Implementation of Calculating Steiner Point for 2D Objects

Jiuzhen Liang[1,2] **Mirko Navara**[2]

[1]School of Information Technology, Jiangnan University, Wuxi 214122, P.R. China
[2]Center for Machine Perception, Faculty of Electrical Engineering, Czech Technical University, Technická 2, 16627 Praha 6, Czech Republic

Abstract

This approach deals with the Steiner point calculation implementation and its application in image processing. Based on Steiner point definition and some properties, a numerical method is proposed for calculating the Steiner point of a 2-dimensional polytope. Also analysis of computational complexity is presented. Some experiments on randomized 2-dimensional data and 2D image processing are provided for testing the algorithm.

Keywords: Steiner point, Image processing, Calculation, Computational complexity

1. Introduction

In 1826, Steiner described a special point of a triangle, which had important influence on geometry [1]. Forty years later, in J. Neuberg, "Sur le point de Steiner," Journal de mathématiques spéciales 1886, the point was constructed as described as in [1] and was given Steiner's name. Nowadays, there are (at least) three different types of points known as Steiner points [2]. One of them, more properly known as the Steiner curvature centroid, is the geometric centroid of the system obtained by placing a mass equal to the magnitude of the exterior angle at each vertex [4].

To locate the Steiner point of an object is helpful for many tasks, because Steiner point is an invariant point of an object while a transform is used on it in certain ways, such as growing uniformly in all directions, moving in a line, rotating around an axis [13]. By finding the Steiner point of an object, one can analyze some properties of an image [12]. To detect or recognize an object in an image, Steiner point can help us in some cases. If two objects are similar but have different Steiner points, one can distinguish them in this way [8]. Tracking moving objects is now a popular approach for research workers, if Steiner points of objects are referenced they could save a large amount of computation.

In the early years, much work has been done on some algebraic and analytic structure and behaviour of Steiner points, such as linear translation, continuity, and even affine translation of an object. Three important properties were studied and known as basic properties of Steiner points, which are shortly denoted as commutation, addition, and continuity [11], [7], [9]. Furthermore, the definition of Steiner point was generalized from a polytope to a nonempty compact convex subset K of $\mathbb{R}^n$ [6]. To implement the calculation of a Steiner point, there are several alternatives [12], one efficient way refers to [5], which is based on the exterior angle of convex points in a polytope. In recent years, Steiner point has been extended to fuzzy sets and provides an alternative strategy of defuzzication, which is regarded as the center of the fuzzy set [13].

This approach deals with the Steiner point calculation implementation, some evaluation on the promoted algorithm, and application in digital image processing. This paper is arranged as follows. The second section introduces the mathematical definition of Steiner point and some transformation properties which mainly refer to [13]. In the third part we present a numerical method for calculation of the Steiner point based on a convex polytope. The fourth part provides analysis of computational complexity. In the fifth section, some experiments on randomized 2-dimensional data and image processing are given. The last part of the paper contains conclusions.

2. Definition of steiner point and some basic properties

In the following let us suppose that $n \geqslant 1$ is an integer. We denote by K^n the set of non-empty compact convex subsets of $\mathbb{R}^n$. The set K^n is endowed with a linear structure in which the addition of two subsets and the multiplication of a subset by a positive real are defined pointwise. We furthermore endow K^n with the Hausdorff metric d_E. Let S^{n-1} denote the unit sphere in $\mathbb{R}^n$, and $C(S^{n-1})$ the space of continuous functions from S^{n-1} to $\mathbb{R}$, endowed with the supremum norm. Now, for $A \in K^n$, we define the *support function* of A, see e.g. [6], by

$$h_A : S^{n-1} \to \mathbb{R}, e \mapsto \max\{\langle a, e\rangle : a \in A\} \qquad (1)$$

where $\langle \cdot, \cdot \rangle$ denotes the usual inner product of $\mathbb{R}^n$.

The following definition is due to [7] and [13].

Definition 2.1. The Steiner point of A in K^n is defined as

$$s(A) = \frac{1}{V(B^n)} \int_{S^{n-1}} e h_A(e)\, \mathrm{d}\lambda(e)\,, \qquad (2)$$

where $e \in S^{n-1}$ varies over the unit vectors of $\mathbb{R}^n$, λ is the Lebesgue measure on S^{n-1}, and $V(B^n)$ is the volume of the unit ball B^n of $\mathbb{R}^n$. Notice that $s(A) \in A$.

More generally, consider a smooth manifold Ω of boundary $\Gamma = \partial\Omega$ containing the origin O. Mattioli [6] gave a similar definition of Steiner point and its subdifferential form definition as follows.

Definition 2.2. For a nonempty compact subset K of $\mathbb{R}^n$, its Steiner point $s_\Omega(K)$ is defined by

$$s_\Omega(K) = \frac{\lambda(\Gamma)}{\lambda(\Omega)} \int_\Gamma p\sigma(K,p)\omega(\mathrm{d}p) \qquad (3)$$

where $\sigma(K,\cdot)$ is the support function of K, ω is the measure on Γ proportional to the Lebesgue measure normalized by $\int_\Gamma \omega(\mathrm{d}p) = 1$.

The subdifferential $\partial\sigma(K,p)$ of the support function $\sigma(K,\cdot)$ is given by

$$\partial\sigma(K,p) = \{x \in K | \langle p, x\rangle = \sigma(K,p)\} \qquad (4)$$

We denote by $m(\partial\sigma(K,p))$ the element of $\partial\sigma(K,p)$ with the minimal norm.

Definition 2.3. Let K be a bounded subset of $\mathbb{R}^n$. Then Steiner point of K is given by

$$s_\Omega(K) = \frac{1}{\lambda(\Omega)} \int_\Omega m(\partial\sigma(K,p))(\mathrm{d}p) \qquad (5)$$

The above definitions of Steiner point are useful in investigation of the properties of Steiner point. From the point of computing, to calculate the Steiner point of an object, we have to use numerical integration of equation (2), (3), or (5). A usual way to approximate an object in Euclidean space is to construct a polytope. This paper focuses on calculating Steiner point on polytopes. In the case of a polytope, the Steiner point is conveniently defined as a sum involving the external angles of the polytope at its vertices. The following definition is described by [11].

Definition 2.4. Let P be any d-dimensional polytope in $\mathbb{R}^n$. The Steiner point of P is defined by

$$s(P) = \sum_{j=1}^{f_0} v_j \psi(F_j^0, P) \qquad (6)$$

where v_j is the position vector of the vertex F_j^0 of $P(j = 1, \cdots, f_0)$, and $\psi(F_j^0, P)$ is the external angle of P at F_j^0 normalized so that it satisfies $\sum_j \psi(F_j^0, P) = 1$.

For the sake of discussion, we will use several Euclidean isometries of $\mathbb{R}^n$. By a rotation, we will always mean a proper rotation which is an isometry leaving the origin fixed and continuously connected to the identity. A rigid motion is an isometry composed of rotations and translations. The following theorem is for the case $n = 2$ due to [11] and for the case $n > 2$ due to [9].

Theorem 2.1. Let $s' : S^{n-1} \to \mathbb{R}^n$ have the following properties:

($S1$) For any $A, B \in K^n, s'(A + B) = s'(A) + s'(B)$.

($S2$) For $A \in K^n$ and any rigid motion τ, we have $s'(\tau A) = \tau s'(A)$.

($S3$) s' is continuous.

Then $s' = s$ is the Steiner point. These three properties are described in [7] as addition, commutation, and continuity of Steiner point.

We denote by $\mathcal{F}^n$ the set of all functions from [0,1] to $\mathcal{K}^n$ which are (i) decreasing and (ii) left-continuous on (0,1] and continuous at 0. For a fuzzy set[1] $u \in \mathcal{F}^n$ and a rigid motion τ we set $\tau u : [0,1] \to K^n$, $\alpha \mapsto \tau(u(\alpha))$. The following theorems are due to [13].

Theorem 2.2. A function $S : \mathcal{F}^n \to \mathbb{R}^n$ is called a *Steiner point* if it has the following properties:

($SF0$) For any $u \in \mathcal{F}^n, S(u) \in u(0)$.

($SF1$) For any $u, v \in \mathcal{F}^n, S(u + v) = S(u) + S(v)$.

($SF2$) For $u \in \mathcal{K}^n$ and any rigid motion τ, we have $S(\tau u) = \tau S(u)$.

[1]Usually a fuzzy set is represented in its *vertical representation* by its membership function $U\colon R^n \to [0,1]$. Here we found it more efficient to use the *horizontal representation* by a function $u\colon [0,1] \to \mathcal{K}^n$ defined by

$$u(\alpha) = \{x \in R^n \mid U(x) \geq \alpha\}$$

for $\alpha > 0$ and by

$$u(0) = \{x \in R^n \mid U(x) > 0\}\,.$$

$(SF3)$ S is continuous.

Definition 2.5. Let $D = (\alpha_0, \cdots, \alpha_k)$ be a division of $[0, 1]$, which is $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_k = 1$. Then we call a fuzzy set $u \in \mathcal{F}^n$ a D-step fuzzy set if it is constant on $[\alpha_0, \alpha_1], (\alpha_1, \alpha_2], \cdots, (\alpha_{k-1}, \alpha_k]$, respectively. We denote by $\mathcal{F}^n_D$ the set of all D-step fuzzy sets.

Theorem 2.3. Let $D = (\alpha_0, \cdots, \alpha_k)$ be a division of $[0, 1]$, i.e. $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_k = 1$. Let $S : \mathcal{F}^n_D \to \mathbb{R}^n$ be a function fulfilling the properties $(SF0)-(SF3)$ of theorem 2. Then there are unique real numbers $\kappa_1, \cdots, \kappa_k$ such that $\kappa_1 + \cdots + \kappa_k = 1$ and for all $u \in \mathcal{F}^n_D$

$$s(u) = \kappa_1 s(u(\alpha_1)) + \cdots + \kappa_k s(u(\alpha_k)) \qquad (7)$$

3. Calculating steiner point in 2-dimensional space

Without special explanation, in the following context we consider a problem in the Euclidean space. Before giving the algorithm for computing the Steiner point of an object or a data set, let us discuss two basic problems associated to the algorithm. The first one is how to find all the extreme points of an object, the second one is how to reduce the computational complexity of the algorithm. For the first one, suppose we have N points in the data set which are denoted by $S = \{p_1, \cdots, p_N\}$. If $p_i = (x_i, y_i)$ is an extreme point, there exists a direction $e_i = (\cos\alpha_i, \sin\alpha_i)$ such that

$$\langle p_i, e_i \rangle > \langle p_j, e_i \rangle \qquad (8)$$

for $j = 1, \cdots, N$ and $j \neq i$. Denote $p_{ij} = p_i - p_j$, then we have

$$\langle p_{ij}, e_i \rangle > 0, \qquad (9)$$

which means all the difference vectors are in the same halfplane whose normal vector is e_i, see figure 1. In fact, we do not need to know the normal vector e_i. Instead, we can judge p_i is an extreme point iff the following condition is satisfied [2]

$$\max_j \{\alpha_{ij}\} - \min_j \{\alpha_{ij}\} < \pi, \qquad (10)$$

where α_{ij} is the polar coordinate angle of vector p_{ij}. Now the second problem comes forth, computing equation (10) will cost $O(N^2)$ time. Usually N is more than thousands, so it is necessary to reduce the computing amount. In fact, much more points

[2] This is a calculation modulo 2π; as such, it requires more attention to be formulated correctly. E.g., there is an interval of length $< \pi$ containing all α_{ij}. So all the angles should be normalized to the interval $[0, 2\pi]$.

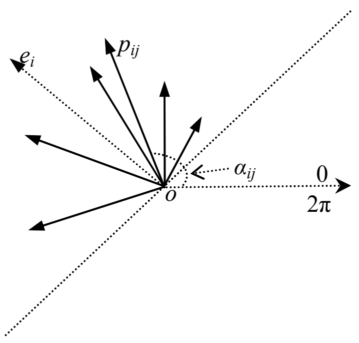

Fig. 1: Distribution of vectors in 2-dimensional space.

can be eliminated before computing equation (10). We call these points inner points. They can be detected by an assistant convex polygon whose constructions will be described in the following algorithm.

Generally, there are three main steps in computing the Steiner point of a 2-dimensional object. We will describe the algorithm in detail as follows.

Step 1. Load 2-dimensional data of an object. All data can be denoted by a set of points $S = \{p_1, \cdots, p_N\}$. We represent 2-dimensional data points by $p_i = (x_i, y_i)$ as Cartesian coordinates, or $p_i = (\alpha_i, r_i)$ as polar coordinates. For 2-dimensional binary image data, we can also use the matrix $p_i = (x_i, y_i)$ to express the image. Before this, if it is a color image, also the colors should be processed separately. The color image has to be segmented to the corresponding binary image, separating an object from others and finding its edges. Here we do not deal with this procedure and we begin with binary image of an object separated from others, so that its borders are known.

Step 2. Select a suitable number of extreme points to form a convex polygon. This can be divided into the following five parts.

1) Compute the mean point as the center (X_m, Y_m) of the original data set and transform two-dimensional Cartesian coordinates stored in corresponding elements of arrays X and Y into polar coordinates with origin in the mean.

2) Find an assistant convex polygon with L points, which is denoted by $\bar{A} = \{\bar{p}_1, \cdots, \bar{p}_L\}$ (in order to eliminate most inner points). Suppose L=8, for example. We can divide the unit disc into L equal parts with fan shapes, and each part has an angle $2\pi/L$. In the data set find L distinct points which satisfy

$$\bar{p}_i = \max_j \{\langle e_i, p_j \rangle | p_j \in S, j = 1, 2, \cdots, N\} \qquad (11)$$

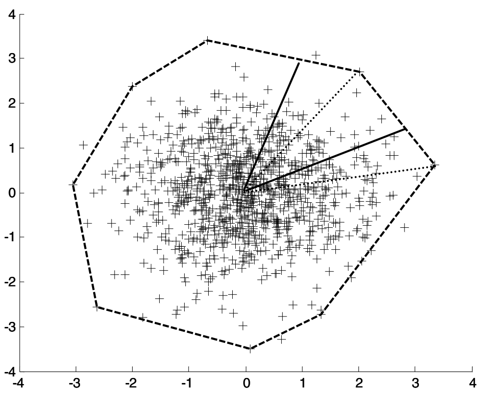

Fig. 2: An assistant convex polygon on given data.

where $i = 1, \cdots, L$, and $e_i = (\cos 2\pi i/L, \sin 2\pi i/L)$ is the unit vector with angle $2\pi i/L$. Figure 2 shows that the points connected to the mean point (X_m, Y_m) with the dotted lines satisfy equation (11).

3) Delete duplicate points generated by step 2). Sometimes, there are more than one e_i, for $i = 1, \cdots, L$, share the same points $\bar{p}_i$ which satisfy equation (11), and of course, these points are numbered sequentially. In this case, the first point is reserved and the duplicate ones should be eliminated in order to achieve a distinct set of extreme points. After this, the selected points form a set which is also denoted as $\bar{A} = \{\bar{p}_1, \cdots, \bar{p}_L\}$.

4) Based on the assistant convex polygon, eliminate all the inner points; i.e. delete those data points which satisfy the following condition

$$\max_j\{\alpha_{ij}\} - \min_j\{\alpha_{ij}\} > \pi \tag{12}$$

Here $\alpha_{ij} = \alpha_i - \bar{\alpha}_j$ for $i = 1, \cdots, N; j = 1, \cdots, L$. Clearly, all the eliminated points are inner points. The survival data set can be denoted as $\bar{S} = \{p_1, \cdots, p_M\}$, and usually $M \ll N$.

5) Based on the survival points set , using a strategy similar to step 4), select all the extreme points according to equation (10), where $p_{ij} = p_i - p_j$ for $i = 1, \cdots, M; j = 1, \cdots, M; j \neq i$. After that, we can form a convex polygon by ordering all the extreme vertices according to their polar coordinate angles, and denote it as $A = \{p_1, \cdots, p_M\}$.

Step 3. Compute Steiner point according to the following formula, see e.g. [11]

$$S(D) = \sum_{i=1}^{M} \psi(p_i, A)p_i\,, \tag{13}$$

where $\psi(p_i, A)$, for $i = 1, \cdots, M$, is the proportion to 2π of the external angle of convex polygon A at p_i.

4. Computational complexity analysis

There are two aspects we will discuss, namely time and space complexity. Let us consider the former firstly. The procedure of computing Steiner point for a 2-dimensional object is composed of three steps as in the last section, so the total computational quantity of time can be expressed as

$$T = T_1 + T_2 + T_3 \tag{14}$$

Here

T_1- time cost on loading 2-dimensional data of an object ;

T_2- on selecting extreme points to form a convex polygon of an object;

T_3- on calculating Steiner point based on the convex polygon.

For a given problem, T_1 is fixed and can be estimated as

$$T_1 = O(N) \tag{15}$$

Now we focus on the second and the third terms[3]. In the second term, there are five sub-steps in detail, therefore we can rewrite T_2 as follows,

$$T_2 = T_{21} + T_{22} + T_{23} + T_{24} + T_{25} \tag{16}$$

Here

T_{21}- on computing the average point of the data set and transform from Cartesian to polar coordinates;

T_{22}- on selecting an assistant convex polygon with L points;

T_{23}- on elimination of duplicate assistant polygon points.

T_{24}- on deleting inner points based on the assistant polygon;

T_{25}- on selecting the final convex points to form a polygon.

Clearly, for 2-dimensional data set

$$T_{21} = O(N) \tag{17}$$

While finding L maximum values in N points according to equation (11), it needs

$$T_{22} = O(LN) \tag{18}$$

[3]The repetition of step 2 or another procedure aiming at the selection of extreme points are crucial for the time complexity. So we focus on discussing step 2 in detail.

Elimination of duplicate convex points needs

$$T_{23} = O(L) \tag{19}$$

Deleting the redundant inner points costs

$$T_{24} = O(LN) \tag{20}$$

Similarly, searching M' maximum values of M survival points by equation (11) and ordering them, it needs

$$T_{25} = O(M(M-1) + M'\log(M')) \tag{21}$$

So we have

$$T_2 = O((1+2L)N + L + M(M-1) + M'\log(M')) \tag{22}$$

Considering that $L \ll N$ and $M' \leq M$, the last formula is approximately

$$T_2 \approx O(LN + M^2) \tag{23}$$

For step 3, the main cost lies in computing the external angle for each extreme point and it counts

$$T_3 = O(M') = O(M') \tag{24}$$

Totally, the sum of the computational complexity of time can be evaluated as

$$T = T_1 + T_2 + T_3 \approx O(LN + M^2) \tag{25}$$

So the complexity of this algorithm is nearly linear, and it should has fast speed under the condition that we choose a proper parameter L, not too large to decrease the computing time, and not too small to eliminate most of inner points such that $M \ll N$.

Now consider the space complexity, it is much easy to estimate the storage cost according to the three steps in the algorithm

$$S = S_1 \vee S_2 \vee S_3 \tag{26}$$

Here,

$$S_1 = O(N) \tag{27}$$

$$S_2 = O(N \vee (N+L) \vee (N+L) \vee M \vee M') = O(N+L) \tag{28}$$

$$S_3 = O(M') \tag{29}$$

In summary, the space cost in the procedure of the algorithm is as follows

$$S = O(N) \vee O(N+L) \vee O(M') = O(N+L) \tag{30}$$

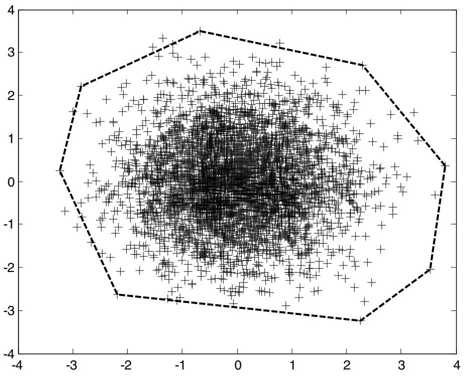
Fig. 3: Random data and assistant convex polygon.

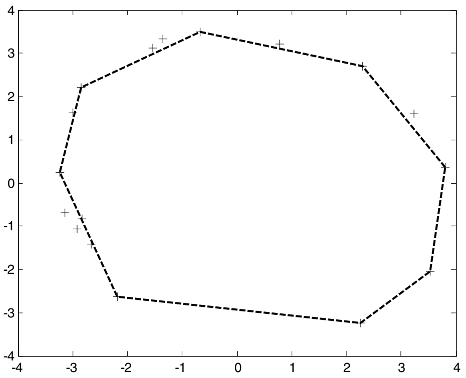
Fig. 4: Survival data after elimination of inner points.

5. Experimental examples

Following the algorithm given in section 3, this paper provides two examples for calculating Steiner point. Its purpose is to verify how to compute the Steiner point when given a 2-dimensional data set or a binary image. It focuses on computing the Steiner point both efficiently and precisely.

Example 1. Random data set in 2 dimensions.

Figure 3 is a random 2-dimensional data set with the Gaussian distribution. We set parameters $N = 4096, L = 8$; i.e. there are 4096 points forming a 2-dimensional object, we use an assistant polygon with 8 vertices to eliminate inner points. Figure 4 shows data after elimination of inner points with the help of the assistant polygon. Figure 5 shows the final polygon and its Steiner point. Figure 6 is the result for testing invariant property on Minkowski sum transform, in which the points marked by $'\oplus'$ are those after Minkowski sum transform.

Example 2. Image data set in 2 dimensions.

For 2D image problems, data points, with size $N = N_1 \times N_2$, the process can be deduced by eliminating inner points easily. In detail, for each row

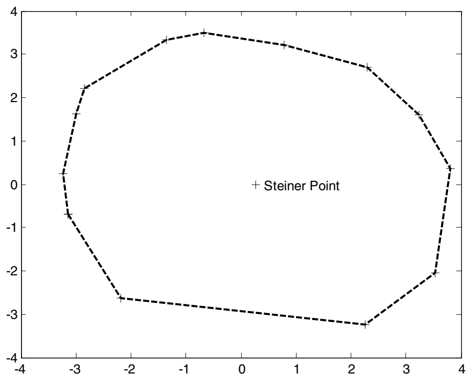

Fig. 5: Final convex polygon and the Steiner Point.

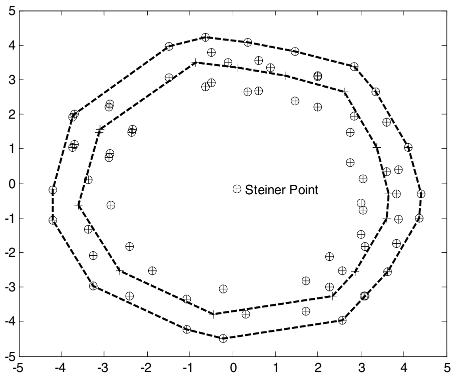

Fig. 6: Invariant test on Steiner point with Minkowski sum.

what we need is finding the leftmost and the rightmost point of the object. These need computational complexity of $O(N)$. After these processes, the survival points is with size of $O(2N_1)$, and the computing amount will be deduced sharply. Here we list parameters as $N = 15\text{K}$, $L = 6$. Figure 7 is the original image. Figure 8 shows edge points detected, whose scale is 615. Figure 9 shows the assistant polygon of the image, with which there are 77 points remained after elimination of inner points. Figure 10 is the final convex polygon and its Steiner point.

Fig. 7: A 2-dimensional image with one object.

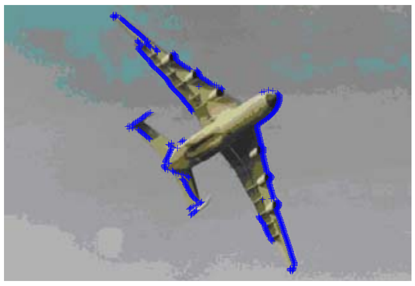

Fig. 8: Boundary points of the object.

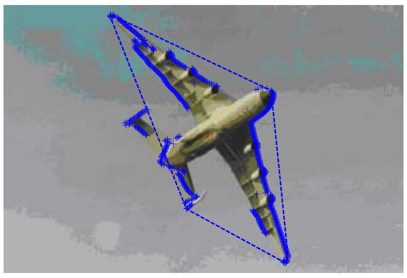

Fig. 9: Assistant polygon for the object in the image.

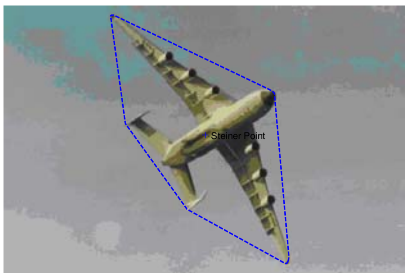

Fig. 10: Convex polygon and Steiner point.

6. Conclusion

There are several ways to calculate Steiner point of an object [12]. This approach provides one of the simplest methods on Steiner point calculation implementation and its application in image processing. Steiner point is a traditional geometry concept for centuries and some properties are known clearly. Based on these, a numerical method is proposed for calculating Steiner point of a 2-dimensional polytope. Also computational complexity is discussed in the text. Some experiments on random 2-dimensional data and 2D image processing are provided for testing the algorithm. Furthermore, for 3-dimensional objects, computing Steiner point is more useful in real life, but it is more complex to realize such calculation, especially for image processing problem, e.g. 3D virtual vision. In the 3D space, the most difficult problem may be memory and computational complexity. Tracking a moving object needs a fast algorithm to recognize and locate the object, so it must be fast enough to calculate the Steiner point in a very limited time. However, it is not easy to extend the current algorithm to 3-dimensional space, as it is too complicated to figure out the convex polytope and the external angle for each extreme point in 3 dimensions. It is an open problem for future study.

Acknowledgement

This work is supported by CMP laboratory, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University. The first author is supported by the Exchange Project Agreement between the Czech Ministry of Education and the Chinese Ministry of Education. Thanks to Tomáš Werner who provides us some suggestion on computing Steiner point and testing its invariant properties. Thanks to many other CMP members who have taken much time on discussions with us on Steiner points.

References

[1] http://*faculty.evansville.edu/ck6/tcenters/class/steiner.html*.

[2] http://*mathworld.wolfram.com/SteinerPoints.html*.

[3] http://*en.wikipedia.org/wiki/Minkowski_addition*.

[4] R. Honsberger, *The Steiner Point and the Tarry Point*, Nineteenth and Twentieth Century Euclidean Geometry, Washington, DC: Math. Assoc. Amer., pp.119–124, 1995.

[5] B. Grunbaum, *Convex polytopes*, second ed., Springer, New York, 2003.

[6] J. Mattioli , Minkowski operations and vector spaces. *Set-valued analysis* 3:33–50,1995.

[7] W. J. Meyer, Characterization of the Steiner Point. *Pacific Journal of mathematics*, 35(3), 1970.

[8] R. Mondaini, D. Freire Mondaini and N. Maculan, The Study of Steiner Points Associated with the Vertices of Regular Tetrahedra Joined Together at Common Faces. *Investigacion Operativa*, 6(1):103–110, 1998.

[9] R. Schneider, On Steiner points of convex bodies. *Israel J.Math*, 9:241–249, 1971.

[10] R. Schneider, *Convex bodies: the Brunn-Minkowski theory*, Cambridge University Press, Cambridge,1993.

[11] G. C. Shephard, A uniqueness theorem for the Steiner point of a convex region. *J. London Math. Soc.*, 43: 439–444, 1968.

[12] S.R. Sternberg, Grayscale Morhpology, Computer vision. *Graphics and Image procession*, 35: 333–355, 1986.

[13] T. Vetterlein , M. Navara, Defuzzification using Steiner points. *Fuzzy Sets and Systems*, 157: 1455–1462, 2006.