Extraction of Moving Areas in Random-Dot Animation with Parallel Computation

Satoshi Ito, Keito Uchizono & Ryosuke Morita

To cite this article: Satoshi Ito, Keito Uchizono & Ryosuke Morita (2020): Extraction of Moving Areas in Random-Dot Animation with Parallel Computation, Cybernetics and Systems, DOI: 10.1080/01969722.2020.1721728

To link to this article: https://doi.org/10.1080/01969722.2020.1721728

CYBERNETICS AND SYSTEMS: AN INTERNATIONAL JOURNAL
https://doi.org/10.1080/01969722.2020.1721728

Extraction of Moving Areas in Random-Dot Animation with Parallel Computation

Satoshi Ito[a], Keito Uchizono[b], and Ryosuke Morita[a]

[a]Faculty of Engineering, Gifu University, Gifu, Japan; [b]Graduate School of Natural Science and Technology, Gifu University, Gifu, Japan

ABSTRACT
The observation of random-dot animations requires the motion detection techniques of moving objects from the background, allowing the boundary of the moving areas to be perceived despite that each frame consists of random-dot patterns with no boundaries. The present study aims to extract the moving parts as well as its moving direction on the random-dot animations based on the local parallel computation of the pixel level. Assuming motion is captured in high frame rate, the variable range of an optical flow is restricted within a finite discretized area of pixels. Then, the system is constructed to select the suitable optical flow among the finite possibilities according to the reaction-diffusion equation dynamics, computed in a distributed manner. This process is one of the regularization methods and is described as the minimization process of the potential functional. Using the computation result of the previous frames, we attempted to reduce the computational iterations and to detect the plural objects moving at different speeds. These effects are demonstrated by computer simulations using actual random-dot animations.

KEYWORDS
Moving area extraction; optical flow; parallel processing; potential functional; random-dot animation; word

1. Introduction

Movies or animations consist of many frames displayed at several tens per second. Slight differences of the images between consecutive frames lead to the perception of the moving subjects depicted in the frames. Namely, persons, objects, or background can be discriminated within a single frame, which helps in connecting the spatial relationship between consecutive frames and in perceiving the motion.

In the animation from random-dot frames, however, any subjects cannot be extracted within the single frame because the frame is a random-dot image that consists of the salt-and-pepper pattern without any boundaries discriminating each subject from the background or the other objects. Nonetheless, we can perceive the motion of the random-dot figures in

CONTACT Satoshi Ito satoshi@gifu-u.ac.jp Faculty of Engineering, Gifu University, 1-1 Yanagido, Gifu 501-1193 Japan.

Supplemental data for this article can be accessed https://doi.org/10.1080/01969722.2020.1721728.

random-dot animations, which implies that the boundaries within each single frame are not necessary for the visual perception of the motion.

Recent image processing technologies realized by deep learnings achieved the real-time detection of objects by the segmentation (Chen et al. 2018) or the bounding box (Russakovsky et al. 2015). Such processes are executed in the single image first, and then the motions are detected by relating the same object between sequential frames. Accordingly, the random-dot animation where the subject extraction is not feasible requires a different approach from these technologies.

The random-dot image also utilized a random-dot stereogram (Thimbleby, Inglis, and Witten 1994) or a random-dot kinematogram to elucidate the neuronal or psychological mechanism of human perception (Müller, Trautmann, and Keitel 2016). Particularly, the random-dot stereogram was also investigated to model the stereoscopic computation in human vision (Marr and Poggio 1976; Kumar and Desai 1994). This paper takes the latter computational approach for motion detection.

Some computational models of the vision adopt a standard regularization (Engl, Hanke, and Neubauer 1996) that changes an ill-posed problem to a well-posed problem by defining appropriate penalty function. Among the visual process, Poggio, Torre, and Koch (1985) described an early vision process such as edge detection, optical flow, surface recognition, shape from shading and so on by the minimization of a cost function using the standard regularization. Recently, this approach was applied to the image mismatch removal between two images (Zhang et al. 2018), the elimination of noise and artifacts in the synthetic aperture radar image (Ley, D'Hondt, and Hellwich 2018), a new texture generation (Li et al. 2018), multi-frame image super-resolution (Laghrib, Hadri, and Hakim 2019) and visual tracking problem (Zhou et al. 2018). In summary, the standard regularization is utilized not only to understand the meaning of the visual process but also to improve the results of the image processing.

The regularization was introduced also to detect the moving area so far. ElTantawy and Shehata (2018) utilized the regularization for detecting the moving object in the movie. The large difference in this paper is found in that the computation is defined in each interval of the random-dot frames whereas the paper, ElTantawy and Shehata (2018) try to find it from the data combined the images from several frames together. Because the calculation restarts every frame in this paper, we will attempt to utilize the result of the former frame to detect the moving area. Furthermore, Chen et al. (2016) proposed the fast algorithm to compute the optical flow applying Split Bregman regularization. However, this does not appear effective for the random-dot animation because the gradient of the images is utilized for calculation.

The technique to separate the moving figures from the background random-dot animations was investigated using a dynamical process. Ueyama et al. (1996) introduced the Ginzburg–Landau equation, a reaction-diffusion equation whose reaction term was described as a bi-stable potential functional. This bi-stability was used to separate the figure and background. Okura, Yuasa, and Arai (2001) extended this method to detect the motion of a 3D object. Although this dynamical process can be computed in a parallel manner, the initialization of this dynamical process requires the global distribution of optical flows, which makes local computation not feasible.

To improve this defect, we report a dynamical computation model for random-dot animations, which can be computed in a parallel manner that includes the initialization process (Ito and Sasaki 2007). This paper summarizes this algorithm as a standard regularization and describes it as an optimization of a potential functional. In addition, the study not only improves the computation speed by reducing the repetition but also extends the recognition of random-dot moving objects to be applicable to different speeds, by modifying the parallel initialization process so as to utilize the result at the previous frame. The rest of this paper is structured as follows: the problem formulation is discussed in Section 2. Section 3 presents the former mathematical concepts from the distributed computation viewpoint. Section 4 explains the computation improvements. In Section 5, the simulations demonstrate the performance of our algorithm using the case studies. Section 6 provides conclusions.

2. Assumptions and Problem Description

The semiconductor technology assists in improving the graphics-processing unit computational speed, allowing in achieving the high-speed image processing in the future. Under a tremendously high processing rate, the images between two consecutive movie frames do not change significantly; if the changes occur owing to the motion, its moving distance will be suppressed at most in a few pixels. To anticipate such a high-speed processing in the future, we set the following assumptions for the random-dot animations:

- A1. All areas either stay still or move constantly by one pixel between the consecutive frames.
- A2. The direction of each moving object is limited to only four directions: left, right, upward or downward.
- A3. There are no rotations or scale-changes across all objects in the images.
- A4. All frames are free from noises.

The first assumption A1 will be removed in Section 4. Then, the problem is defined as follows:

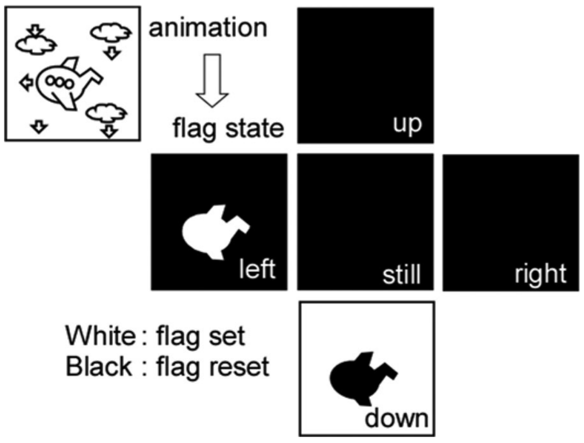

Figure 1. Concept.

- Separate the moving areas from the background and detect their moving direction in the random-dot animations.

3. Method

3.1. Concept and System Structure

We aim to achieve a distributed processing executable in a local and parallel manner between the frames. The distributed processing unit described in this paper is called an element: that is supposed to be assigned to each pixel of the animation frame. Computation is required at the processing element and is executable between the frames.

The element possesses five states, which are called flags in this study. Each flag denotes the direction to which a pixel is moving or staying still, such as left flag, right flag, up flag, down flag, and still flag. Then, we construct the five fictive planes, left plane, right plane, up plane, down plane, and still plane, by aligning each flag while keeping the topological relation of the original elements or pixels.

The moving areas and their directions are discriminated using the flags as explained in Figure 1. In the moving left area, which is the pixels of the airplane, all flags position are selected on the left plane. Given that the background moves downward, all flags on the down plane are selected except the moving object (the airplane).

This flag selection is equivalent to the optical flow calculation based on a way where we restrict it to five possibilities and decide the adequate one of five considering the spatial movement between frames and the local information at the neighboring pixels.

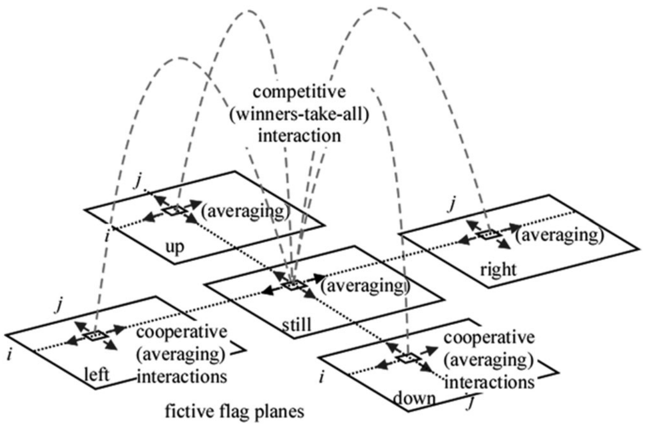

Figure 2. Two kinds of interactions among or within the fictive flag planes.

Next, we attempt to define the processing dynamics to achieve the final state as shown in Figure 1.

3.2. Dynamics

The flags take a continuous value between 0 and 1, though it should be originally a discrete value indicating yes or no for the question: whether the pixel the flag belonging to is moving to the direction assigned to the flag. Let us denote the flag state as $f_k = f_k\ (x,\ y,\ t) \in [0,\ 1]$, where $(x,\ y)$ is the position in the image, t is the calculation time, and $k = 1,\ \ldots,\ m$ discriminates the fictive plane. m indicates the number of flags in the processing elements, and $m = 5$ according to the second assumption A2.

We define the flag dynamics considering the following properties.

- P1. For every five flags in each element, one flag takes the value of one, whereas all others take the value of zero.
- P2. Almost all flags take the same value as their neighbors in each flag plane.

Property P1 is obvious given that the moving direction of the pixel is unique. Contrarily, property P2 implies that the flags are different from their neighbors only in the boundary of the moving objects. To develop both the P1 and P2 properties, two kinds of interactions are introduced as shown in Figure 2.

For the first property, the following competitive dynamics is defined:

$$\tau_1 \frac{df_k}{dt} = -\frac{\partial V_1}{\partial f_k} \tag{1}$$

$$V_1 = \iint_\Omega \left[-\frac{1}{2} \sum_{\ell=1}^{m} f_\ell^2 + \frac{1}{4} \sum_{\ell=1}^{m} \sum_{\ell' \neq \ell}^{m} f_\ell^2 f_{\ell'}^2 + \frac{1}{4} \left(\sum_{\ell=1}^{m} f_\ell^2 \right)^2 \right] dxdy \tag{2}$$

Here, τ_1 controls the speed of the dynamics, and Ω denotes the whole frame area. These dynamics work within each processing elements, i.e., all flags located in the same position in each flag plane where a winner-take-all property results in all zero states except the sole flag that takes the value of one (Fuchs and Haken 1988).

In the second property, the averaging or smoothing dynamics are introduced within the respective flag plane:

$$\tau_2 \frac{df_k}{dt} = -\frac{\partial V_2}{\partial f_k} \tag{3}$$

$$V_2 = \frac{1}{2} \sum_{k=1}^{n} \iint_\Omega (\nabla f_k)^2 dxdy \tag{4}$$

where τ_2 controls the speed of the dynamics and $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right)$.

Finally, the entire dynamics are written as an optimization process by connecting these interactions, which minimizes the potential functional $V = V_1 + K_D V_2$:

$$\begin{aligned} \tau \frac{df_k}{dt} &= -\frac{\partial V}{\partial f_k} \\ &= f_k - \sum_{\ell \neq k}^{m} f_\ell^2 f_k - \sum_{\ell=1}^{m} f_\ell^2 f_k + K_D (\nabla f_k) \end{aligned} \tag{5}$$

Here, $\tau = \tau_1$, and $K_D = \tau_1 / \tau_2$ that adjusts the time scale of the above two dynamics.

The image data are digitalized in the computation: (x, y) is replaced with (i, j) where i and j are integers. Then, the dynamics from V_2 can be replaced as the digital Laplacian filter.

3.3. Initialization

The initialization of each flag is based on the correlation of the local area between consecutive frames. Let $p_{i,j}^{(n)}$ denotes the pixel value at the position (i, j) in the n-th frame. Then, the image piece vector $v_{i,j}^{(n)}$ consisting of the pixel values around the position (i, j) is defined in the n-th frame:

$$v_{i,j}^{(n)} = \left[p_{i-1,j-1}^{(n)} \; p_{i,j-1}^{(n)} \; p_{i+1,j-1}^{(n)} \; p_{i-1,j}^{(n)} \; p_{i,j}^{(n)} \; p_{i+1,j}^{(n)} \; p_{i-1,j+1}^{(n)} \; p_{i,j+1}^{(n)} \; p_{i+1,j+1}^{(n)} \right] \tag{6}$$

The flags are initialized using the local correlation coefficient $r_k^{(n)}(i, j)$ as

$$f_k^{(n)}(i,\ j,0) = r_k^{(n)}(i,\ j) \tag{7}$$

Where

$$r_k^{(n)}(i,\ j) = \frac{v_{i,j}^{(n-1)} \cdot z}{\left\| v_{i,j}^{(n-1)} \right\| \|z\|} \tag{8}$$

$$z = \begin{cases} v_{i,j-1}^{(n)} & k = \text{up} \\ v_{i-1,j}^{(n)} & k = \text{left} \\ v_{i,j}^{(n)} & k = \text{still} \\ v_{i+1,j}^{(n)} & k = \text{right} \\ v_{i,j+1}^{(n)} & k = \text{down} \end{cases} \tag{9}$$

Here, the operation "·" denotes the inner product, and $||\cdot||$ is the L_2-norm. The above operation allows us to detects one of the five movements by finding the largest within each element, because (8) is a correlation coefficient between the image piece at (i, j) and the 1-dot shifted point in the next frame. The positiveness of all the elements of the image piece vector denotes $0 \le |f_k(i, j, 0)| \le 1$.

The initialization (7) and the dynamics (5) are calculated locally since the flag values other than those of the neighboring pixels are not required. Dynamics (5) is a minimization process of the potential functional V for the initial state given by (7).

3.4. An Example

A simple simulation was performed to demonstrate the result of the dynamical process. This six-frame example comprises of a random-dot square with 40 by 40 pixels rightward moving one pixel per frame on the random-dot background. The dynamics are computed using Euler's method programmed in Python 3.6 with Keras. The parameters are set as $K_D = 0.5$ and $\tau = 5$. The number of computational iterations between frames is set at 80.

The simulation results are shown in Figure 3. The top row shows a series of 6 frames, and the final state of each plane after 80 iterations are depicted in the following order: up plane, left plane, still plane, right plane, and down plane. The white color represents the flag state 1, whereas the black color represents the flag state 0. The right plane displays the square moving to the right. The background is detected as the white-tone area in the still plane.

The time course of each plane between the 3rd and 4th frames is depicted in Figure 4. The diffusion process averages each flag plane to

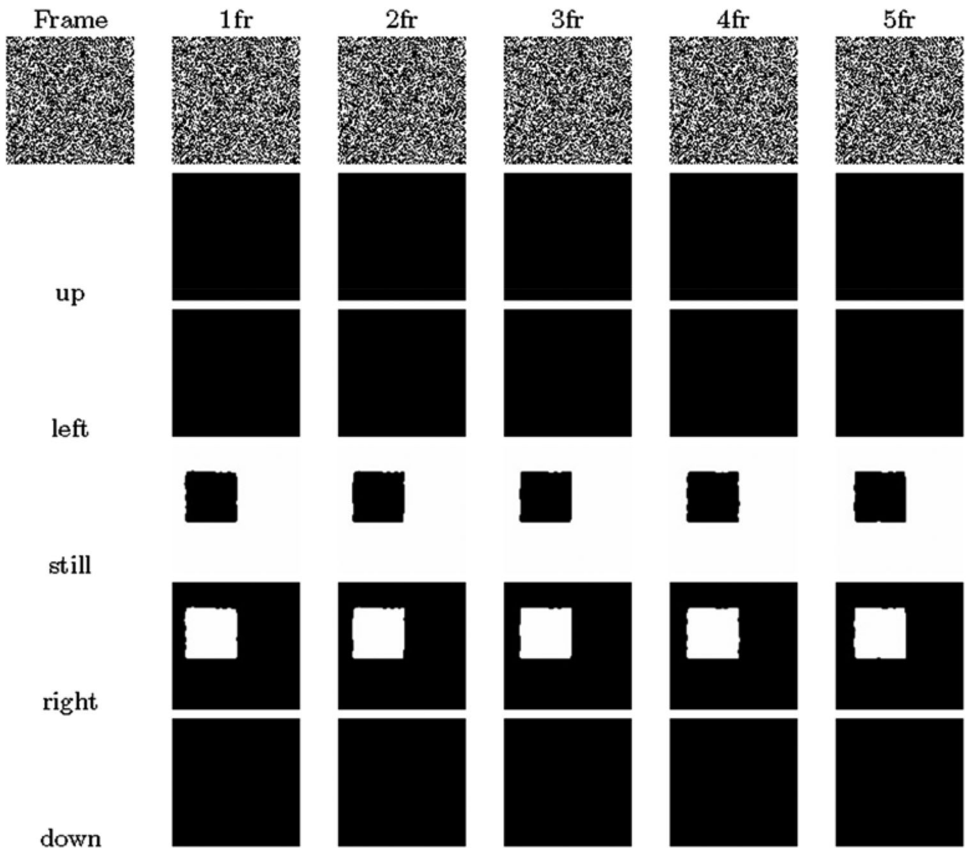

Figure 3. Final states of each fictive plane for the first 5 frames. Black denotes 0, and white denotes 1.

remove the noises so as to satisfy P1, and reaction term works to discriminate the motion direction solely at each pixel to satisfy P2.

4. Improvements

4.1. Problems to Be Solved

The separation of moving parts from the background is achieved in form of flag selection process on each fictive flag plane, as shown in the previous section. However, this separation requires some computations, such as 80 iterations in the above example, between the consecutive frames. The computations must be completed in a short time between the frames. Therefore, a reduction of the computational time (iterations) is a crucial issue.

In the previous section, in addition to the computational time, the movement speed is limited to only 1-pixel per frame under the assumption A1.

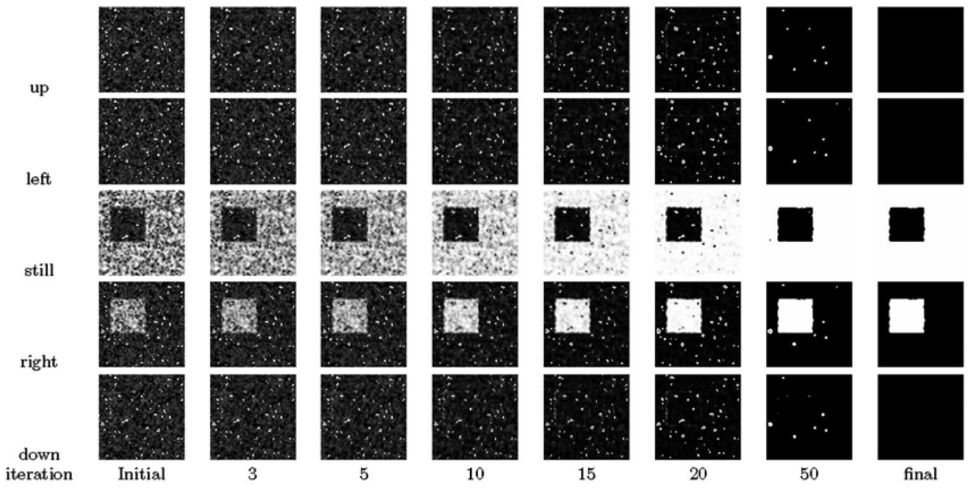

Figure 4. Time evaluations of each plane between the 3rd and 4th frame.

The animation normally contains several objects moving at different speeds. The mechanism for coping with various speeds for multiple objects is another important problem.

4.2. Solutions

4.2.1. Assumption and Idea

To solve the speed of the moving object, we loosen the assumption A1 as follows:

- A1'. All areas are either staying still or moving constantly. The constant speed is one pixel in several frames.

This assumption implies that the objects do not move more than two pixels in one frame.

The process of the flag selection, proposed in the previous section, is equivalent to the calculation of the optical flow. If the motion direction is constant, the optical flows are the same in almost all images except the boundaries of the moving objects. Suppose a square area is moving to the right and the background is moving upward, as illustrated in Figure 5(a); then, the optical flow on the background faces upward as denoted by the dark area in Figure 5(b), whereas those of the square area face rightward as denoted by the white area. However, the position of the white area shifts to the right in the next frame given that the square area has moved to the right. By comparing the optical flows in two consecutive frames, it is revealed that most of the optical flows do not change. Only the boundary area, as shown in Figure 5(c), has different optical flow by the influence of the movement of the square area.

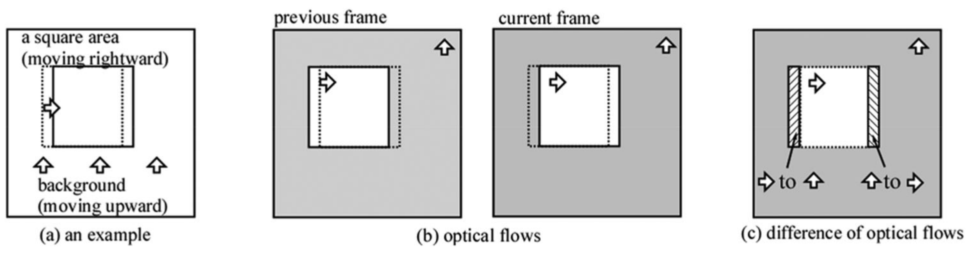

Figure 5. Optical flow differences between two consecutive frames.

This finding implies the possibility of reducing the computations by sustaining the calculation of the optical flow, i.e., the flag selection, through the several frames as illustrated in Figure 6. Suppose a flag value converges to 0 after a long iteration, as shown in Figure 6(a); if the iterations of the calculations are reduced, the result obtained will be a value in the transient state due to the short of the iterations. In addition, this flag value is initialized when the next frame starts, as depicted in Figure 6(b). Not so, we should continue the flag calculation without initialization almost at all areas, because the optical flow does not change as seen in Figure 5(c). As a result of this, not the transient but the stationary value can be obtained in the small iterations as shown in Figure 6(c), although it takes several frames.

4.3. Formulation

4.3.1. Reduction of Computation

Based on our discussion in the previous section, we initialize the flag according to the local correlation only if it belongs to the boundary of the moving parts; otherwise, continue the calculation without initialization.

The boundary pixels are discriminated by comparing the local correlation coefficient $r_k^{(n)}(i', j')$ with those of the previous frame $r_k^{(n-1)}(i', j')$. Actually, two correlation coefficients became the same besides the boundary if the speed is constant. Based on this discrimination rule, we redefine the initialization as follows:

$$f_k^{(n)}(i,\ j, 0) = \begin{cases} r_k^{(n)}(i,\ j) & r_k^{(n)}(i,\ j) \neq r_k^{(n-1)}(i',\ j') \\ f_k^{(n-1)}(i,\ j, t_{end}) & otherwise \end{cases} \tag{10}$$

where t_{end} denotes the end time of the computation of dynamics (5). (*i*', *j*') should be selected depending on the maximal flag after the computation result of the previous frame: If the still flag has the maximum value among five within the element, this pixel can be regarded to stay still and thus the correlation coefficient should refer to the same position, $(i', j') = (i, j)$. If, for example, the up flag has the maximum value, this pixel can be regarded to come from the downward direction and thus the correlation coefficient

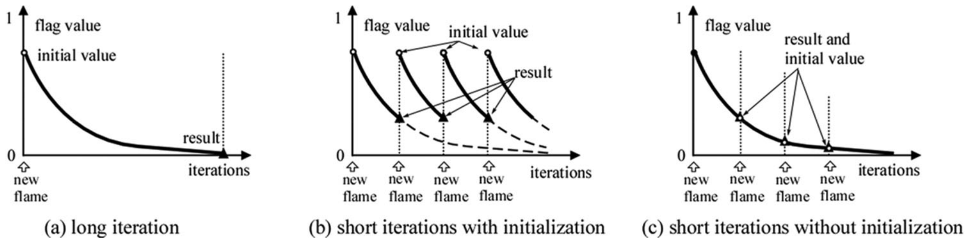

Figure 6. The flag value evolution with the iteration times and initialization.

should refer to one-pixel below, $(i', j') = (i, j+1)$. In the same way, the maximal value is taken at the left, right, and down flag, (i', j') is defined as $(i+1, j)$, $(i-1, j)$ and $(i, j-1)$, respectively.

4.3.2. Moving Objects at Different Speeds

Next, let us consider a case where the objects are moving at different speeds. The assumption A1' means that these objects move one pixel in a few frames, in other words, at most one pixel per frame. In such moving objects, the flag corresponding to its direction should keep 1. However, the objects only move every few frames, e.g., move, stop, stop, move, stop, stop,… and thus $r_k^{(n)}(i,\ j)$ becomes different from $r_k^{(n-1)}(i,\ j)$: (10) will reset the flags in waiting for the next motion

In fact, there are two cases where $r_k^{(n)}(i,\ j) = r_k^{(n-1)}(i,\ j)$ does not hold. The first case corresponds to the shaded area in Figure 5(c), as mentioned in the above section. However, when the slowly moving objects stop their motion until the next motions, the white areas in Figure 5(c) do not satisfy $r_k^{(n)}(i,\ j) = r_k^{(n-1)}(i,\ j)$. These two cases can be discriminable, because the latter is limited to only the case where the object ceases the motion just at this frame, implying that, in the previous frames, the flag other than the still flag should have taken maximum, i.e., $r_{still}^{(n)} = 1$ and argmax $f_k^{(n-1)}(t_{end}) \neq$ still.

Therefore, to utilize this historical information, i.e., the result of the previous frame, we redefine the initialization using the following weighted summation:

$$f_k^{(n)}(i,\ j, 0) = \begin{cases} \textit{if } r_k^{(n)}(i,\ j) \neq r_k^{(n-1)}(i',\ j'): \\ r_k^{(n)}(i,\ j) \\ \textit{otherwise}: \\ \textit{if } r_{still}^{(n)}(i,\ j, t_{end}) = 1 \textit{ and } \operatorname{argmax}_k f_k^{(n-1)}(t_{end}) \neq \text{still}: \\ \quad \alpha f_k^{(n-1)}(i,\ j, 0) + (1-\alpha) r_k^{(n)}(i,\ j) \\ \quad \textit{otherwise}: \\ f_k^{(n-1)}(i,\ j, t_{end}) \end{cases} \tag{11}$$

where α is a parameter that controls its weight at the initialization. When α is set to 0, a completely new initial value is set based on the correlation coefficient as is the same as the case $r_k^{(n)}(i, j) \neq r_k^{(n-1)}(i', j')$. When α is set to 1, the flag is set the initial value in the previous frame again assuming that the motion certainly occurred at the previous frame; This definition continues to set this initial value recursively even if the objects stay during the several frames.

4.3.3. Initial Value Mapping

Another improvement we can do to shorten the computation time is to manipulate the flag initial values defined as the correlation coefficient (8). The reaction terms make only the largest flag to 1 as well as all the other small ones to 0 in each element. This implies that, if we want to fasten the convergence, we should set the larger initial value if the correlation is high and the smaller initial value if the correlation is low.

The correlation is important information for estimating the motion and therefore its largeness relationship should be maintained. Thus, we map it with the monotonically increasing function $g(x)$ $(x \in [0, 1])$ with $g(0) = 0$ and $g(1) = 1$ as follows:

$$r_k^{(n)}(i, j) = g\left(\frac{v_{i,j}^{(n-1)} \cdot z}{\left|\left|v_{i,j}^{(n-1)}\right|\right| ||z||}\right) \tag{12}$$

For example, the sigmoid function with high slope will enable us to set 0 or 1 if the appropriate threshold is selected: Such initial values make the winner-take-all dynamics converge rapidly. The original definition (8) corresponds to the case where the identity function is selected, i.e., $g(x) = x$.

5. Simulations

5.1. Purposes and Conditions

Incorporating the improvements, simulations were conducted to confirm whether:

- The reduced iteration of the computation can produce proper results,
- The change of moving direction can be detected rapidly even on the moving background, and
- The two moving parts with different speeds can be detected correctly.

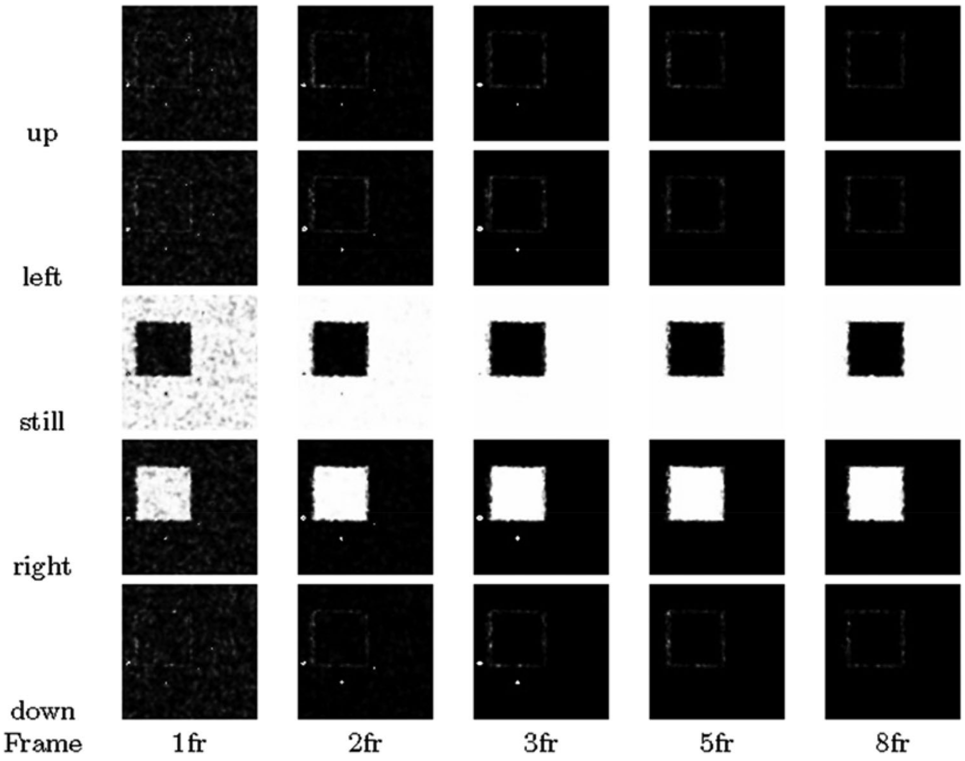

Figure 7. Final states of each fictive plane for the first 8 frames obtained by the improved algorithm.

As in Section 3.4, the dynamics are computed using Euler's method programmed in Python 3.6 with Keras, and the parameters are also set as $K_D = 0.5$, $\tau = 5$ and $\alpha = 0.75$. $g(x) = x^2$ is adopted to all the simulations in this section. The number of computational iterations between frames is reduced to 10 times.

5.2. Results

5.2.1. Reduction of Iterative Computation

The first example adopts exactly the same random-dot animation presented in Section 3.4. However, the iteration of the computation was set to 10, 1/8 of Section 3.4.

The calculation results between each frame are shown in Figure 7. Although the flags had not converged to 0 or 1 in the 1st frame owing to the short iteration, they converged over the 8th frame without initialization and the final values are similar to those in Figure 3. Figure 8 shows the time course of the flag plane covering 8 frames. As the flag initialization has been restricted at the boundary of the moving part, almost all flags are

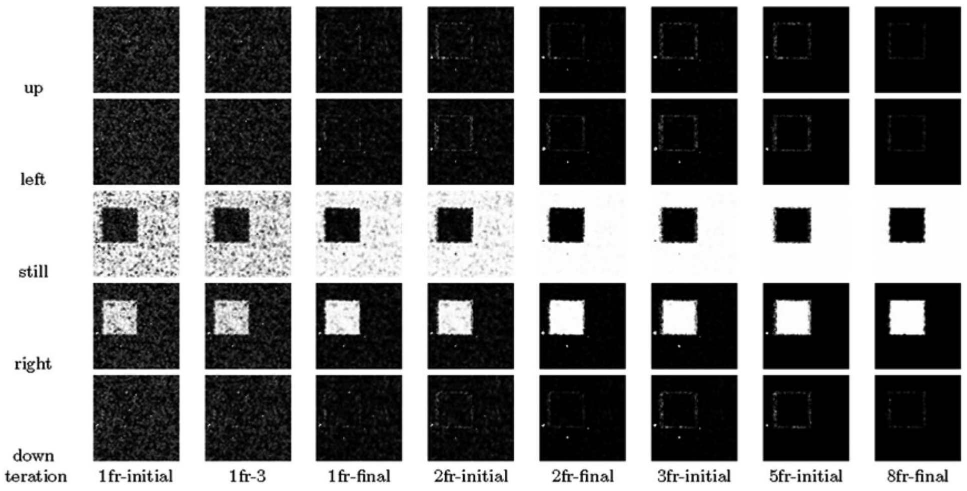

Figure 8. Time evaluations of each plane between the 1st and 8th frame obtained by the improved algorithm.

changing continuously and converging to 0 or 1 in the whole number of iterations calculated in the 8th frames.

5.2.2. Detection of the Moving Direction Change

To reduce the iteration between frames, our improved method selects whether the flag is initialized or not at the start of each frame. Then, inappropriate skips of the initializations may not immediately detect change of the moving direction if the initialization should be done there. Actually, in such cases, the skip of initialization requires more computation time to detect the correct moving direction. Thus, the next simulation examines how quickly the change of the moving direction can be detected, using the following random-dot animation: the square stays on the background moving downward until the 20th frames. From the 21st frame, the background changes the motion to the left while the square start moving upward simultaneously.

The result of the flag planes from 20th to 24th frame is shown in Figure 9. The background that moves downward is detected in the down plane, and the white square can be seen in the still plane as we expected at the 20th frame result. Consuming only a couple of frames, the motion change of both square and background are almost detected, although the complete detection seems to need some more frames.

5.2.3. Detection of Two Areas Moving at Different Speeds

Our improvement enables the system to detect the slower speed motion than 1-pixel per frame. Finally, we tested an animation where two squares

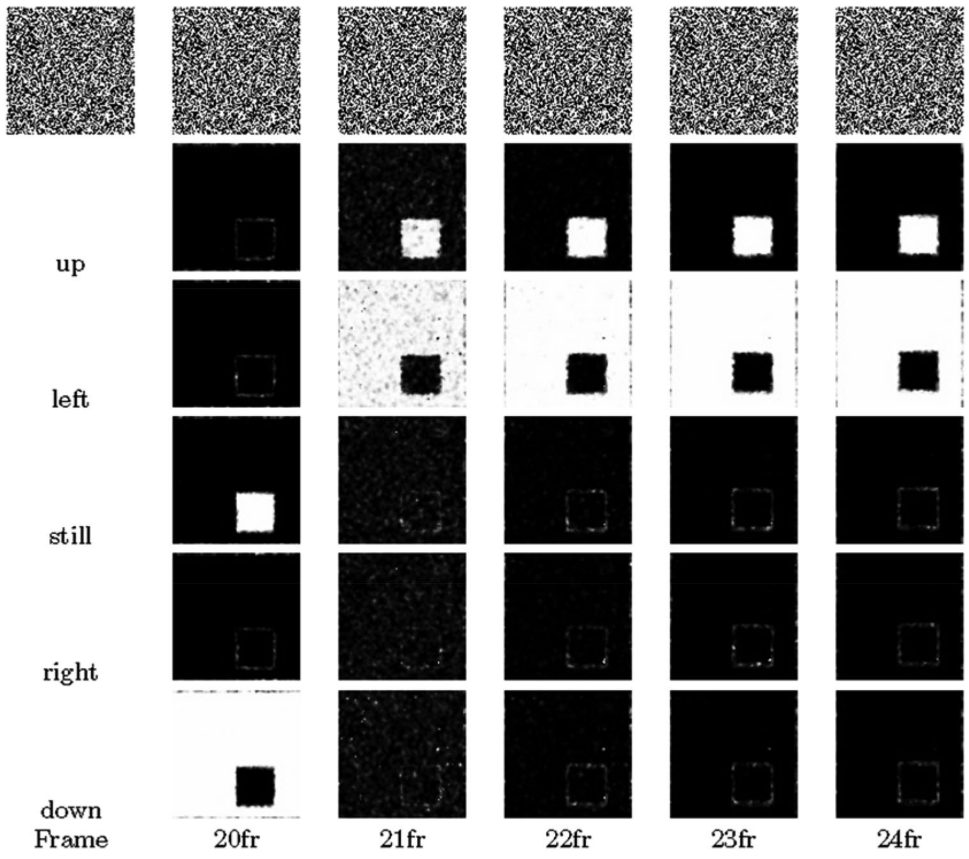

Figure 9. Final states of each fictive plane from the 20th to 24th frame.

are moving at various speeds: the speed of the smaller square moving to the right is 1-pixel per frame, while the speed of the larger part moving upward is 1-pixel per three frames.

The result is shown in Figure 10. Although the slow larger part stops every 2 frame, the upward motion is detected in the up flag planes successfully. There, white color is getting dark as the temporal stop continues. This darkness will be available to know the speed of the slow moving objects.

6. Conclusion

In this study, we proposed an image processing system that can abstract moving parts from random-dot animation developed from the concepts in our previous study (Ito and Sasaki 2007). Restricting the movement of the animation objects to the four directions, namely, up, down, left, and right, the constant speed to 1 pixel per frame, the constructed system discriminated the moving parts to each direction from the background. Then, five

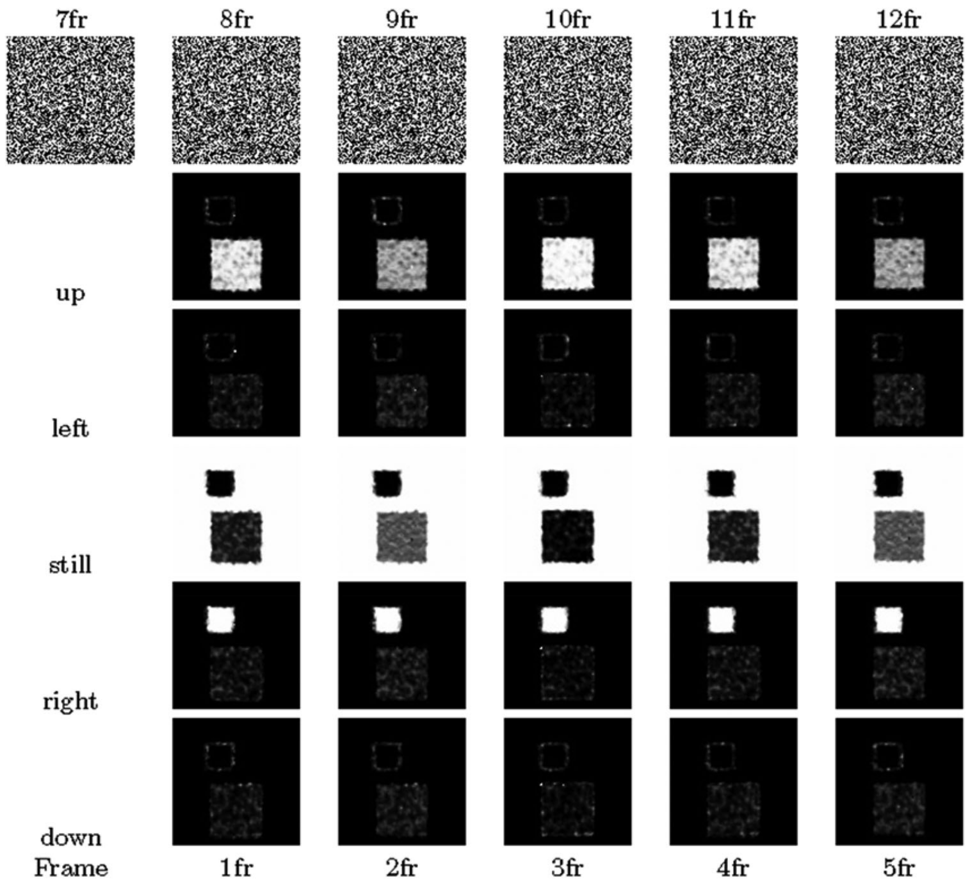

Figure 10. Final states of each fictive plane for the first 5 frames.

flags that discriminated the five possible movements, namely, up, down, right, left, and still, were introduced for each pixel of the animation frame. The flags are evolved between the frames according to the dynamics classified to the reaction-diffusion equations, which select only one of the five flags from each pixel of the smoothed neighboring pixels. These dynamics are expressed as the minimization process of a potential functional, and its calculation is executed in the distributed and parallel manner. In other words, among various methods for optical flow computation (Fortun, Bouthemy, and Kervrann 2015), the feature of this system is found in the computation based on the regularization process.

Next, based on the fact that the optical flows vary only at the boundary of the objects, the number of the iterative computation between the frames was reduced by skipping the flag initialization at the start of each frame. This skip allowed us to continuously calculate the flag dynamics over the several frames. This was the main reason for the reduction of the computational iterations, though some frames are required to obtain the final calculation result. Furthermore, utilizing the latest result of the flag calculation, this system was improved to detect slower movements other than 1 pixel per frame. The effects were confirmed by computer simulations using suitable random-dot animations.

Conflict of interest

No potential conflict of interest was reported by the authors.

References

Chen, J., Z. Cai, J. Lai, and X. Xie. 2016. Fast optical flow estimation based on the split Bregman method. *IEEE Transactions on Circuits and Systems for Video Technology* 28 (3):664–78. doi:10.1109/TCSVT.2016.2615324.

Chen, L.-C., Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. https://arxiv.org/abs/1802.02611.

ElTantawy, A., and M. S. Shehata. 2018. KRMARO: Aerial detection of small-size ground moving objects using kinematic regularization and matrix rank optimization. *IEEE Transactions on Circuits and Systems for Video Technology* 29 (6):1672–86. doi:10.1109/TCSVT.2018.2843761.

Engl, H. W., M. Hanke, and A. Neubauer. 2000. *Regularization of inverse problems. Mathematics and Its Applications*. vol. 375. Dordrecht: Kluwer Academic Publishers.

Fortun, D., P. Bouthemy, and C. Kervrann. 2015. Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding* 134:1–21. doi:10.1016/j.cviu.2015.02.008.

Fuchs, A., and H. Haken. 1988. Pattern recognition and associative memory as dynamical processes in a synergetic system. *Biological Cybernetics* 60 (1):17–22. doi:10.1007/BF00204703.

Ito, S., and M. Sasaki. 2007. Parallel processing of figure-ground separation in random dot kinematogram with optical flow discretization. Paper presented at the SICE 2007 Annual Conference, Takamatsu, Japan, September 17–20.

Kumar, K. S., and U. Desai. 1994. New algorithms for 3D surface description from binocular stereo using integration. *Journal of the Franklin Institute* 331 (5):531–54. doi:10.1016/0016-0032(94)90036-1.

Laghrib, A., A. Hadri, and A. Hakim. 2019. An edge preserving high-order PDE for multi-frame image super-resolution. *Journal of the Franklin Institute* 356 (11):5834–57. doi:10.1016/j.jfranklin.2019.02.032.

Ley, A., O. D'Hondt, and O. Hellwich. 2018. Regularization and completion of TomoSAR point clouds in a projected height map domain. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 11 (6):2104–14. doi:10.1109/JSTARS.2018.2814219.

Li, J., Y. Xiang, J. Hou, and D. Xu. 2018. Non-local texture optimization with wasserstein regularization under convolutional neural network. *IEEE Transactions on Multimedia* 21 (6):1437–49. doi:10.1109/TMM.2018.2880604.

Marr, D., and T. Poggio. 1976. Cooperative computation of stereo disparity. *Science* 194 (4262):283–7. doi:10.1126/science.968482.

Müller, M. M., M. Trautmann, and C. Keitel. 2016. Early visual cortex dynamics during top–down modulated shifts of feature-selective attention. *Journal of Cognitive Neuroscience* 28 (4):643–55. doi:10.1162/jocn_a_00912.

Okura, A., H. Yuasa, and T. Arai. 2001. Autonomous decentralized 3D recognition system for moving object. Proceedings of the 19th Annual Conference of the Robotics Society of Japan, 853–4. https://ci.nii.ac.jp/naid/10025471251/

Poggio, T., V. Torre, and C. Koch. 1985. Computational vision and regularization theory. *Nature* 317 (6035):314–9. doi:10.1038/317314a0.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–52. doi:10.1007/s11263-015-0816-y.

Thimbleby, H. W., S. Inglis, and I. H. Witten. 1994. Displaying 3D images: Algorithms for single-image random-dot stereograms. *Computer Magazine* 27 (10):38–48. doi:10.1109/2.318576.

Ueyama, E., H. Yuasa, S. Hosoe, and M. Ito. 1996. Figure-ground separation for dynamic image with Ginzburg-Landau equation by use of local clustering. *Transactions of the Society of Instrument and Control Engineers* 32 (11):1544–52. doi:10.9746/sicetr1965.32.1544.

Zhang, Y., X. Xie, X. Wang, Y. Li, and X. Ling. 2018. Adaptive image mismatch removal with vector field interpolation based on improved regularization and Gaussian kernel function. *IEEE Access* 6:55599–613. doi:10.1109/ACCESS.2018.2871743.

Zhou, Y., J. Han, F. Yang, K. Zhang, and R. Hong. 2018. Efficient correlation tracking via center-biased spatial regularization. *IEEE Transactions on Image Processing* 27 (12): 6159–73.