0 - different, 1 - equal).

IV. Results of registration

Results are presented at Fig.3, 4. One can see that after registration corresponding points get more closer than on original images. Working time t_{alg} of algorithm (finding lines + registration) for Intel Pentium 800 MHz is about 10 seconds for 1024×1024 image and about twenty lines.

Time t_{alg} is the sum of three values:

$$t_{alg} = t_{radon} + t_{proj} + t_{best} \tag{44}$$

where $t_{radon} = O(N \log N)$ - time to calculate Radon transforms of images and find lines ($t_{radon} \sim 80\% t_{alg}$). N is the number of image pixels. $t_{proj} = O(N_1 N_2)$ - time to find hypotheses about homographies ($t_{proj} \sim 3\% t_{alg}$). N_1, N_2 - numbers of lines on images. $t_{best} = O(N_{hyp} N)$ - time to select best hypothesis ($t_{best} \sim 10\% t_{alg}$). N_{hyp} - number of found hypotheses (about a dozen).

V. Conclusion

A new algorithm for image registration is presented. The algorithm uses information about straight lines on images and color distributions around them. The algorithm is adequate for complex scenes when there is no exact homography between images but only approximate one can be found. Fast Hough transform implemented via Hartley transform is used to find lines on images. A functional is build and its minimization follows to find hypotheses about possible homography.

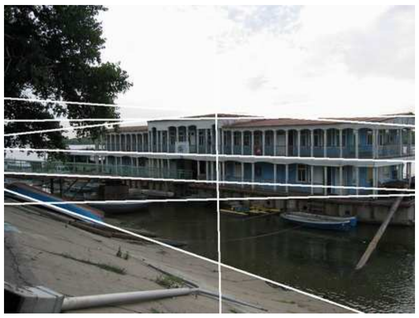
The advantages of algorithm are the following. Firstly, straight lines are very stable features on the images which survive under change of illumination conditions, camera position, projective transforms. Secondly, the algorithm works when there is a fraction of lines which are not found on the second image. Thirdly, the algorithm generates hypotheses about homographies and the best one can be chosen. The latter factor significantly increases the probability to find "the right" transform.

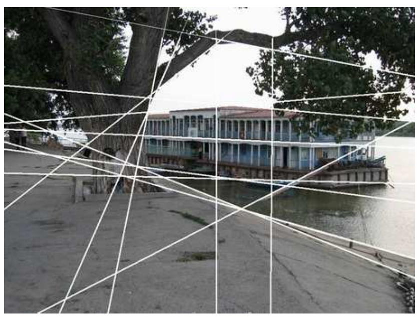
VI. Acknowledgment

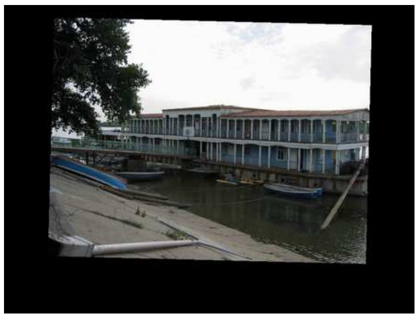
The authors thank prof. C.R. Chatwin and doctor R.C.D. Young for reprint of [15], associated professor Krylov A.S. (Dept. CMC, MSU), prof. Klimenko S.V. (Chair SIM, MIPT) for their interest, valuable discussions and support.

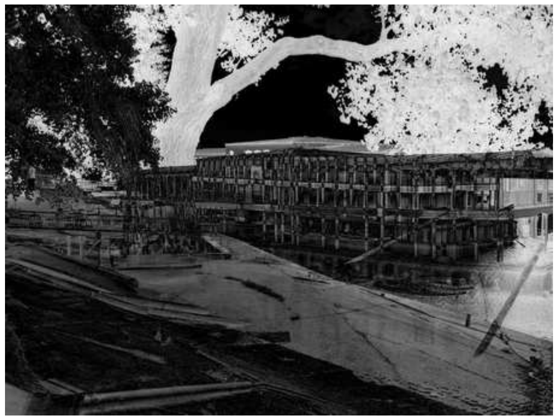
References

[1] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.
[2] B. Reddy and B. Chatterji, "An fft-based technique for translation, rotation, and scale-invariant image registration," *IEEE TRANSACTIONS ON IMAGE PROCESSING.*, vol. 5, no. 8, pp. 1266–1271, august 1996.
[3] J. Shi and C. Tomasi, "Good features to track," in *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, 1994, pp. 593 – 600.
[4] Y. Dufournaud, C. Schmid, and R. P. Horaud, "Matching images with different resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina, USA*. IEEE Computer Society Press, Juin 2000, pp. 612–618. [Online]. Available: http://perception.inrialpes.fr/Publications/2000/DSH00a
[5] C. Harris and M. Stephens, "A combined corner and edge detection," in *Proceedings of The Fourth Alvey Vision Conference*, 1988, pp. 147–151.
[6] T. Lindeberg, *Scale-Space Theory in Computer Vision*. Norwell, MA, USA: Kluwer Academic Publishers, 1993.
[7] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 530–534, May 1997. [Online]. Available: http://perception.inrialpes.fr/Publications/1997/SM97
[8] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "Scale and the differential structure of images," *Image Vision Comput.*, vol. 10, no. 6, pp. 376–388, 1992.
[9] M. A. Fischler and R. C. 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.
[10] J. Princen, J. Illingworth, and J. Kittler, "A formal definition of the hough transform: Properties and relationships," *Journal of Mathematical Imaging and Vision*, vol. 1, no. 2, pp. 153–168, July 1992.
[11] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: randomized hough transform (rht)," *Pattern Recognition Letters*, vol. 11, no. 5, pp. 331–338, 1990.
[12] N. Kiryati, Y. Eldar, and A. M. Bruckstein, "A probabilistic hough transform," *Pattern Recognition*, vol. 24, no. 4, pp. 303–316, 1991.
[13] D. Ben-Tzvi and M. Sandler, "A combinatorial hough transform," *Pattern Recognition Letters*, vol. 11, pp. 167–174, 1990.
[14] P. A. Toft, "The radon transform: Theory and implementation," Ph.D. dissertation, Technical University of Denmark, 1996.
[15] C. G. Ho, R. C. D. Young, C. D. Bradfield, and C. R. Chatwin, "A fast hough transform for the parametrisation of straight lines using fourier methods," *Real-Time Imaging*, vol. 6, no. 2, pp. 113–127, 2000.
[16] D. B. Volegov, V. V. Gusev, and D. V. Yurin, "Straight line detection on images via hartley transform. fast hough transform," in *16-th International Conference on Computer Graphics and Application GraphiCon'2006*, Novosibirsk Akademgorodok, Russia, july 2006.
[17] D. Donoho and X. Huo, "Applications of beamlets to detection and extraction of lines, curves and objects in very noisy images." [Online]. Available: citeseer.ist.psu.edu/446366.html
[18] R. N. Bracewell, *The Hartley Transform*. Oxford University Press, Inc, 1986.
[19] S. D. Zenzo, "A note on the gradient of a multi-image," *Comput. Vision Graph. Image Process.*, vol. 33, no. 1, pp. 116–125, 1986.
[20] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes*. Cambridge University Press, 1992.


(a) First image


(b) Second image


(c) Transformed first image


(d) Difference of Fig.3a, 3b

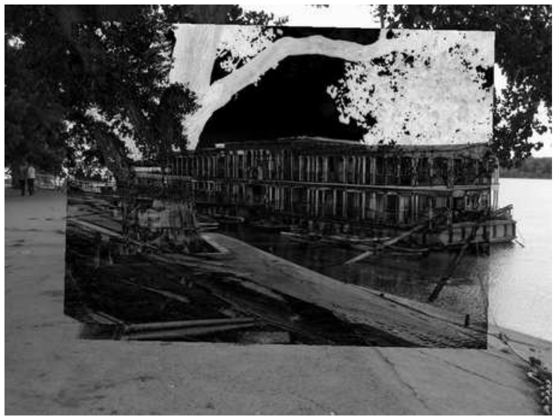
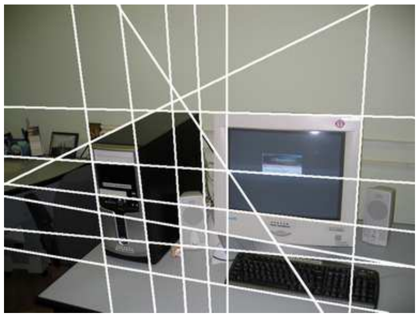
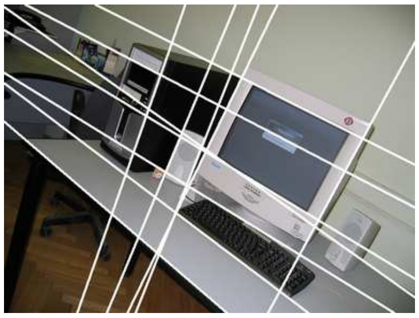
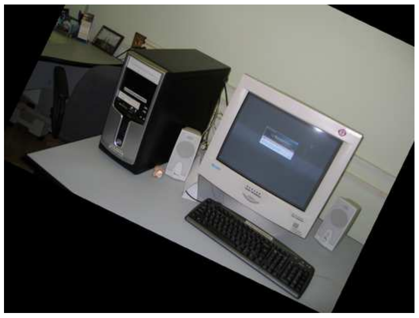
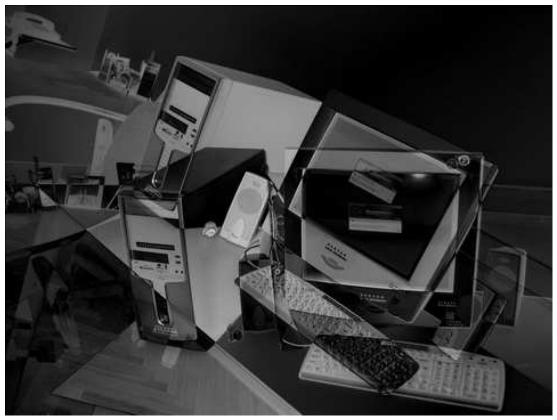

(e) Difference of Fig.3c, 3b

Fig. 3. Image registration (”The wharf”)


(a) First image


(b) Second image


(c) Transformed first image


(d) Difference of Fig.4a, 4b

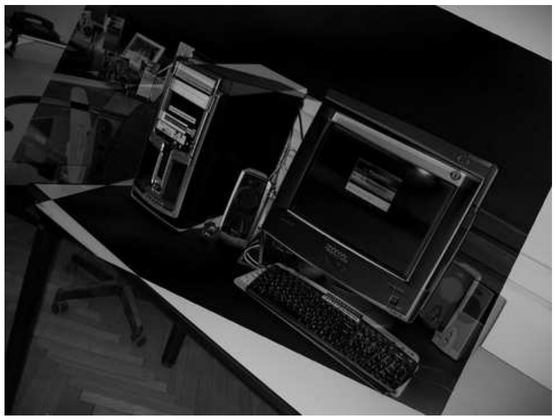

(e) Difference of Fig.4c, 4b

Fig. 4. Image registration (”The computer”)