Pseudo-real Image Sequence Generator for Optical Flow Computations

Vladimír Ulman and Jan Hubený

Centre for Biomedical Image Analysis
Masaryk University, Brno 621 00, Czech Republic
xulman@fi.muni.cz

Abstract. The availability of ground-truth flow field is crucial for quantitative evaluation of any optical flow computation method. The fidelity of test data is also important when artificially generated. Therefore, we generated an artificial flow field together with an artificial image sequence based on real-world sample image. The presented framework benefits of a two-layered approach in which user-selected foreground was locally moved and inserted into an artificially generated background. The background is visually similar to input sample image while the foreground is extracted from original and so is the same. The framework is capable of generating 2D and 3D image sequences of arbitrary length. Several examples of the version tuned to simulate real fluorescent microscope images are presented. We also provide a brief discussion.

1 Introduction

There is hardly any new method being used in image analysis that hadn't been tested thoroughly beforehand. The situation is not different in the field of optical flow computing methods. The optical flow is, according to Horn and Schunck [1], the distribution of apparent velocities of movement of brightness patterns in an image. In other words, the outcome of an optical flow method is a flow field in which a velocity vector is assigned to every voxel in the image. The vector represents movement of a given voxel. Thus, this is often used for representation of a movement in the sequence of images [2]. In particular, optical flow methods are used for analysis of such time-lapse image sequences acquired from fluorescence optical microscope [3, 4].

The most common approach to the validation of a method for computing optical flow is to compare its result to some certain flow field [5], which is commonly termed as ground-truth flow field. Unfortunately, we don't have the ground-truth information at hand when testing some method on real data. Therefore, the automatic generation of pseudo-real high-fidelity data together with correct flow field is very useful. Not only it can produce vast amount of unbiased data, it also may speed up the tuning of an existing or newly developed optical flow computation method by allowing for its immediate evaluation over close-to-real data. This may lead to a real improvement of the investigated method's precision.

B.K. Ersbøll and K.S. Pedersen (Eds.): SCIA 2007, LNCS 4522, pp. 976–985, 2007.

We propose a framework in this paper that allows for the evaluation of optical flow computing methods. The primary aim is to create a pair of new grayscale images similar to the given real-world image (e.g. Fig. 3A) together with appropriate flow field. The framework should be flexible: it should handle global cell motion together with independent local motions of selected intracellular structures which is a phenomenon often observed in the field of fluorescence microscopy. It should be accurate: the created images should perfectly resemble the real-world images as well as created flow field should describe the movements that are displayed in the image data. Since we considered 3D image as a stack of 2D images, we didn't have to utilize any 3D-to-2D projection – the flow field remained three-dimensional in this case. Hence, we referred to such a 2D or 3D flow field as to a ground-truth flow field. Last but not least, the framework should be fast and simple to use too.

The next section describes our proposed framework. It also gives the motivation to the adopted solution by means of very decent overview of some possible approaches. The third section will describe its behaviour and present few sample images coming out of the system tuned to fluorescence optical microscopy. The paper is concluded in the last section.

2 The Framework

Basically, there are just two possible approaches to obtain image sequences with ground-truth flow fields. One may inspect the real data and manually determine the flow field. Despite the bias [6] and possible errors, this usually leads to a tedious work, especially, when inspecting 3D image sequences. The other way is to generate sequences of artificial images from scratch by exploiting some prior knowledge of a generated scene. This is most often accomplished by taking 2D snapshots of a changing 3D scene [7, 5, 8]. The prior knowledge is encoded in models which control everything from the shape of objects, movements, generation of textures, noise simulation, etc. [9, 10]. This may involve a determination of many parameters as well as proper understanding of the modeled system. Once the two consecutive images are created, the information about movement between these two can be extracted from the underlying model and represented in the flow field.

We have adopted, in fact, the latter approach. Every created artificial image consisted of two layers. The bottom background layer contained an artificial background generated by the algorithm that will be described later. The background area was given by a mask image $\mathbf{M}$. The parameters of the algorithm were determined online from a real-world image which we will refer to as the sample input image $\mathbf{I}$. The foreground layer contained exact copies of regions of the sample input image. The foreground regions were defined by a mask image $\mathbf{m}$. All images had to be of the same size. We denote the value of voxel intensity in image $\mathbf{I}$ at position $\mathbf{x}$ as $\mathbf{I}(\mathbf{x})$. Similarly, the flow field $\mathbf{FF}$ holds a vector $\mathbf{FF}(\mathbf{x})$ at each position $\mathbf{x}$.

The background was subject to a global movement while the foreground was subject to global and independent local movements. The ground-truth flow field

was a composition of these. Since both background and foreground were given only by mask images, we accomplished the movements using the backward transformation technique [11]. Let us write $\mathbf{O} = \mathrm{BackT}(\mathbf{I}, \mathbf{FF})$ and say that the image $\mathbf{O}$ is the image $\mathbf{I}$ transformed according to the flow field $\mathbf{FF}$. Basically, the backward technique translates $\mathbf{I}(\mathbf{x}+\mathbf{FF}(\mathbf{x}))$ to $\mathbf{O}(\mathbf{x})$ for every $\mathbf{x}$ – voxel values move "against" the flow field. There exists a forward transformation technique too, refer to [11] for more detailed explanation. Figures 1 and 2 show the major pitfalls of both techniques. The artifacts occur when the flow field is not smooth enough. Unfortunately, that was the case owing to the local independent movements. Hence, we developed the following framework in order to avoid that. We use the notation $\mathbf{O} = \mathrm{Copy}(\mathbf{I}, \mathbf{m})$ to state that only a regions given by mask $\mathbf{m}$ are copied from $\mathbf{I}$, the rest of $\mathbf{O}$ remains untouched.

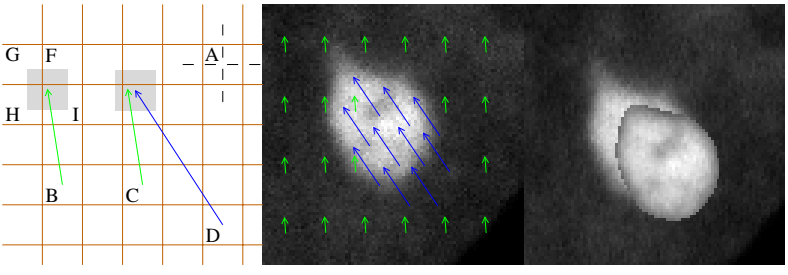

Fig. 1. The backward transformation. Left section: The grid represents voxels and their boundaries. The position of coordinate of voxel A is illustrated by dashed lines. The transformation moves value from vector's end to its beginning. In the case of real-valued vector (as is the one originating from voxel B), the moved value is the weighted sum of the nearest voxels' values with weights given by the portion of the gray area (values of voxels F, G, H and I). More vectors from distant places (as demonstrated with vectors originating from C and D) may fetch almost the same value when the flow is not smooth enough. This drawback results in the "copy" effect. Middle section: An example of input image with non-smooth flow field. Right section: A result of the backward transformation with the "copy" effect.

The framework's input was a sample input image $\mathbf{I}$, the background mask $\mathbf{M}$ and the foreground mask $\mathbf{m}$. The output would consist of images $\mathbf{I}_{1\mathrm{st}}$, $\mathbf{I}_{2\mathrm{nd}}$ and ground-truth flow field $\mathbf{gtFF}$ between $\mathbf{I}_{1\mathrm{st}}$, $\mathbf{I}_{2\mathrm{nd}}$ which denoted the first and the second image in the created sequence, respectively. It would hold: if $\forall \mathbf{x}\colon \mathbf{gtFF}(\mathbf{x})$ is an integer valued vector then $\forall \mathbf{x}\colon \mathbf{I}_{1\mathrm{st}}(\mathbf{x}) = \mathbf{I}_{2\mathrm{nd}}(\mathbf{x} + \mathbf{gtFF}(\mathbf{x}))$.

The preliminary step of the algorithm was to prepare a pool $\mathbf{R}$ of voxel intensities. Only voxels $\mathbf{x}$ satisfying $\mathbf{M}(\mathbf{x}) > 0$ and $\mathbf{m}(\mathbf{x}) = 0$ (exactly the background voxels) were copied into the pool. The mean value μ was computed within $\mathbf{R}$ since we observed that histogram of the background voxels resembled Gaussian-shaped distribution (Fig. 3B). Because of that, voxels with intensities i for which $i \notin (\mu - \sigma, \mu + k\sigma)$ were removed from the pool. We chose $\sigma = 11$ and $k = 3/2$ to fit the histogram better. This interval is shown as the white strip in Fig. 3B.

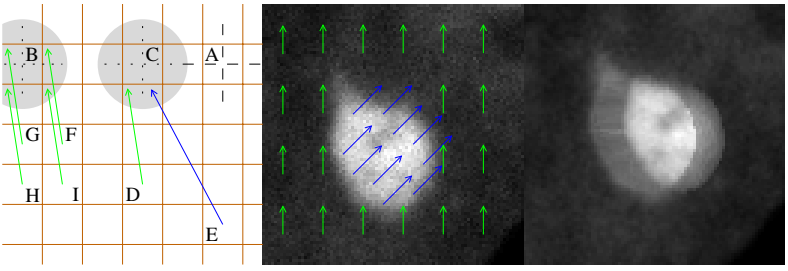

Fig. 2. The forward transformation. Left section: The grid represents voxels and their boundaries. The position of coordinate of voxel A is illustrated by dashed lines. The transformation moves value from vector's beginning to its end. In the case of real-valued vectors (as are those originating from voxels F, G, H and I), the value for particular voxel must be searched for in the close vicinity (illustrated by gray region around voxel B). The value is then weighted with weights being the distance of the nearest vectors' ends in each quadrant of the marked area. More vectors from distant places (as demonstrated with vectors originating from D and E) may end up in almost the same location when the flow is not smooth enough. This drawback results in the "averaging" effect. Middle section: An example of input image with non-smooth flow field. Right section: A result of the forward transformation with the "averaging" effect.

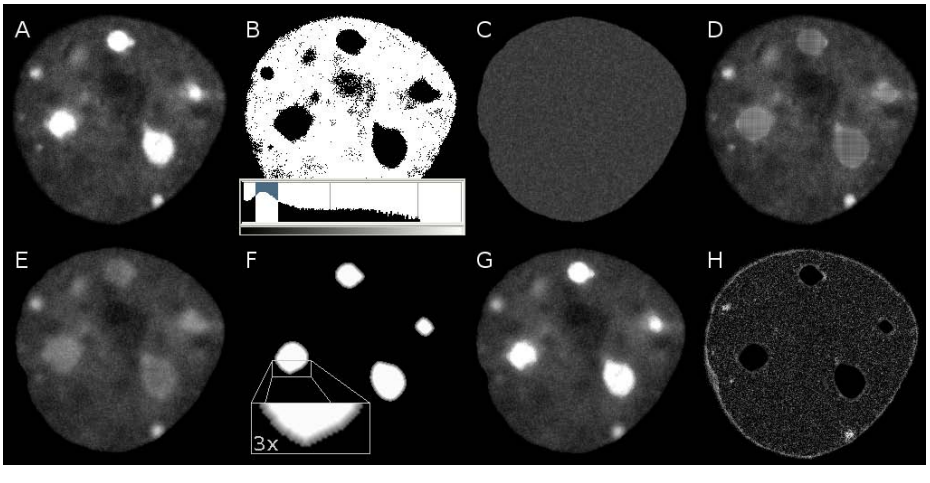

Fig. 3. Example of an image formation. A) The input image **I** of fluorescently marked HP1 proteins in the HL60 cell. B) The intensity histogram of the input image at the bottom, binary image above displays voxels with intensities in the white strip of the histogram. C) Outcome of the rand(**R**) generator. D) **fI** from (4). E) $\mathbf{I}_{2\text{nd}}$ from (5). F) The weights of the extended foreground mask,brighter intensity shows higher weights, see section 3. G) $\mathbf{I}_{2\text{nd}}$ from (6). H) The map of intensity differences between **I** and $\mathbf{I}_{2\text{nd}}$, the maximal brightness shows the value of 30. All images were enhanced for the purpose of displaying.

An artificial background was generated in two steps. First, the foreground was replaced by interpolated values. For each $\mathbf{x} = (x, y, z),\ \mathbf{m}(\mathbf{x}) > 0$ find x_1 and x_2 such that

$$x_1 = \max\left(x' < x \text{ and } \mathbf{x}' = (x', y, z) \text{ and } \mathbf{m}(\mathbf{x}') = 0\right), \tag{1}$$

$$x_2 = \min\left(x' > x \text{ and } \mathbf{x}' = (x', y, z) \text{ and } \mathbf{m}(\mathbf{x}') = 0\right) \tag{2}$$

and set $V_{x_1} = \mathbf{I}(x_1, y, z)$ and $V_{x_2} = \mathbf{I}(x_2, y, z)$. If there were no such x_1, which happens exceptionally only when a mask touches the image border, then set $V_{x_1} = \mu$ and x_1 to the leftmost coordinate in $\mathbf{I}$. The x_2 was treated in the similar fashion. The value for $\mathbf{I}(\mathbf{x})$, proportionally along x-axis, was

$$V_x = (V_{x_2} - V_{x_1}) \cdot \frac{x - x_1}{l_x + 1} + V_{x_1} \tag{3}$$

with $l_x + 1 = x_2 - x_1$. The V_y and V_z values were obtained in the similar fashion. The replacing of foreground was finished by assigning (Fig. 3D):

$$\forall \mathbf{x}\colon \mathbf{fI}(\mathbf{x}) = \begin{cases} \frac{l_y l_z V_x + l_x l_z V_y + l_x l_y V_z}{l_y l_z + l_x l_z + l_x l_y} & \text{if } \mathbf{m}(\mathbf{x}) > 0 \\ \mathbf{I}(\mathbf{x}) & \text{otherwise .} \end{cases} \tag{4}$$

Second, the new artificial background was generated. The **fI** image was convolved with separable averaging kernel. We used the filter $\frac{1}{9}(1,1,1,1,1,1,1,1,1)$ for each axis. The new background image was then computed as

$$\forall \mathbf{x}\colon \mathbf{I}_{2\text{nd}}(\mathbf{x}) = \text{rand}(\mathbf{R}) + (\mathbf{fI}(\mathbf{x}) - \mu) \tag{5}$$

where rand() is a generator of random numbers obeying uniform distribution. The effect of this term in (5) was to uniformly choose intensity values from the pool $\mathbf{R}$. This ensured the generated background to share similar statistics, including intensity fluctuations and noise. The last term in (5) enabled to display intracellular structures in the background, e.g. nucleolus as in Fig. 3E. Finally, the image was Gaussian blurred with sigma set to 0.7px.

To finish the output image $\mathbf{I}_{2\text{nd}}$, the foreground was overlaid over the artificial background:

$$\mathbf{I}_{2\text{nd}} = \text{Copy}(\mathbf{I}, \mathbf{m}) \ . \tag{6}$$

The ground-truth flow field for global movement of the whole image was created into **gtFF**. We utilized an arbitrary rotation around arbitrary centre together with arbitrary translation. The flow field was created regardless of masks $\mathbf{m}$ and $\mathbf{M}$. In fact, any flow field could have been used provided it is reasonably smooth. The images were transformed:

$$\mathbf{I}' = \text{BackT}(\mathbf{I}, \mathbf{gtFF}), \tag{7}$$

$$\mathbf{M} = \text{BackT}(\mathbf{M}, \mathbf{gtFF}), \tag{8}$$

$$\mathbf{m} = \text{BackT}(\mathbf{m}, \mathbf{gtFF}) \ . \tag{9}$$

We repeated the process of artificial background generation in the new position with $\mathbf{I}'$ instead of $\mathbf{I}$. The result was stored into $\mathbf{I}_{1\mathrm{st}}$. Note that we used the same intensity pool $\mathbf{R}$.

A random translational vector, say v_i, was assigned to each component i of the mask $\mathbf{m}$. For each i, we created flow field $\mathbf{FF}_i$ and mask image $\mathbf{m}_i$

$$\forall \mathbf{x}: \qquad \mathbf{FF}_i(\mathbf{x}) = v_i, \tag{10}$$

$$\forall \mathbf{x}: \quad \mathbf{m}_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ belongs to component } i \\ 0 & \text{otherwise .} \end{cases} \tag{11}$$

Note that $\mathbf{FF}_i$ is uniformly filled what guarantees a smooth flow field. Independent local movements are embedded into **gtFF** by computing the following equations:

$$\forall \mathbf{x}: \quad \mathbf{gFF}(\mathbf{x}) = \mathbf{gtFF}(\mathbf{x}), \tag{12}$$

$$\forall i: \qquad \mathbf{m}'_i = \mathrm{BackT}(\mathbf{m}_i, \mathbf{FF}_i), \tag{13}$$

$$\forall i: \qquad \mathbf{I}'_i = \mathrm{BackT}(\mathbf{I}', \mathbf{FF}_i), \tag{14}$$

$$\forall i: \quad \mathbf{gtFF} = \mathrm{Copy}(\mathrm{BackT}(\mathbf{gFF}, \mathbf{FF}_i), \mathbf{m}'_i), \tag{15}$$

$$\forall i: \qquad \mathbf{FF}_i = \mathrm{Copy}(\mathbf{0}, 1 - \mathbf{m}'_i), \tag{16}$$

$$\forall \mathbf{x}: \forall i: \quad \mathbf{gtFF}(\mathbf{x}) = \mathbf{gtFF}(\mathbf{x}) + \mathbf{FF}_i(\mathbf{x}) \tag{17}$$

with the following interpretations: backup **gtFF** (12), translate the patch corresponding to each component in **gtFF** (15) together with its mask (13) according to its flow field, zero the component's flow field outside of its moved mask (16) and add the result to the **gtFF** (17). Equations (15) to (17), in fact, concatenate global and local flow fields since the movement of the foreground consisted of global (9) and then local (13) movement. The image of each component was separately moved according to its flow field (14). Moving the entire image $\mathbf{I}'$ according to the final **gtFF** would produce image corrupted by the "copy" effect of non-smooth flow field.

Finally, the output image $\mathbf{I}_{1\mathrm{st}}$ was computed:

$$\forall i: \mathbf{I}_{1\mathrm{st}} = \mathrm{Copy}(\mathbf{I}'_i, \mathbf{m}'_i) \ . \tag{18}$$

The Copy() operation just overlaid the moved foreground regions over the artificially generated background. Optionally, the ground-truth flow field could be trimmed:

$$\mathbf{gtFF} = \mathrm{Copy}(\mathbf{0}, 1 - \mathbf{M}) \ . \tag{19}$$

The presented framework also allows for generation of an arbitrary long time-lapse image sequence. Due to the property of the backward transformation technique, the generation proceeds from the last image $\mathbf{I}_{n\mathrm{th}}$ of the sequence, given some $n \geq 2$, towards the first image $\mathbf{I}_{1\mathrm{st}}$. Clearly, the last image is the artificial substitute for the sample input image and so $\mathbf{I}$ can be used as a sample without any modification. For the other images in the generated sequence, $\mathbf{I}$ must be transformed to the actual position. Instead of iteratively moving the image, we

suggest to hold the flow fields that prescribe the transformations required to get $\mathbf{I}$ to the demanded position. Two flow fields should be enough. Let $\mathbf{gtFF}_{i,j},\ i<j$ denote the flow between images $\mathbf{I}_{i\text{th}}$ and $\mathbf{I}_{j\text{th}}$. For the purpose of consecutive generation of $\mathbf{I}_{k\text{th}},\ k=n-2\ldots 1$, compute

$$\forall \mathbf{x}\colon \mathbf{gtFF}_{k,n}(\mathbf{x}) = \mathbf{gtFF}_{k,k+1}(\mathbf{x}) + \mathrm{BackT}(\mathbf{gtFF}_{k+1,n}, \mathbf{gtFF}_{k,k+1})(\mathbf{x}), \quad (20)$$

$$\forall \mathbf{x}\colon\ \mathbf{glFF}_{k,n}(\mathbf{x}) = \mathbf{glFF}_{k,k+1}(\mathbf{x}) + \mathrm{BackT}(\mathbf{glFF}_{k+1,n}, \mathbf{glFF}_{k,k+1})(\mathbf{x}) \quad (21)$$

where $\mathbf{glFF}_{k,k+1}$ is the flow field corresponding to just the global component of $\mathbf{gtFF}_{k,k+1}$. Then, repeat the second part of the proposed framework from (7), as if $\mathbf{I}_{1\text{st}}$ should be created, with the following exceptions: for the background generation use $\mathbf{glFF}_{k,n}$ in (7) while for the foreground movements prepare $\mathbf{I}'$ with $\mathbf{gtFF}_{k,n}$ in (7) and set $\mathbf{gtFF} = \mathbf{gtFF}_{k,k+1}$. Also start the background generation from its second step with the convolution of $\mathbf{fI}$.

3 Results and Discussion

We implemented and tested presented framework in variants utilizing both backward and forward transformations. The framework was designed to transform images only according to the smooth flow fields. This and the definition of both transformations justify the ground-truth property of the created flow field.

Table 1. Comparisons of images $\mathbf{I}$ and $\mathbf{I}_{2\text{nd}}$. The column heading "Ext." shows the number of dilations performed on the foreground mask $\mathbf{m}$. The mask controlled the foreground extraction as well as its plain overlaying[1] or weighted merging[2] (explained in section 3). A) and B) Comparisons over two 2D images. C) Comparison over a 3D image. D) Comparison over the same 3D image, separate pools of voxel intensities were used for each 2D slice during the formation of the artificial background.

	Ext.	Corr.[1]	Avg. diff.[1]	RMS[1]	Corr.[2]	Avg. diff.[2]	RMS[2]
A	0	0.989	3.87	5.13	0.989	3.87	5.12
	1	0.989	3.80	5.03	0.989	3.85	5.05
	2	0.989	3.73	4.94	0.989	3.82	5.00
	3	0.989	3.68	4.90	0.989	3.83	4.98
B	0	0.992	2.76	3.83	0.992	2.77	3.85
	1	0.992	2.62	3.69	0.992	2.74	3.75
	2	0.993	2.41	3.46	0.992	2.62	3.58
	3	0.993	2.33	3.40	0.992	2.64	3.57
C	0	0.980	3.67	4.79	0.980	3.67	4.79
	1	0.980	3.73	4.89	0.980	3.81	4.92
	2	0.981	3.53	4.69	0.981	3.70	4.77
	3	0.981	3.42	4.59	0.981	3.66	4.72
D	0	0.982	3.15	4.16	0.982	3.16	4.17
	1	0.983	3.07	4.08	0.982	3.13	4.11
	2	0.983	3.00	4.03	0.983	3.11	4.08
	3	0.984	2.92	3.96	0.983	3.10	4.05

The masks were generated by thresholding of sample input image with manually chosen threshold. The thresholded output was considered as a starting point for advanced segmentation method [12] which produced final cell and foreground masks. The generator was tested on several different 2D real-world images and one such 3D image.

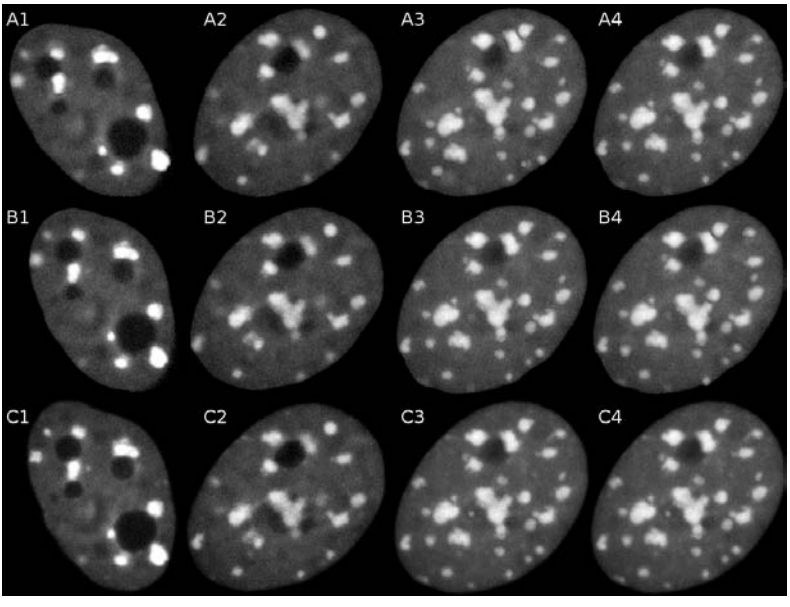

Fig. 4. Examples of generated pseudo-real 2D images. The $\mathbf{I}_{1\mathrm{st}}$, the $\mathbf{I}_{2\mathrm{nd}}$ and the sample input image $\mathbf{I}$ are shown in rows A), B) and C), respectively. Notice the similarity between rows B) and C) in columns 1), 2) and 3). Images A4) and C4) should be similar too. Foreground objects (the white spots) in each cell were subject to additional local movements. 1) An example of the cell rotated 9 degrees clock-wise around its edge. 2) An example of another cell rotated 9 degrees clock-wise around its centre. 3) An example of a similar cell with more foreground objects and with no global motion. 4) The same as 3) but the generator based on the forward transformation was used. All images were enhanced for the purpose of displaying.

All generated images were inspected. Since every generated image arose from some supplied sample image $\mathbf{I}$, we could compare $\mathbf{I}$ and $\mathbf{I}_{2\mathrm{nd}}$. For each pair, we computed the correlation coefficient (Corr.), average absolute difference (Avg. diff.) and root mean squared difference (RMS). The results are summarized in Table 1. The generator achieved minimal value of 0.98 for correlation, see Fig. 3H. This quantitatively supports our observations that generated images were very similar to their originals. A few 2D examples are shown in Fig. 4. Decent improvement was observed when artificial background of 3D images was formed in a slice-per-slice manner what is also acknowledged in Table 1.

The image $\mathbf{I}_{1\text{st}}$ could not be evaluated quantitatively for the obvious reason. Nevertheless, the ratio would be definitely worse since all moved images are blurred a little. This is a feature of both backward and forward transformations when processing flow fields containing vectors with non-integer elements. In order to make both output images appear the same, we suggest to let $\mathbf{I}_{2\text{nd}}$ image perform the translation along vector $(0.5, 0.5, 0.5)$ and modify the **gtFF** correspondingly.

Inappropriately created foreground mask may emphasize the borders of extracted foreground when inserted into artificial background. We replaced the Copy() operation in eqs. (6) and (18) by the following sequence of operations: extend the foreground mask by several dilations (the "Ext." column in Table 1), compute the distance transform (we used [13]) on the mask and threshold it (see Fig. 3F), insert the foreground according to the weights (for details refer to [14]). We generally observed visually better results with this modification. According to Table 1, just 2 dilations achieved qualitatively better results in comparison to overlaying of foreground driven by unmodified input mask $\mathbf{m}$.

We also tried the local movements mask which permitted the foreground to translate only inside this mask. This should prevent the structures from moving into the regions where there were not supposed to be, i.e. outside the cell. The masks are simple to create, for example by extending the foreground mask into demanded directions. The generated images became even more real.

We argue against further iterations of the framework to get $\mathbf{I}_{k\text{th}}$ from $\mathbf{I}_{(k+1)\text{th}}$. When proceeding towards smaller k, transforming images iteratively leads to worse quality images because of the smoothing effect (Fig. 4A) of both transformations. Our suggested solution guarantees not more than two transformations of sample input image when creating $\mathbf{I}_{k\text{th}}$ for arbitrary $k \in \langle 1, n-1 \rangle$.

We implemented the algorithm in C++. We confirm that forward variant is up to two orders of magnitude slower than backward variant for 2D images. This is mainly because of greater complexity of forward transformation in contrast to backward transformation.

4 Conclusion

We have proposed a framework for generating time-lapse pseudo-real image data. It allows for automatic synthesis of unbiased sequences of 2D and 3D images. By suppling real-world sample image we could force images in the sequence to look more realistic. The background mask of the cell and the foreground mask of selected intracellular structures were supplied too. This gave us a layered control over the regions where global and local movements should occur. The aim was to automatically generate a vast amount of data together with corresponding flow field, that we called ground-truth, in order to evaluate methods for foreground tracking as the next step. The methodology was targeted at fluorescence optical microscopy.

We have tested the framework mainly in 2D. From Table 1 we may conclude that it generated images very similar to the sample image. The foreground was a

copy from the sample image which implicitly assured its quality. The background voxels posed the same statistics since they were generated to do so. Theoretically, the presented framework has ambitions to work reliably on arbitrary data comprising of unimodal background distribution. The framework is also less sensitive to errors in the foreground segmentation. This is due to the seamless overlaying of the foreground. We also made use of local movements mask which gave us ultimate control over the foreground movements.

Acknowledgements. This work has been supported by the Ministry of Education of the Czech Republic (Grant No. MSM0021622419, LC535 and 2B06052).

References

[1] Horn, B.K.P., Schunck, B.G.: Determining optical flow. Artificial Intelligence 17, 185–203 (1981)
[2] Cédras, C., Shah, M.A.: Motion based recognition: A survey. Image and Vision Computing 13(2), 129–155 (1995)
[3] Gerlich, D., Mattes, J., Eils, R.: Quantitative motion analysis and visualization of cellular structures. Methods 29(1), 3–13 (2003)
[4] Eils, R., Athale, C.: Computational imaging in cell biology. The Journal of Cell Biology 161, 447–481 (2003)
[5] Barron, J.L., Fleet, D.J., Beauchemin, S.S.: Performance of optical flow techniques. Int. J. Comput. Vision 12(1), 43–77 (1994)
[6] Webb, D., Hamilton, M.A., Harkin, G.J., Lawrence, S., Camper, A.K., Lewandowski, Z.: Assessing technician effects when extracting quantities from microscope images. Journal of Microbiological Methods 53(1), 97–106 (2003)
[7] Galvin, B., McCane, B., Novins, K., Mason, D., Mills, S.: Recovering motion fields: An evaluation of eight optical flow algorithms. In: Proc. of the 9th British Mach. Vis. Conf (BMVC '98) 1, 195–204 (1998)
[8] Beauchemin, S.S., Barron, J.L.: The computation of optical flow. ACM Comput. Surv. 27(3), 433–466 (1995)
[9] Lehmussola, A., Selinummi, J., Ruusuvuori, P., Niemisto, A., Yli-Harja, O.: Simulating fluorescent microscope images of cell populations. In: IEEE Engineering in Medicine and Biology 27th Annual Conference pp. 3153–3156 (2005)
[10] Young, I.: Quantitative microscopy. IEEE Engineering in Medicine and Biology Magazine 15(1), 59–66 (1996)
[11] Lin, T., Barron, J.: Image reconstruction error for optical flow. In: Vision Interface. pp. 73–80 (1994)
[12] Hubený, J., Matula, P.: Fast and robust segmentation of low contrast biomedical images. In: Proceedings of the Sixth IASTED International Conference VIIP. vol. 8 (2006)
[13] Saito, T., Toriwaki, J.I.: New algorithms for Euclidean distance transformations of an n-dimensional digitized picture with applications. Pattern Recognition 27, 1551–1565 (1994)
[14] Ulman, V.: Mosaicking of high-resolution biomedical images acquired from wide-field optical microscope. In: EMBEC'05: Proceedings of the 3rd European Medical & Biological Engineering Conference. Vol. 11 (2005)