Proceedings of the Eighth IASTED International Conference
SIGNAL AND IMAGE PROCESSING
August 14–16, 2006, Honolulu, Hawaii, USA
ISBN Hardcopy: 0-88986-583-3 / CD: 0-88986-585-X

AN IMPROVED ALGORITHM OF VIEWPOINT SPACE PARTITION IN 3D OBJECT RECOGNITION APPLICATION

Yuan Luo, Huimin Ma, Fengting Li
Dept. of Electronics Engineering
Tsinghua University, Beijing, 100084
China
lybachuss@yahoo.com, mhmpub@tsinghua.edu.cn , lift@tsinghua.edu.cn

ABSTRACT
Aspect graph is an important method in 3D object recognition. To effectively compute the viewpoint space partition (VSP) is critical in acquiring a good aspect graph. In this paper, we present improvements to the VSP algorithm under perspective model: computing EV (Edge Vertex) events using triangle face feature; simplifying VSP representation; boundary intersection updating; closed loop region computing; generating multi-resolution VSP. We show the partition results, which demonstrate our algorithm's effectiveness and fastness (less than 2 min). Based on the aspect graph constructed using the improved algorithm, we perform 3D object recognition, which matched all tested objects to the correct counterparts in database.

KEY WORDS
Aspect graph, viewpoint space partition, critical events, and 3D object recognition

1. Introduction

The aspect graph was first proposed in [1] in 1976, and its application to 3D object recognition has been recognized an effective approach [2]. It provides complete topological information of the target object, which is located at origin and viewed from all directions from view points in space. In our work, we adopt the perspective projection model, and the VSP is modeled as Gaussian ball with appropriate radius[1]. The viewpoint space is then partitioned into finite number of areas. The projections from view points in the same (different) areas should be same (different) regarding topological structure. Hence there is exactly one projection for each area (such projection is called prototypical projection or stable view). Then we can construct the object's associated aspect graph, where every node consists of one partitioned area of VSP and the prototypical projection associated with this area. The constructed aspect graph also maintains view adjacency information: two stable views are said to be adjacent when it is possible for a moving viewer to pass from one view to the other without intervening stable views; and adjacent views are joined by an edge. Thus we can reduce the problem of 3D object recognition to matching a sequence of 2D projections.

Of all above, the most effort demanding task is partitioning the viewpoint space. Generally, there are two types of approaches:

(1) One is the theoretical approach based on catastrophe theory. The basic idea of this method is to first establish the analytical equations for all possible visual events (see [4] [5], [6], [7], [8] and [9]), and then solve these equations to obtain the critical regions (boundary of partitioned areas) and acquire the partition of the viewpoint space.
(2) The other one is to use the clustering algorithm based on measurement of distance of similarity (see [3]). This approach first equally divides the Gaussian ball to n areas, and then generates projection with viewpoint located at center of each area. Next it clusters those projections, merges the regions whose projections' similarity differences are less than threshold, deletes projections of old regions and then generates new projection with viewpoint located at center of merged region. It repeats doing so until all adjacent regions' similarity differences are less than threshold. It then considers the remaining projections as prototypical projections. This approach typically requires less number of prototypical projections than theoretical method.

* This work is supported by the National Natural Science Foundation of China (No. 60502013)
[1] Usually 10 times larger than object's diameter in our implementation.

2. Modeling and construction of real objects

In this paper, we construct the 3D object using mesh model. The mesh model of an object consists of geometry data (3D points associated with color information, reflectance data) and texture data. 3D points are connected, making the mesh grid look like a collection of structured triangles. Reflectance data are mapped to the surface of 3D points and rectified, generating object view without texture. Then texture data, if presented, are mapped to object using image rendering technique.

Fig.1 (a) bomber T:680 V:424

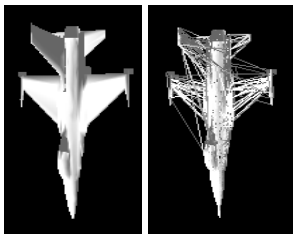


Fig.1(b) F-16 T:2514 V:1174

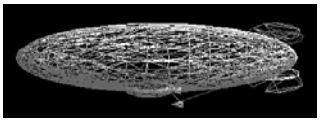


Fig.1 (c) blimp T:1382 V:795

Fig. 1 shows 3 examples of 3D objects. For each object, the left image is its solid view; the right is mesh grid view[2]. T is the number of object's triangles; V is the number of vertices.

As we see, the object surface consists of thousands of triangles and vertices, thus it can be considered as a complex non-convex polyhedron, all of whose faces are triangles. In our implementation, mesh model is saved as [object].obj[3] file, which serves as input for computing VSP. In order to consider VSP of such polyhedron, we only need to consider EV events and EEE (Edge-Edge-Edge) events. Both of those events have straightforward and clear analytical equations.

3. Algorithm overview

We provide an improved technique combining the above visual event approach and clustering method to partition viewpoint space. We use perspective model and rectangular coordinate system.

[2] Our mesh model data is acquired from www.quick3d.org

[3] Object is an optional file name

Fig.2 is the flowchart of the algorithm which synthesizes our improvements. Step 1 imports 3D object geometry information from .obj file. Step 2 renders 3D view using texture data. Steps 3-5 are the core parts of algorithm. They are further described in part 4. Step 6 is where clustering method finds itself.

Under perspective model, space complexity of VSP for non-convex polyhedral is $O(n^9)$, time complexity of VSP for non-convex polyhedral is $O(n^9 \log n)$.

The rest of the paper is organized as follows: in part 4, we propose our improvements to the algorithm; in part 5 we show our experiment on VSP and its application results to real 3D object recognition; finally part 6 is the conclusion of our work.

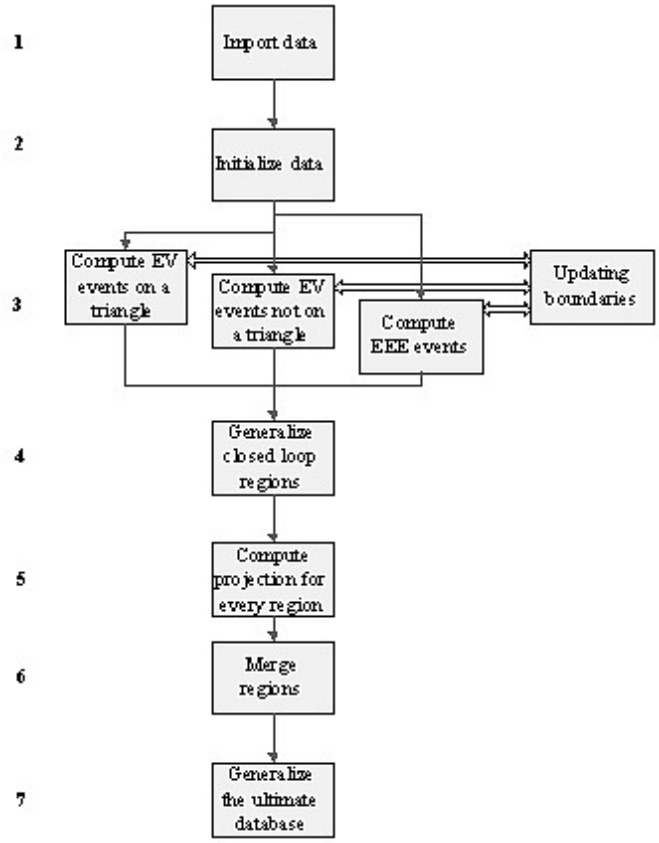


Fig.2 flowchart of improved algorithm

4. Improvement to VSP algorithm

[4], [5] modeled viewpoint space of perspective model as cubic, and provided the analytical equations for the EV events and EEE events. However, we find following problems:

- Their algorithm shows redundancy in computing EV

events.

- Their algorithm's boundary updating strategy has complexity of $O(n^6)$, which could be reduced to $O(n^4)$.
- Besides, their method cannot be applied into multi-resolution model.

In this paper, we propose our improvements to VSP based on the mesh model data. Our improvements to steps are listed according to the steps' appearing order in computing.

4.1 Compute EV events using triangle face feature

Assuming the vertex and edge of an EV event is v and e = (a, b) respectively. Generally, the critical regions of EV events have no definite relationship, and we have to consider interaction for every pair. In our mesh model data, thousands of triangles contribute large amount of EV events. We consider critical regions rising up from a triangle face and find the relationship below.

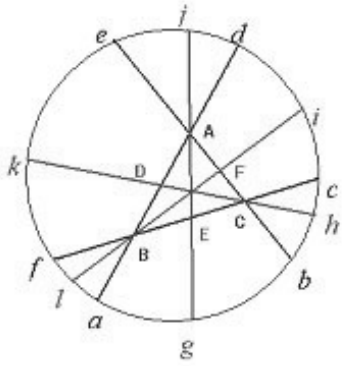


Fig.3 EV events in a triangle plane

In Fig.2, let the triangle face be ΔABC, the midpoint of three edges are *D*, *E*, *F* respectively. The intersection of ΔABC plane and Gauss Ball is a circle. The extensions of three edges meet the circle at points *a*, *b*, *c*, *d*, *e*, *f* respectively, which partitioned the circle into six arcs. These 6 arcs are all boundary segments and can be computed simultaneously. Since these arcs are coplanar, we do not have to consider their intersections. This feature benefits the updating procedure in that it reduces significantly the number of arcs to be considered for intersecting. Thus using this feature effectively reduces computing time.

4.2 Simplifying VSP representation

To represent VSP, we have to first represent boundaries (critical regions). It is easy to see that EV events' critical regions on Gaussian ball are arcs, which are easy to compute and represent. While for EEE events, critical regions are intersections between ruled surfaces and Gaussian ball. Although we could easily find the coefficients of their equations, the critical regions will have to be represented as solution of the equation set:

$$c_1x^2 + c_2y^2 + c_3z^2 + c_4xy + c_5xz + c_6yz = 0 \quad (1)$$

$$x^2 + y^2 + z^2 = R^2 \quad (2)$$

(1) is the ruled surface equation and (2) is Gaussian ball equation, assuming the Gaussian ball radius is R. It is hard to solve the 3-unknown quadric equation set directly. However, in order to generate aspect graph, what we actually need to store in a node are the prototypical viewpoint and its associated projection. It is enough to determine whether a given viewpoint (on the Gaussian ball) is to the left of the ruled surface or to the right. We define a particular viewpoint to be on the left (right) of the surface if $c_1x^2 + c_2y^2 + c_3z^2 + c_4xy + c_5xz + c_6yz$ is greater (less) than 0. Thus it is enough for us to store the EEE ruled surface equation as EEE critical regions. In our algorithm, two data types are needed to represent the boundaries: one is the arc, the other is the ruled surface equations (actually the coefficients are stored in an array). These two types of data are the outputs of step 3 in the flowchart.

4.3 Boundary intersection updating algorithm

In [4], the boundary intersections are computed after step 3. There are $O(n^3)$ boundaries and each boundary may intersect with others, thus time complexity for computing boundaries is $O(n^6)$. We propose to update boundary intersection while computing new boundaries (see Fig. 2 updating boundary module). We maintain and update two linked lists, one for current boundary set while the other for intersections on the newly computed boundary. Algorithm is outlined as follows:

(1) Compute a new boundary <*cd*>.[4]
(2) For each boundary <*ab*> in current boundary set *C*, determine whether it intersects with the new boundary.
(3) If <*ab*> and <*cd*> have one intersection *e* (see Fig. 4), then delete <*ab*> from *C*, add <*ae*>, <*be*> to *C*, and add *e* to <*cd*>'s intersection list *iList*.

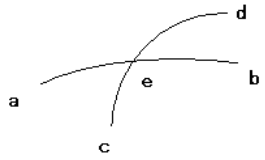


Fig. 4 having one intersection

[4] Without lost of clarity here, we simply denote a boundary by its two end points.

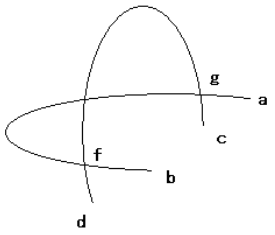


Fig. 5 having two intersections

(4) If <*ab*> and <*cd*> have two intersections *f*, *g* (see Fig. 5), then delete <*ab*> from *C*, add <*ag*>, <*gf*>, <*fb*> to *C*, and add *f*, *g* to <*cd*>'s intersection list *iList*.
(5) After scanning all boundaries in *C*, sort intersections on <*cd*> and compute segments which these intersections divide <*cd*> into. Then add these segments to *C*.

As the procedure goes, we only need to consider <*cd*>'s intersection(s) with 1, 2, 3, … other boundaries. Since there are $O(n^3)$ boundaries, algorithm's time complexity is $O(1+2+...+n^3)$, which is in fact $O(n^4)$.

4.4 Compute closed partition regions

This corresponds to step 4 in flowchart. The basic idea is to connect the boundaries to form closed loops. Since every boundary has only two adjacent regions, if a boundary has been used twice, it can be deleted from boundary set. As connecting goes on, boundary set *C* will be smaller and smaller, until it turns empty. The algorithm is outlined as follows:

(1) Initialize a new region *nr* whose boundary list is empty.
(2) Starting from the first boundary *b* in *C*, add it to *nr*'s boundary list, update *nr*'s open endpoints *e1, e2* to be *b*'s endpoints *eb1*, *eb2*. Increase *b*'s using time by 1; if it becomes 2, delete *b* form *C*.
(3) While *nr* is not closed (*e1* is different from *e2*), repeat searching *C* for next boundary *b* (one of b's endpoints is the same as *e1* or *e2*). Increase *b*'s using time by 1; if it becomes 2, delete *b* form *C*.
(4) While *C* is not empty, repeat (2)-(3).

Each time we search *C*, the number of boundaries will decrease by 1, thus the time complexity is $O(1+2+...+n^3)$, which is $O(n^4)$. The space complexity, which is the number of *nr*'s boundaries, is $O(n^3)$.

4.5 Generate multi-resolution VSP

This corresponds to steps 5-7 in flowchart. Our previous work [10] indicates that multi-resolution aspect graph can significantly benefit 3D object recognition. In order to generate hierarchical VSP, clustering method is used to merge the regions with indistinguishable projections under current resolution scale and threshold. In order to measure the distance between projections p1, p2 of the object, we choose Hausdorff distance [11] of the form Hd(p1, p2). By setting different resolution scale, we finally partition the viewpoint space at multi-level and generate a hierarchical aspect graph. The algorithm is outlined as follows:

(1) Compute prototypical projection for each VSP region. We use OpenGL 3D view generating facility: the input is prototypical viewpoint for each VSP region, then OpenGL generates corresponding 3D view, which is stored as prototypical projection.
(2) Such generated prototypical projections and their associated viewpoints consist of aspect graph at infinite resolution level. However, in practical usage, we could only achieve finite resolution. Suppose the highest resolution level is r_0, then repeat merging those regions with prototypical projections p1, p2 such that Hd (p1,p2)< r_0 (i.e. indistinguishable), and adopt p1(or p2) and its viewpoint as the new region's prototypical projection and viewpoint. Thus, we acquire a more cursory VSP. For example, in Fig. 6, six small regions merge into one large area (indicated by shadow).
(3) Generate a descending sequence of resolution scale $\{r_i\}$, (i=1,…,n), $r_i = (d_i/d_0)\ r_0$, where $\{d_i\}$ is increasing sequence that denotes distance from viewpoint to object (d_0 is the baseline distance that varies between difference objects. For our experiment below, it is set to 1km, r_0 is the resolution when distance equals d_0 and it can be acquired by experiment). For different r_i's, merge regions as (2) describes. Thus we acquire a multi-resolution VSP.

Fig.6 Merge regions

In practice, r_0 and $\{d_i\}$ are manually set depending on factors such as sensor quality and object size. For details, please refer to [10].

5. Experiment and discussions

Using the algorithm described above, we construct the aspect graph database for 7 objects belonging to 4 different types, which are:

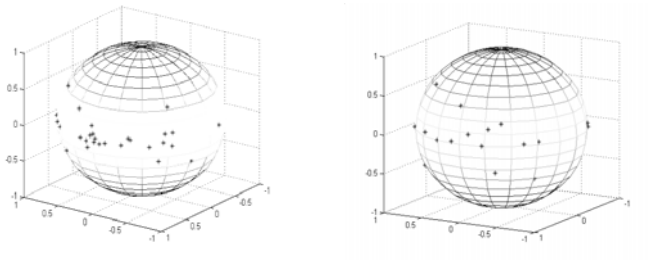
(1) Airplane: F-117 (T:1791, V:921)
F-4 (T:642, V:214)

	F-16 (T: 2514, V:1174)
	bomber (T:680, V:424)
(2) Airship:	blimp (T: 1382, V:795)
(3) Automobile:	car (T:1416, V:449)
(4) Missile:	generic (T:852, V:445)

For every object, the database consists of the following:

a) VPS (viewpoint space) file, consisting of the prototypical viewpoints for partitioned regions;
b) The prototypical projection associated with each viewpoint in VPS file.

We perform the 3 resolution level VSP on Pentium IV 2.4GHz CPU, and for every object, the VSP takes less than 2 minutes.



(a) 41 viewpoints total (b) 24 viewpoints total

Fig.7 the distribution of all the viewpoints for one object

Fig. 7 (a), (b) show the distribution of prototypical viewpoints for F-4 object on normalized Gaussian ball at resolution level r_0 (r_0=0.001d, d is object's diameter) and r_2 (=$2r_1$=$4r_0$) respectively. Fig. 8 shows the prototypical projections and silhouettes of F-4 corresponding to the viewpoints at resolution level r_2. Note that the airplane spans a large area in XY plane, while its Z-axis span is relatively short. Namely, its horizontal information is abundant while vertical information is much less, which corresponds to the viewpoint distribution (dense horizontally while sparse vertically).

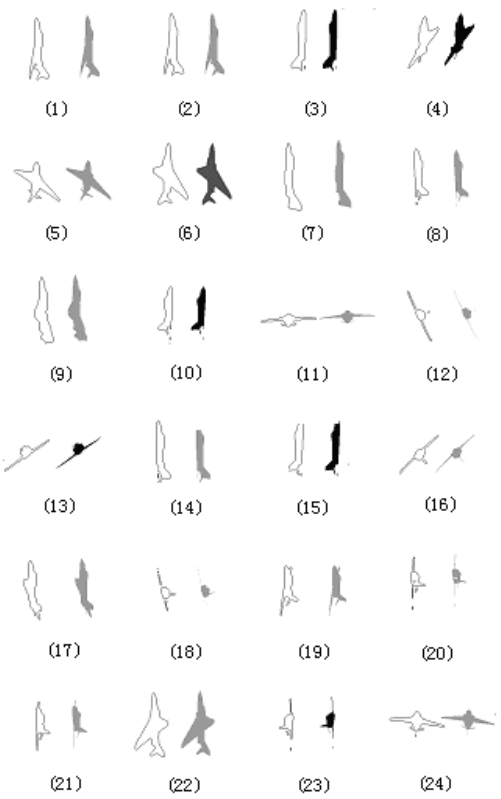


Fig.8 F-4 plane's projections and silhouettes corresponding to the viewpoints in Fig.7

UNs	database	viewpoint			distance
f117_un	blimp	1406.2	442.5	86.3	17
	Btm.	1112.7	229.9	156.6	11
	f117	**9**	**39.9**	**-2.1**	**0**
	bomber	6.8	1493	8.7	19.8
	f4	-7.4	43.4	1.1	9
	Generic	-533.0	1174.7	-6.3	16.1
	f16	202.2	198.4	-30.7	21.4
f4_un	blimp	-133.	637.9	-60.5	13.0
	Btm.	897.1	-535.1	473.8	18.4
	f117	-20.8	35.4	-0.2	21.9
	bomber	459.1	1420.6	-7.6	42
	f4	**8.3**	**43.2**	**0.9**	**10.3**
	Generic	304.7	1254	-7.6	18.2
	f16	202.3	198.4	-30.7	45.4
generic_un	blimp	698.4	1295.3	125.9	16
	Btm.	1132.4	152.9	99	9
	f117	-39.9	9.1	1.5	5
	bomber	712.3	1312.1	-9.5	6.3
	f4	-16.2	40.9	1.1	8.2
	Generic	**-9.2**	**1289.9**	**-12.6**	**4**
	f16	-53.1	233.4	-154.8	12
f16_un	blimp	-238.7	1445.5	187.7	15.1
	Btm.	-204.3	1104.7	-231.59	12.1

	f117	-38.1	14.9	-2.3	4.1
	bomber	113.9	1444.6	-359.6	9.1
	f4	22.2	35.6	-13.2	7
	Generic	96.6	1286.3	-15.3	5.4
	f16	**235.6**	**160.3**	**6.5**	**3.8**
Btm_un	blimp	1309.4	588.5	-347.4	13
	Btm	**1147**	**1.8**	**-3.4**	**5.1**
	f117	-40.1	-8.1	-1.8	15.2
	bomber	1290	751.52	-10.7	33.4
	f4	-37.2	23.5	0.1	13
	generic	-733.7	2020.2	55.6	31.8
	f16	242.9	148.3	-16	43.2
blimp_un	**blimp**	**1406.5**	**442.5**	**86.3**	**7.3**
	Btm.	993	549.7	-165.9	13.2
	f117	-28.6	28.2	-8.3	22.2
	bomber	1290	751.5	-10.7	39.4
	f4	-32	22.5	-20.1	23.9
	generic	-31.8	2149	55.9	36.2
	f16	284	2.4	-24	47

Table 1 Match Result

We test 6 unidentified projections (UNs), in each of which object's attitude is random. We match the UNs to the projections in our database based on Hausdorff distance [11]. The result is shown in Table 1. The distance is the minimum distance from the unknown projection to the projections in certain database at resolution r_0. Details on how multi-resolution matching progresses and how Hausdorff distance is used are presented in [10].

In Table 1, all six unidentified projections are matched to the correct database, the distance from whom is obviously smaller than other databases. The result shows that based on the database constructed using the improved algorithm, we can distinguish not only objects from different types (e.g. blimp belongs to airship, car belongs to automobile, generic belongs to missile etc.), but also objects from the same type (i.e. distinguish different airplane objects).

6. Conclusion

In this paper, we improve the algorithm of VSP, and show the result of the partitioned space and test the algorithm in our object recognition experiment. Our improvements focus on computing EV events using triangle face feature; simplifying VSP representation; boundary intersection updating; closed loop region computing; generating multi-resolution VSP. The visualized VSP indicates agreement with object's characteristics our intuition. The experiment shows that using the database generated by improved VSP, we can not only recognize target from different types of interfering objects, but also recognize target from interfering objects that belong to the same type of target.

References

[1] J. J. Koenderink and A. J. van Doorn. The singularilarities of the visual mapping. *Biol. Cyber.*, 24, 1976, 51–59.

[2] Robert D. Schiffenbauer, A survey of aspect graphs. *http://citeseer.ist.psu.edu/schiffenbauer01survey.html*, *Technical Report, TR-CIS-2001-01*, Department of Computer and Information Science, Polytechnic University, Brooklyn, Long Island, Westchester., 2001.

[3] Cyr, C.M.; Kimia, B.B., 3D object recognition using shape similarity-based aspect graph. *Proc. of Eighth IEEE Conf. on Computer Vision*, Vol.1, Vancouver, Canada□2001, 254 – 261.

[4] Gigus Z., Canny J. and Seidel R., Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6), 1991, 542 – 551.

[5] Gigus Z. and Malik J., Computing the Aspect Graph for Line Drawings of Polyhedral Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2), 1990, 113 – 122.

[6] H.B. Zhang, Partitioning Viewpoint Space for the Aspect Graph of Polyhedral Objects under Perspective Projection. *ACTA ELECTRONICA SINICA*, 23(12), 1995, 23-28.

[7] J.X. Zeng, Y.M. Lu, M. Li, J. Chu, A Representation Method Based on Aspect Graph for 3D Objects. *Journal of Image and Graphic*, 7(A)(9), 2002, 906-910.

[8] H.B. Zhang, Dissection of E^d by a Finite Set of Hyperplanes and Its Applications. *Chinese Journal of Computers*, 17(9), 1994, 697-702.

[9] Maha Sallam, John Stewman, Kevin Bowyer, Computing the Visual Potential of an Articulated Assembly of Parts. *Proc. of Third International Conference on Computer Vision*, Osaka, Japan 1990, 636-643.

[10] Luo Yuan, 3D object recognition based on aspect graph. *Undergraduate Dissertation*, Tsinghua University, 2005.

[11] D.P. Huttenlocher, G.A. Klanderman, &W.J. Rucklidge, Comparing images using the Hausdorff distance *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9), 1993, 850~863.