Sensors and Materials, Vol. 33, No. 2 (2021) 789–804
MYU Tokyo

S & M 2495

Adaptive Method to Locate Seed Points Based on Information Entropy and Quadtree

Xiaofu Du,[1,2] Huilin Liu,[1*] and Hsien-Wei Tseng[3**]

[1]School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
[2]School of Information Science and Technology, Xiamen University Tan Kah Kee College, Zhangzhou 363105, China
[3]School of AI, Guangdong & Taiwan, Foshan University, Foshan, Guangdong 528225, China

(Received July 20, 2020; accepted January 21, 2021)

Keywords: vector field visualization, streamline, entropy gradient field, quadtree

Streamlines in a signal field are analyzed to describe the changes in the signal distribution of wireless sensors in this study. To generate streamlines effectively and efficiently with seed points in a vector field, we combine several algorithms to propose an adaptive method. The method is based on a quadtree data structure and information entropy. First, we improve the speed of calculating the entropy field in a vector field by an order of magnitude using a fast entropy field calculation algorithm. In the entropy gradient field, seed points are deployed along the direction of the gradient at a certain interval from the existing seed points using an entropy gradient field seeding algorithm. Then, a quadtree grid in the entropy field is obtained by dividing the field into multiple levels with high entropy using the quadtree entropy field segmentation algorithm. Upon doing this, all nodes of the grid become seed points. These algorithms significantly improve the efficiency of seed point deployment, with different densities in different locations. As a result, a better layout of streamlines in the vector field is generated.

1. Introduction

The signal strength of a sensor network influences the use of wireless sensors in the network. The distribution of signal intensity is considered as a signal field. In particular, the gradient distribution of a signal constitutes a vector field. The signal gradient field indicates the vector direction in which the signal strength fades fastest, so the analysis of the signal gradient field is of great significance in studying the sensor signal distribution. There are many visualization techniques for vector signal gradient fields. Among them, streamline analysis is the most common and important method as a method of multivariate data analysis.(1) The visualization technology of a vector field uses the tangent direction of a point on a streamline, which is always identical to the vector direction at that point. Therefore, the use of a streamline provides a full picture of a vector field. The seed point of a streamline is the starting point and influences the final layout of the streamline. The layout of streamline seed points is important for analyzing the signal field of wireless sensors, which is discussed in the present article.

*Corresponding author: e-mail: liuhuilin@mail.neu.edu.cn
**Corresponding author: e-mail: hsienwei.tseng@gmail.com
https://doi.org/10.18494/SAM.2021.3047

It is important to deploy seed points in streamlines to obtain important information. However, too many seed points reduce the drawing speed and cause visual confusion. To solve this problem, many researchers have proposed methods of deploying seeds. In 2010, Xu *et al.* proposed the use of information entropy 1) to measure the information content at each point in a vector field and 2) to design a reasonable layout of seed points in streamlines.[2] Xu *et al.* also used the concept of the information entropy of the vector field with the entropy field and conditional entropy to suggest methods of template-based initial seed selection, importance-based seed sampling, and redundant streamline pruning. These methods enabled as much information as possible to be expressed with fewer streamlines, minimized visual interference, and fewer occlusion problems. Other methods based on information entropy have been proposed subsequently but had a long calculation time. Even though the seed point layout based on conditional entropy and redundant streamline pruning improved the final streamline layout, a long time was required.

In recent years, research on algorithms for seed point deployment in streamlines has focused on technology based on information entropy. Previous studies showed recognizable results (refer to Sect. 2) but needed improvement. First, algorithms deploying seed points in streamlines based on information entropy had low time efficiency. In particular, the algorithm with the conditional entropy for streamline advection required repeated iteration, making it time-consuming. Secondly, although some of the methods used a different way of locating seed points with information entropy, the seed points did not reflect the local information entropy.

To solve these problems, we propose an adaptive method for deploying seed points based on information entropy and a quadtree. A quadtree has a tree data structure with four children. Quadtrees are used to divide a two-dimensional (2D) area into quadrants. The proposed method consists of three different algorithms: a fast entropy field calculation algorithm, an entropy gradient field seeding algorithm, and a quadtree entropy field segmentation algorithm. The combined method generates high-quality streamlines at a high speed. It not only retains all of the important information of vector fields but also avoids visual interference and occlusion. The generated streamlines not only guarantee the coverage of a vector field without blank areas but also extract all feature points. Moreover, iterations are not necessary, which saves calculation time. We expect the new adaptive method to be applied in various fields of study.

The remainder of this paper is arranged as follows. In Sect. 2, we review the relevant works. In Sect. 3, we introduce the theoretical background and main idea of the method with specific layouts. In Sect. 4, we discuss test cases to evaluate the method. In Sect. 5, we summarize the results and discuss the limitations and possible future research.

2. Related Works

It is well known that the locations of seed points in streamlines have a direct impact on streamline visualization. Early studies proposed fast seed point deployment methods. Jobard and Lefer proposed an algorithm to create a 2D steady flow field of equally spaced lines that had a high time efficiency.[3] Liu *et al.* used an algorithm that increased the time efficiency by an order of magnitude while obtaining better quality streamlines.[4] Spencer *et al.* tested

an equal-spacing streamline generation method on a three-dimensional (3D) object surface. The method could self-adapt and be applied to complex 3D object surfaces at a high speed.[5] Another fast seed point deployment method was proposed by Mebarki *et al.*[6] All of these methods have advantages such as simple processes and high speeds, but they do not carry out the structural analysis of vector fields and have the important features of the vector field.

Then, a feature-based vector field visualization method was proposed. This method extracts the feature points in the vector field, which include the center point, the saddle point, the source point, the convergence point, and so forth. The method finds the boundary of the influence of the streamline from each feature point. Then, the feature points and boundaries constitute the topological structure of the vector field. Helman and Hesselink first proposed the concept of a vector field topology (VFT) and a method to classify feature points by eigenvalue–eigenvector analysis of the Jacobian matrix.[7] This method was developed in a 3D vector field.[8] Later studies focused on how to extract the critical points. For example, Batra and Hesselink found the critical points by linear interpolation in a triangular grid,[9] and Li *et al.* extracted the higher-order critical points using the local linear vector field on a triangular plane or curved surface grid.[10] Tricoche *et al.* used the Poincaré index to extract critical points.[11] Some researchers used non-numerical methods to extract feature points. Polthier and Preuss proposed a critical point extraction method in an unstructured triangular mesh vector field using a discrete Hodge decomposition in 2D space.[12] Tong *et al.* extended this method to a 3D tetrahedral mesh,[13] and Guo *et al.* extended the method to a regular grid.[14]

Xu *et al.* proposed an independent streamline layout optimization method that was based on the information entropy[2] introduced by Shannon[15] to measure the amount of information. Chen *et al.* improved Xu *et al.*'s method, also using information entropy to find critical points in the vector field, then grouping all streamlines by a clustering method.[16] Finally, streamlines were replaced by the Streamtape technology for drawing images that better expressed the direction and structure information. Wang *et al.* introduced information entropy into the process of texture rendering in a vector field, proposing a texture generation algorithm based on fuzzy feature measurement and information entropy.[17] Zhang *et al.* made an in-depth analysis of a streamline generation method using information entropy and proposed algorithms for feasible streamline generation and streamline quality evaluation.[18] However, the algorithms were time-consuming. Thus, Yusoff *et al.* first proposed the use of vector magnitudes to calculate information entropy and optimize a streamline layout.[19] On the basis of this research, Guo *et al.* calculated information entropy by utilizing the vector direction and vector magnitude comprehensively, which not only extracted feature regions but also visualized regions with abrupt changes in the vector magnitude such as shock waves.[20]

Through these studies, the occlusion of a vector field became an important problem to solve. Tao *et al.* introduced the concept of viewpoints based on information entropy and created viewpoint sets based on global vector data, thus realizing the visualization of flow field information under different viewpoints.[21] From this, Lu *et al.* combined information entropy and the k-means clustering method to select viewpoints more efficiently.[22] Liu *et al.* suggested a method based on information entropy and Clifford algebra for the automatic detection of flow field feature information. The automatic feature-matching operation was carried out by

the Clifford convolution method.[23] Huang and Zhang proposed two algorithms for seed point deployment that used information entropy: an algorithm with a greedy strategy and the Monte Carlo algorithm. These two methods generated uniformly distributed seed points and used information entropy for screening the points.[24]

3. Basic Ideas and Algorithms

A discrete definition of Shannon's entropy was given by Xu *et al.* as[2]

$$H(X) = -\sum_{x_i \in X} p(x_i) \log_2 p(x_i), \tag{1}$$

where X is a random variable for x_i and $p(x_i)$ is the probability of the possible result x_i. x_i is obtained by the following equation:

$$p(x_i) = \frac{C(x_i)}{\sum_{i=1}^{n} C(x_i)}, \tag{2}$$

where $C(x_i)$ is the frequency of the possible result x_i.

The specific process of using these two equations to calculate the entropy field is discussed as follows.

3.1 Fast entropy field calculation algorithm

This algorithm calculates an entropy field and improves the computing efficiency by an order of magnitude. To solve the inconvenience of irregular grids, we used mesh unit filling preprocessing (MUFP) to transform irregular grids into regular grids composed of many small square pixels.[1] For a 2D vector field, the random variable X in Eq. (1) refers to the possible directions of all vectors in the neighborhood of a sampling point in the vector field. In Xu *et al.*,[2] the range of the angle of the 2D vector (0, 360) was discretized and divided into 60 angle intervals. That is, in Eq. (2), $n = 60$. Here, x_1 is the event in the angle range between 0 and 6°, x_2 is the event in the angle range between 6 and 12°, and so on. A reasonable definition of the neighborhood of the sampling point was also given by Xu *et al.* In this study, we define the neighborhood of a sampling point as a square around the sampling point whose side length is 13 and whose area contains $13^2 = 169$ pixels.

That is, in Eq. (2), the value of $\sum_{i=1}^{n} C(x_i)$ is fixed to 169. Therefore, Eq. (2) can be simplified to

$$p(x_i) = \frac{C(x_i)}{169}. \tag{3}$$

Equation (1) can be simplified to

$$\begin{aligned} H(X) &= -\sum_{x_i \in X} p(x_i)\log_2 p(x_i) \\ &= -\sum_{i=1}^{60} p(x_i)\log_2 p(x_i) \\ &= -\sum_{i=1}^{60} \frac{C(x_i)}{169}\log_2 \frac{C(x_i)}{169} \\ &= -\sum_{i=1}^{60} \frac{C(x_i)}{169}\left(\log_2 C(x_i) - \log_2 169\right) \\ &= -\sum_{i=1}^{60} \frac{C(x_i)}{169}\log_2 C(x_i) + \sum_{i=1}^{60} \frac{C(x_i)}{169}\log_2 169 \\ &= \frac{\log_2 169}{169}\sum_{i=1}^{60} C(x_i) - \frac{1}{169}\sum_{i=1}^{60} C(x_i)\log_2 C(x_i) \\ &\because \sum_{i=1}^{60} C(x_i) = 169, \\ &\therefore \quad \approx 7.4 - \frac{1}{169}\sum_{i=1}^{60} C(x_i)\log_2 C(x_i) \end{aligned} \tag{4}$$

Thus, the simplified version of Eq. (1) for information entropy is

$$H(X) \approx 7.4 - \frac{1}{169}\sum_{i=1}^{60} C(x_i)\log_2 C(x_i). \tag{5}$$

Equation (5) applies to the case that the side length of the square surrounding the sampling point is fixed to 13. When programming the process to obtain information entropy, we define an integer array *CX*(60) and initialize all of its elements to 0. We use the array to store the polar histogram of a sampling point. Then the direction angles and angle intervals of 169 vectors in the neighborhood of the sampling point are calculated. Then, 1 is added to the value of the element of the corresponding *CX* array. After all of these 169 neighborhood vectors are processed, the frequency $C(x_i)$ of each x_i is stored in the *CX* array. Finally, the entropy is obtained with Eq. (5).

The direction angle of a vector is calculated as

$$Angle = \left(floor\left(\arcsin\left(\frac{v_y}{\sqrt{{v_x}^2 + {v_y}^2}} \right) \times \frac{180}{\pi} \right) + 360 \right) \%360, \tag{6}$$

where v_x and v_y are the *X*-axis and *Y*-axis components of the vector, respectively.

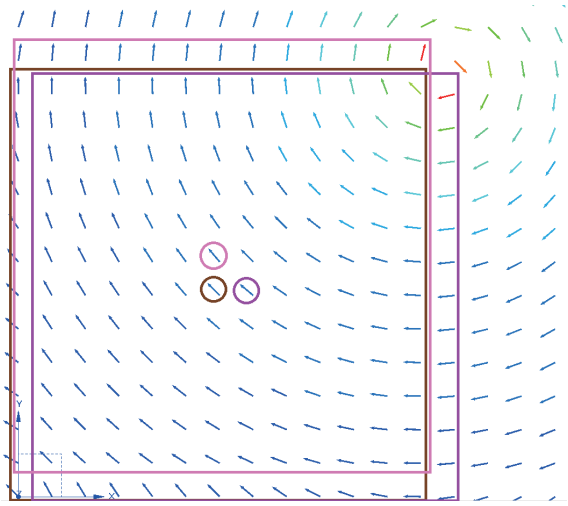


Fig. 1. (Color online) Example of calculating an entropy field.

With repeated calculation, the total frequency $C(x_i)$ of the two adjacent sampling points is obtained as shown in Fig. 1. To calculate the entropy of the sampling point marked by the brown circle in Fig. 1, we need to analyze the direction angles of 13 × 13 = 169 vectors within the range of the brown square and enter the 169 angles in the *CX* array. The entropy of the adjacent purple sampling point is calculated by the same process. The 169 additions for the calculation of the entropy of the purple sampling point and the brown sampling point are repeated 11 × 13 = 143 times. We use the histogram of the *CX* array of the brown sampling points to calculate the entropy value of the purple sampling point and eliminate 143 repeated calculations. The method is as follows.

The calculation of the entropy of the brown sampling point retains the *CX* array in the information of the direction distribution of the 169 vectors in the brown square. We subtract the 13 vectors at the left edge of the brown square from the *CX* array and then add the 13 vectors at the right edge of the purple square. Then, we obtain the distribution information of the directions of the 169 vectors in the neighborhood of the purple sample point. This method simplifies the 169 operations to 13 subtractions and 13 additions. If the side length of the sampling point in the neighborhood square is N, the time efficiency changes from $O(N)^2$ to $2O(N)$ for obtaining the frequency array of the distribution information on the direction of the neighbor of a sampling point. This result greatly improves efficiency. Similarly, using the histogram array of the brown sampling point to solve that of the pink sampling point greatly reduces the number of calculations. Using the above logic, the pseudocode of the algorithm is described as follows.

```
   If (Vector field using irregular grid) {
   Transform it into a regular square grid using the MUFP method;
}
int x, y;
double angle[width][height];// Saves the vector direction angles of all pixels in the vector field
for(x=0;x< width;x++){
for(y=0;y< height;y++){
```

```
            angle[x][y]=The direction angles of pixel p(x,y) calculated by Eq. (4);
}
} // This two-layer loop calculates the direction angles of all pixels in the vector field in advance, avoiding
repeated calculation in the later stage and improving the speed.
double oneEntropy;// Save the entropy of pixel p(x,y)
int CX[60]={0};// Save the histogram of direction angles of neighborhood vectors of a sampling point
int pCX[60]={0};// Save the histogram of the direction angle of the neighborhood vector of the previous
sampling point

double EntropyField[width][height];// Save the entropy field of the vector field
int N=13;// The length of the neighborhood square's side
for(x=0;x<N;x++){
for(y=0;y<N;y++){
   CX[angle[x][y]/6]++;
   pCX[angle[x][y]/6]++;
}
} // This two-layer loop calculates the histogram of the bottom left point (N/2, N/2) of the vector field
oneEntropy=The entropy calculated using Eq. (3) and the array CX;
x=N/2, y=N/2;
   EntropyField[x][y]=oneEntropy;
for(y=N/2;(y+ N/2)< height; y++){
   pCX=CX;
   for(x=N/2;(x+ N/2)< width; x++){
if(y==N/2&& x==N/2){
break;
}// Skip the sampling point at the bottom left because it has already been calculated.
if(x==N/2){
for(int i=0;i<N; i++){
            CX[angle[x- N/2+i][y- N/2-1]/6]--;
                        CX[angle[x- N/2+i][y+ N/2]/6]++;
}
                  oneEntropy=The entropy calculated using Eq. (3) and the array CX;
   EntropyField[x][y]=oneEntropy;
continue;
}
for(int i=0;i<N; i++){
   CX[angle[x- N/2-1][y- N/2+i]/6]--;
                  CX[angle[x+ N/2][y- N/2+i]/6]++;
}
            oneEntropy=The entropy calculated using Eq. (3) and the array CX;
            EntropyField[x][y]=oneEntropy;
}
   CX= pCX;
}
```

The above algorithm is invalid for the sampling points at the edge of the vector field because they do not have enough neighbors. In this case, the entropy of these edge sampling points needs to be calculated using the basic method.

3.2 Entropy gradient field seeding algorithm

An entropy gradient field in an original vector field is first obtained from the information entropy, and then all important areas in the vector field are found with various feature points. Seed points are located in the area of a rhombus. The entropy gradient is calculated using the

entropy value at each sampling point to obtain the entropy gradient field. A series of seed points from the rhombus is obtained at appropriate intervals along the gradient direction. The obtained seed points ensure that no important information is missed. The streamlines generated from these seed points cover most of the non-critical areas. However, in some cases, the series of seed points create some blank areas in the original vector field, which need to be supplemented by the second algorithm. Entropy is mathematically a scalar, and so is the entropy field. All scalar fields are calculated to obtain the corresponding gradient field. This gradient field is a vector field, and the direction of the vector in the field contains some information. For example, in the gradient fields of the scalar fields of height, temperature, and entropy, the gradients indicate the direction in which the height, temperature, and entropy decrease most rapidly. The information entropy is the amount of information. Thus, the directions in which the entropy and the amount of information decrease most rapidly are the same. The fastest transition from more information to less information is most likely to be achieved by placing a series of seed points at an appropriate interval in the direction. This means that the streamlines from these seed points cover as many representative areas as possible according to the amount of information. This is the basic principle of locating seed points using the entropy gradient field.

The determination of the origin of the sequence of seed points helps understand the algorithm of deploying seed points in the entropy gradient field. It is natural to start with the feature points of the vector field. In the rhombic layout, there are four sides around a feature point and each side has four seed points. From the four seed points, calculation of the entropy gradient field is started and extended to the four series of seed points at a certain interval to perfectly cover the whole vector field.

Then, it is possible to determine the distance between seed points. A larger modulus of the entropy gradient at a certain point means that the change in the information entropy around the point is larger. Thus, more streamlines are needed to describe the changes. A smaller modulus pertains to a smaller change in the information entropy and fewer streamlines. The basic idea is to calculate an appropriate interval between seed points based on the entropy gradient modulus. The larger the gradient modulus, the shorter the distance.

This is described in the following equation:

$$L = \left(1 + B \times (5.907 - G_m)\right) \times A\,, \tag{7}$$

where A is the shortest distance between two adjacent seed points in the same row, B is a coefficient used to adjust the distance between two seed points, and G_m is the modulus of the information entropy gradient at the current sampling point. The constant of 5.907 in the equation is the maximum information entropy of a sampling point according to Eq. (1).

Common methods for calculating the gradient of an entropy field include a central difference operator, an adjacent gradient factor, and the Sobel operator.[25] We choose the Sobel operator as it has the advantages of high precision and speed. Figure 2 shows a schematic diagram of the Sobel operator of a 2D scalar field.

The equations for calculating the entropy gradient of point (x, y) are as follows.

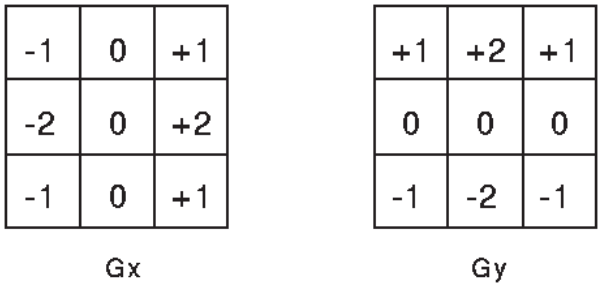


Fig. 2. Schematic diagram of Sobel operator of a 2D scalar field.

$$\begin{aligned} Gx = &\left[f(x+1, y-1) + 2 * f(x+1, y) + f(x+1, y+1) \right] \\ &- \left[f(x-1, y-1) + 2 * f(x-1, y) + f(x-1, y+1) \right] \end{aligned} \tag{8}$$

$$\begin{aligned} Gy = &\left[f(x-1, y-1) + 2f(x, y-1) + f(x+1, y-1) \right] \\ &- \left[f(x-1, y+1) + 2 * f(x, y+1) + f(x+1, y+1) \right] \end{aligned} \tag{9}$$

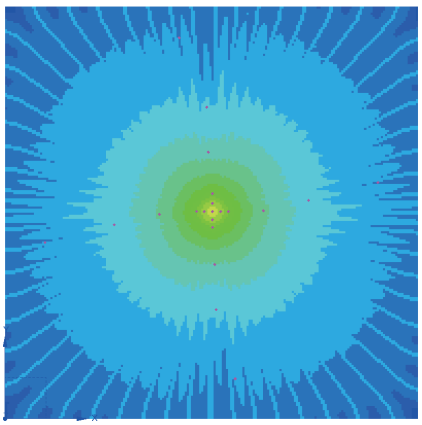
Figure 3 shows the entropy field calculated using a typical center vector field. Figure 3(a) depicts the cloud map generated from the entropy field. The closer to the center, the greater the change in the vector direction of the local region, resulting in a higher entropy value. The seed points are shown in Fig. 3(b). The seed point in the center is the feature point with four seed points around it arranged in a rhombus, as in the traditional entropy method. Then, starting from the four seed points in the periphery, four series of seed points are generated by the above entropy gradient field method. Figure 3(c) shows the streamline effect traced from the above seed points.

It is worth noting that the example vector field is the standard central vector field, and the theoretical result of the calculated entropy field with the standard model has the structure of strictly concentric circles. However, the calculation error makes the entropy field not strictly concentric. The trajectories of the four seed points are also straight lines from the center in the top, bottom, left, and right directions. Figure 4 shows the entropy field, seed points, and streamlines in a standard saddle point vector field.

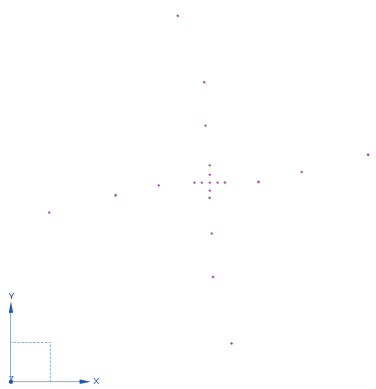
3.3 Quadtree entropy field segmentation algorithm

The entropy gradient field seeding algorithm covers the important areas of the vector field with fewer seed points, producing excellent streamlines. However, as shown in Figs. 3 and 4, there are large gaps in the vector field. Thus, we use the quadtree method to make sure that there are no large gaps in the vector field.

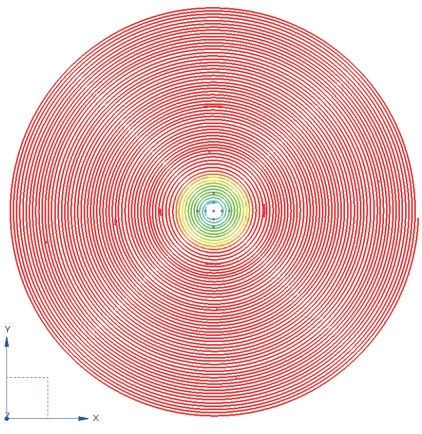
The quadtree is an idea of analysis and classification for 2D spatial data that is often used in image segmentation, image compression, geographic information analysis, and so forth.[26] The basic idea is to judge the 2D space according to a certain attribute, divide it into four equal



(a)

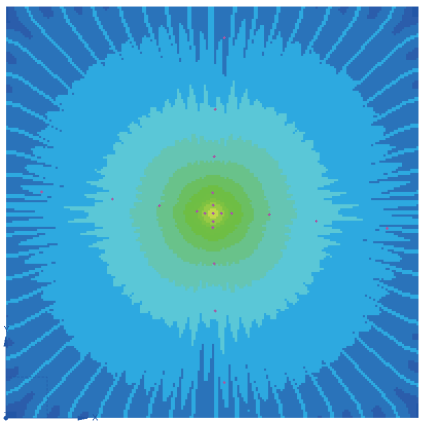


(b)

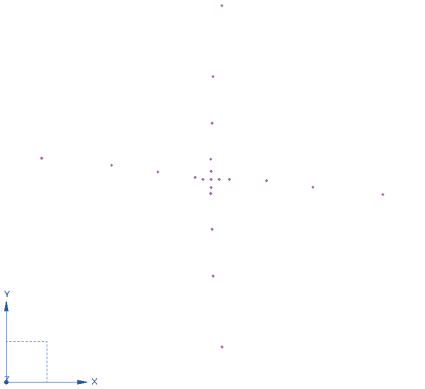


(c)

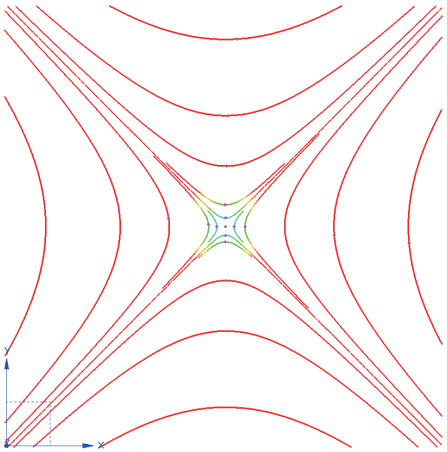
Fig. 3. (Color online) Entropy field, seed points, and streamlines in a central vector field. (a) Cloud map of the entropy field. (b) Seed points obtained by the entropy gradient field seeding algorithm. (c) Streamlines traced from the seed points in (b).



(a)



(b)



(c)

Fig. 4. (Color online) Entropy field, seed points, and streamlines in a saddle point vector field. (a) Cloud map of the entropy field. (b) Seed points obtained by the entropy gradient field seeding algorithm. (c) Streamlines traced from the seed points in (b).

parts, and then recursively judge and divide each part. The quadtree technology divides a 2D space into several small squares, where the higher the attribute value, the denser the partition.

To ensure that there are basic seed points in each region, the vector field is first divided into squares of a basic size defined in the algorithm. Then, all vertices become seed points. The sides of these basic squares have appropriate lengths. If the sides are too long, they have too few seed points, with important information missed. However, if the sides are too short, the seed points are too densely located and the speed of calculation is lowered, causing visual confusion. In the algorithm, we take the longer side of the vector field as a criterion and then divide it into 10 equal parts. After that, we divide each basic square into a quadtree to judge whether the average entropy value of all pixels inside the current square is higher than a threshold. If so, the square is divided into four equal parts again. Then, the newly divided square is judged recursively. After a few recursions, a set of target seed points is obtained.

The quadtree should not be divided into too many squares. In theory, each square is divided three times (in three degrees) and the length of each side becomes one-eighth of the original. In the worst case, the long side of the entire vector field is divided into 80 segments with 81 seed points. As a vector field cannot have such high entropy, the degree of the quadtree division by the algorithm is fixed to be 3.

The threshold for a square must be different for different degrees of quadtrees. The lower the degree, the larger the squares and the smaller the average of their internal entropy. Moreover, as the degree increases, the squares become smaller, and the possible average value of entropy in these small squares becomes higher once the feature points are included. Therefore, we design a threshold of the average value of entropy that increases with the degree. The equation used to calculate the threshold is

$$E_{Threshold} = A \cdot 2^{D-4} \cdot 5.907\,, \tag{10}$$

where A is the control coefficient and D is the current degree of the quadtree from 1 to 3. The value of A decides the density of the seed points. The constant of 5.907 in the equation is the maximum information entropy of a sampling point according to Eq. (1).

We give the flow chart of the quadtree entropy field segmentation algorithm based on the above explanation in Fig. 5.

The function in Fig. 5(a) performs the basic segmentation of the vector field to obtain the basic square. This step does not use the quadtree idea but basic average division. The function *DivOneQuaTree*(*thisSquare*,1) is then called for each square. This function is recursive, as parameter 1 is the square to be divided and parameter 2 is the degree.

Figure 5(b) shows the result using the *DivOneQuaTree* function. The recursion ends when the parameter degree is 4. When the degree is less than 4, the average entropy in the current square is judged to be higher or lower than the threshold value. If the average entropy is higher, the recursion continues, but if the average entropy is lower, the recursion ends.

Figure 6 shows the set of seed points and streamlines of the central vector field obtained by the quadtree entropy field segmentation algorithm. In the programming, the quadtree entropy field segmentation algorithm and the entropy gradient field seeding algorithm are used

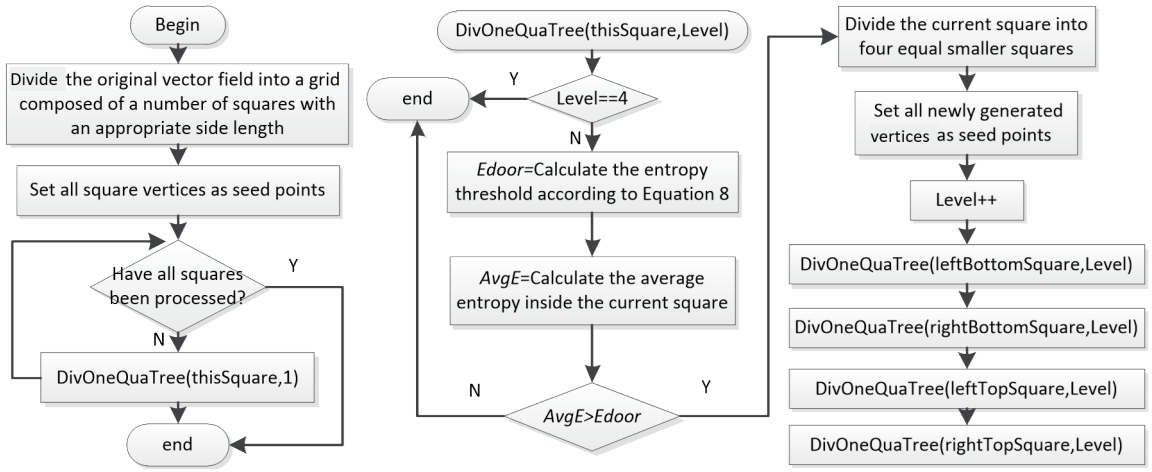


Fig. 5. Program flow chart of the quadtree entropy field segmentation algorithm. (a) Main function of the quadtree entropy field segmentation algorithm. (b) Recursive function used to divide a square.

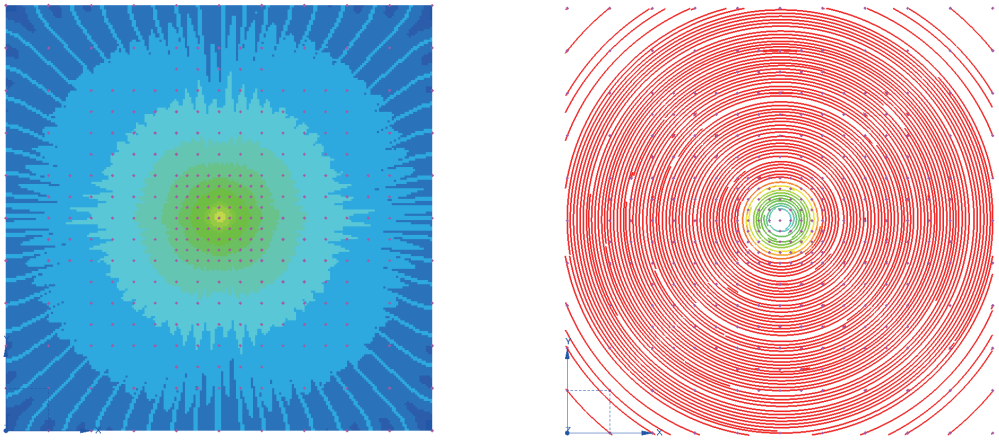


Fig. 6. (Color online) Seed points and streamlines of a central vector field obtained by using the quadtree entropy field segmentation algorithm. (a) Cloud map of the entropy field and the seed points obtained by the quadtree entropy field segmentation algorithm. (b) Streamlines traced from the seed points in (a).

in combination. Firstly, the entropy field in the vector field is calculated by the fast entropy field calculation algorithm to find the feature points. Each feature point is surrounded by four seed points in a rhombus. In the second step, starting from the four seed points outside of each feature point, the entropy gradient field seeding algorithm is used to produce multiple series of seed points. The third step uses the quadtree entropy field segmentation algorithm to accomplish the complete coverage of the vector field.

4. Results and Discussion

To verify the effectiveness and applicability of the above algorithms, we use two sets of wind field data to conduct experiments. The experimental results are as follows.

An example of a wind field is shown in Fig. 7. To draw Fig. 7(a), we calculated the characteristic points using the entropy field and made a rhombic form to arrange the seed points, and then used the seed points only to trace the streamlines. There are many blanks in the flow field. Figure 7(b) shows the generated streamlines after expanding the sets of seed points with the entropy gradient field seeding algorithm. Owing to the calculation error and other factors, there are still some blank areas. Figure 7(c) shows the entropy in the vector field and the set of seed points obtained by the quadtree entropy field segmentation algorithm. Figure 7(d) shows the streamlines generated from the seed points of a quadtree. It can be seen that

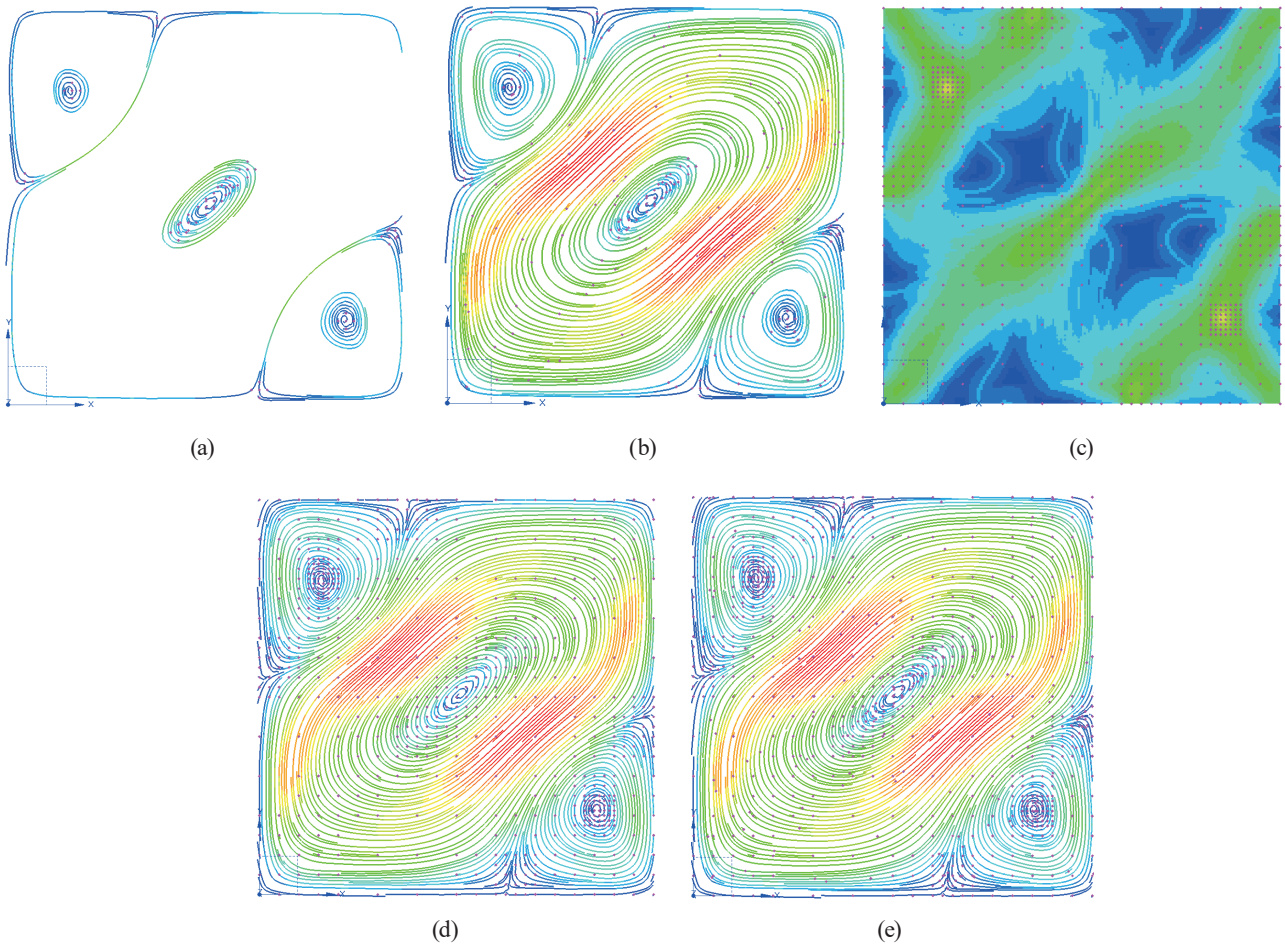


Fig. 7. (Color online) Seed points and streamlines of a wind field. (a) Rhombic seed points around the feature points and the streamlines traced from them. (b) Seed points obtained by the entropy gradient field seeding algorithm and the streamlines from them. (c) Cloud map of the entropy field and the seed points obtained by the quadtree entropy field segmentation algorithm. (d) Streamlines traced from the seed points in (c). (e) Streamlines traced from the seed points in (a), (b), and (c).

areas with higher entropy have more seed points, while areas with lower entropy have fewer seed points. Finally, Fig. 7(e) shows the streamlines after mixing the three different sets of seed points. Figure 8 shows seed points and streamlines of the other wind field data obtained by the different algorithms.

Table 1 shows the times required to calculate the entropy field by the two methods. The results demonstrate that the fast entropy field calculation algorithm is ten times faster than the classical entropy field calculation algorithm. Table 2 shows the time required for each step for the two wind field examples in Figs. 7 and 8. Each step requires a very short time. The three algorithms are completed in less than 0.2 s in each example. Table 3 compares the method in this study with other non-entropy methods and indicates that the new method gives better results with a higher speed.

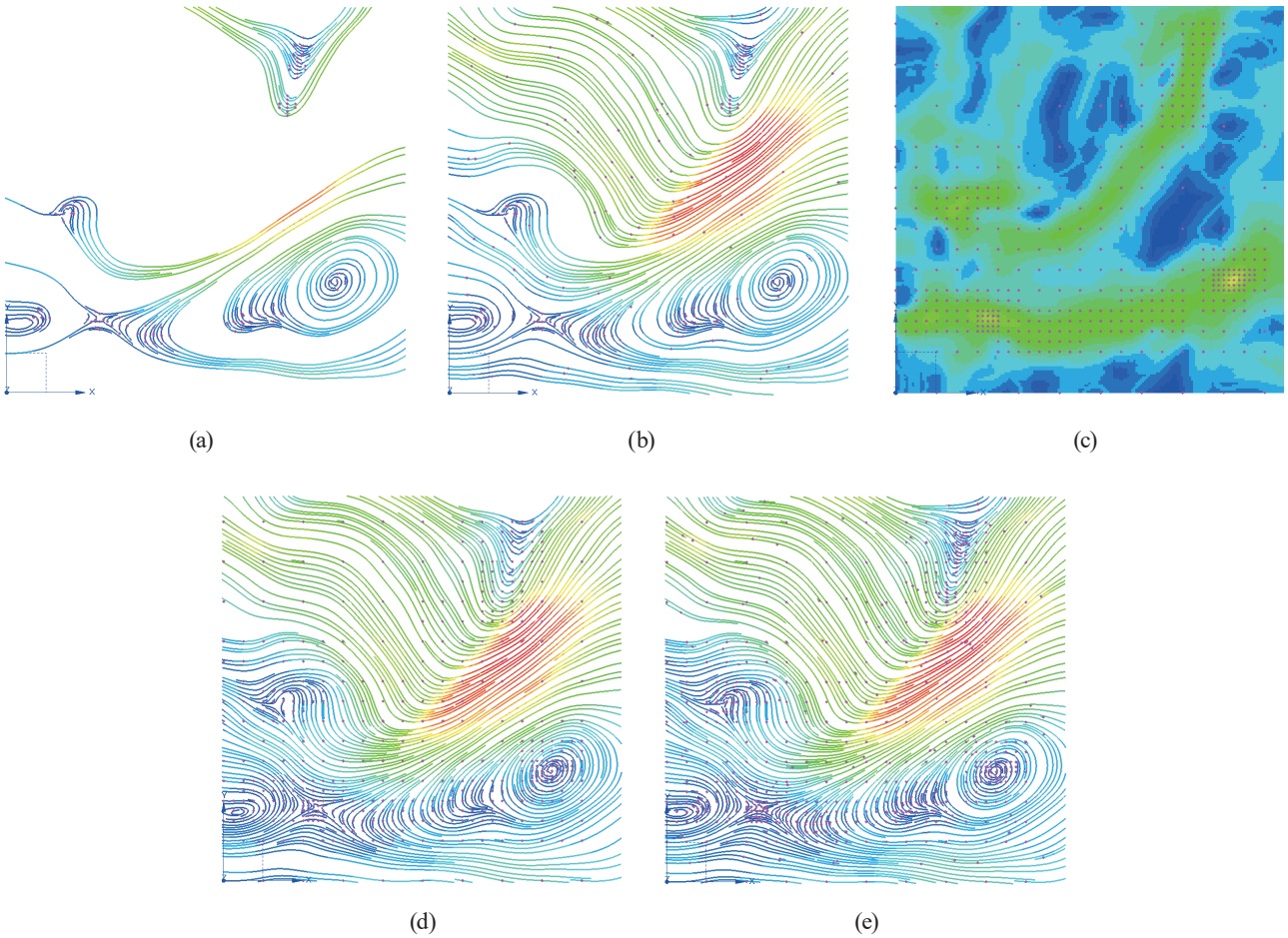


Fig. 8. (Color online) Seed points and streamlines of a wind field. (a) Rhombic seed points around the feature points and the streamlines traced from them. (b) Seed points obtained by the entropy gradient field seeding algorithm and the streamlines from them. (c) Cloud map of the entropy field and the seed points obtained by the quadtree entropy field segmentation algorithm. (d) Streamlines traced from the seed points in (c). (e) Streamlines traced from the seed points in (a), (b), and (c).

Table 1
Times required to calculate the entropy field by the two methods (in s).

Algorithm	Fig. 7	Fig. 8
Classical entropy field calculation algorithm	0.489	0.364
Fast entropy field calculation algorithm	0.042	0.031

Table 2
Times required for each step (in s).

Step	Fig. 7	Fig. 8
Mesh unit filling preprocessing	1.026	1.096
Fast entropy field calculation algorithm	0.042	0.031
Determination of feature points and use of rhombic form to locate seed points	0.004	0. 004
Entropy gradient field seeding algorithm	0.137	0.182
Quadtree entropy field segmentation algorithm	0.006	0.005
Tracing of streamlines from set of seed points	2.798	2.350
Total time	4.013	3.668

Table 3
Comparison of characteristics of several streamline layout methods.

Method	Use of information entropy	Adopting seed point layout	Blanks in streamlines of vector field	Topology of vector field	Calculation speed
Ref. 2	Yes	Yes	Yes	Yes	Very slow
Ref. 3	No	No	Yes	No	Fast
Ref. 4	No	No	Yes	No	Very fast
Ref. 11	No	No	No	Yes	Slow
Ref. 19	Yes	Yes	Yes	No	Very fast
Ref. 20	Yes	Yes	Yes	No	Very fast
This study	Yes	Yes	Yes	Yes	Fast

5. Conclusions and Future Work

To generate better sets of seed points at a higher speed, three algorithms are proposed in this paper: 1) the fast entropy field calculation algorithm, which increases the computation speed of the entropy field by more than ten times, 2) the entropy gradient field seeding algorithm, which uses the concept of the entropy gradient field to arrange the seed points along the entropy gradient direction, and 3) the quadtree entropy field segmentation algorithm, which segments a vector field using quadtree theory and entropy value judgment, where regions with higher entropy are divided into more areas by quadtrees to obtain a denser grid. The combination of these three algorithms produces high-quality sets of streamlines with the seed points at a higher speed and describes as much information as possible with as few streamlines as possible.

The algorithms still have some drawbacks. The generated streamlines do not give a fully developed visual effect as the streamlines are crowded. For a future study, a fast streamline redundancy clipping algorithm needs to be developed to reduce visual confusion. Moreover, the application of the above three algorithms to vector fields in 3D space and dynamic vector fields (which vary with time) should be researched with an appropriate selection of path line seed points. The results of this study are expected to lead to further analysis of the signal gradient field of sensor networks for developing improved sensor layouts.

Acknowledgments

This work was supported by the Natural Science Foundation of Zhangzhou, Fujian (Project No. ZZ2020J30), the TKKC Pre-Research Project (Project No. YY2019L02), and the Natural Science Foundation of Fujian, China (Project No. 2018J01101).

References

1 X. Du, H. Liu, H.-W. Tseng, and T.-H. Meen: Symmetry **12** (2020) 724. https://doi.org/10.3390/sym12050724
2 L. Xu, T. Lee, and H. Shen: IEEE Trans. Vis. Comput. Graph. **16** (2010) 1216. https://doi.org/10.1109/TVCG.2010.131
3 B. Jobard and W. Lefer.: Proc. 8th Eurographics Workshop on Visualization in Scientific Computing (Boulogne sur Mer, France, 1997) 45. https://doi.org/10.1007/978-3-7091-6876-9_5
4 Z. Liu, R.-J. Moorhead, and J. Groner: IEEE Trans. Vis. Comput. Graph. **12** (2006) 965. https://doi.org/10.1109/TVCG.2006.116
5 B. Spencer, L.-S. Robert S, G. Chen, and E. Zhang: Computer Graphics Forum **28** (2009) 1618. https://doi.org/10.1111/j.1467-8659.2009.01352.x
6 A. Mebarki, P. Alliez, and O. Devillers: Proc. IEEE Visualization 2005 (Minneapolis, MN, USA, October 23-28. 2005) 61. https://doi.org/10.1109/visual.2005.1532832
7 J. Helman and L. Hesselink: IEEE Comput. **22** (1989) 27. https://doi.org/10.1109/2.35197
8 J. Helman and L. Hesselink: IEEE Comput. Graphics Appl. **11** (1991) 36. https://doi.org/10.1109/38.79452
9 R. Batra and L. Hesselink: Proc. IEEE Visualization'99 (San Francisco, CA, USA, 24-29 Oct. 1999) 105. https://doi.org/10.1109/VISUAL.1999.809874
10 W. Li, B.Vallet, N. Ray, and B. Levy: IEEE Trans. Vis. Comput. Graph. **12** (2006) 1315. https://doi.org/10.1109/TVCG.2006.173
11 X. Tricoche, C. Garth C, and A. Sanderson: IEEE Trans. Vis. Comput. Graph. **17** (2011) 1765. https://doi.org/10.1109/TVCG.2011.254
12 K. Polthier and E. Preuss: Proc. Visualization and Mathematics III (Berlin, Germany, May 22-25, 2002) 113. https://apps.webofknowledge.com/full_record.do?product=UA&search_mode=GeneralSearch&qid=12&SID=6EqKKnSDe3Z7sxHkMQu&page=1&doc=1&cacheurlFromRightClick=no
13 Y. Tong, S. Lombeyda, A. Hirani, and M. Desbrun: ACM Trans. Graphics **22** (2003) 445. https://doi.org/10.1145/882262.882290
14 Q. Guo, M. Mandal, and M. Li: Pattern Recognit Lett. 26(2005) 493. https://doi.org/10.1016/j.patrec.2004.08.008
15 C.E. Shannon: ACM SIGMOBILE Mobile Computing and Communications Review **5** (1948) 3. https://doi.org/10.1145/584091.584093
16 C. Chen, S. Yan S, and H. Yu: Computer Graphics Forum **30** (2011) 1941. https://doi.org/10.1111/j.1467-8659.2011.02064.x
17 H.-H. Wang, H.-X. Xu, L. Zeng, and S.-K. Li: Proc. 2011 Int. Conf. Virtual Reality and Visualization (Beijing, China, 4-5 Nov. 2011) 303. https://doi.org/10.1109/ICVRV.2011.41
18 S. Zhang, L. Xie, L. Gui, and Y. Zheng: Comput. Eng. Appl. **51** (2015) 181. https://xueshu.baidu.com/usercenter/paper/show?paperid=aeb282c0f07d9e8867468cfa3bdbe03c&site=xueshu_se&hitarticle=1
19 Y.A. Yusoff, F. Mohamed F, M.K Mokhtar, B. Tomi, C.V Siang, and M.I.M. Isham: Proc. 2017 IEEE Conf. Big Data and Analytics (Kuching, Malaysia, 16 – 17 Nov. 2017) 81. https://doi.org/10.1109/ICBDAA.2017.8284111
20 Y. Guo, W. Wang, and S. Li: Proc. 12th Int. Symp. Visual Information Communication and Interaction (Shanghai, China, September 20 -22, 2019) 3356442. https://doi.org/10.1145/3356422.3356442
21 J. Tao, J. Ma, and C. Wang: IEEE Trans. Vis. Comput. Graph. **19** (2013) 393. https://doi.org/10.1109/TVCG.2012.143
22 D.-Y. Lu, D.-M. Zhu, and Z.-Q. Wang: J. Computer-aided Design Graphics **29** (2017) 2281. https://doi.org/10.3724/SP.J.1089.2017.16433
23 X. Liu, W. Zhang, and N. Zheng.: 2015 Int. Conf. Image and Graphics (Tianjin, China, August 13-16, 2015) 292-30. https://doi.org/10.1007/978-3-319-21963-9_27
24 D. M. Huang and L. W. Zhang: Comput. Eng. Sci. **3** (2018) 411. http://en.cnki.com.cn/Article_en/CJFDTotal-JSJK201803005.htm
25 L. Yang and B. Zhang: Contributions to Geology and Mineral Resources Research **33** (2018) 306. http://en.cnki.com.cn/Article_en/CJFDTotal-DZZK201802020.htm
26 H. Liu, K.-K. Huang, C.-X. Ren, Y.-F. Yu, and Z.-R. Lai: Signal Process.: Image Commun. **55** (2017) 1. https://doi.org/10.1016/j.image.2017.03.011