Road Structure Feature Extraction for Urban Aerial Image Matching

Road Structure Feature Extraction for Urban Aerial Image Matching

Hu Shuo
Graduate School of Computer and Information Sciences
Hosei University
Tokyo 184-8584, Japan
shuo.hu.56@stu.hosei.ac.jp

***Abstract*—Aerial photo matching is one of the key technologies indispensable in various services using GIS (Geographic Information Systems), and is widely applied in many areas including geography analysis, rescue, and city planning. This paper presents the design of an automated aerial photo discovery system, in which a match with a given photo, with unknown location orientation, and scale, is to be identified in a large photo image database. In this paper, we propose an approach which uses the road structure for matching, and present our implementation, test results, and our analysis on the accuracy and the robustness of our approach.**

Keywords*—Aerial image; Content-based image retrieval(CBIR); Convolutional neural network(CNN); road detection; road structure; image retrieval;

I. INTRODUCTION

Aerial image database is an essential infrastructure for various services using GIS (Geographic Information Systems), such as geographic analysis, disaster relief, rescue, and city planning. In addition to those governmental, business, and professional uses, it is nowadays also available to public through Web-based map services such as Google Map. Aerial image discovery is one of the key technologies indispensable in various services using GIS, in which a match is to be identified in a large aerial photo image (file image) database with a given aerial photo (query image), with unknown location, orientation, and scale.

There are lots problem in aerial image matching. As we know, some image retrieval system use the bag of the visual word [1] to do image searching. For example, to mark a image with some tags like 'dog', 'yellow', 'Akita', using machine to do auto classification and add tags. Then we also apply this process to the query image. Then the image matching problem changes to the tags matching problem. And some image retrieval system use k-means [2] to do classification, to classify the image into different categories, use it to simplify the search process. And some image retrieval use perceptual hash algorithm [3] , this algorithm make a fingerprint for each image, and use the hamming distance to calculate the similarity of two picture.

But all the algorithm and approach can not apply when we search the aerial image, because the aerial is almost the same, especially when the aerial images come from same area. Each picture has same element like roads, houses, blocks, we cannot use bag of the visual word, or the k-means, because they are in one category, and also cannot use perceptual hash, because they looks very similar. So, we decide to use some simple and consist things as the feature to do the searching, like the roads structure.

Supervisor: Prof. Kaoru Uchida.

Convolutional neural network (CNN) can be trained as robust feature extractors from raw pixel values and at the same time [4], learn classifiers for object recognition tasks, repressors for human pose estimation tasks [5], or mappings for semantic segmentation task [6]. Consider the power of CNN in image classification and object recognition, we can also use CNN to predict the road structure, then we can use this structure feature to do faster and more accurate search on a very big database.

II. RELATED WORK

Extracting road and building from aerial image is very important in this aerial image search system as well as other applications. To let this task can be automatically done, lots of people work on using local image features to classify each pixel or segment to be an object label. Bischof et al. and Boggess explored adding contextual information by using spectral values from a small patch centered at the pixel of interest as the input to a neural network, allowing it to learn some contextual features. However, such features were still very local since they used at most a 7 by 7 window for context due to the low-speed device in that age. Haralick et al. [7] introduced a popular set of features derived from gray level spatial dependence matrices Hd,θ, where H(i, j)d,θ specifies the frequency at which gray level values i and j co-occur at distance d and angle θ. Statistical quantities derived from Hd,θ were shown to be good for discriminating between different types of textures.

Mnih [8] use Restricted Boltzmann Machines (RBM) to do road and building extraction. They firstly divide the input aerial image into 64 x 64 patches and apply Principal Component Analysis (PCA) to get a PCA vectors, which is used for RBM, the output of the RBM is the label images. Shunta Saito [9] also use CNN to learn mapping from raw pixel values in aerial imagery to three object labels (buildings, roads, and others), generate three-channel maps from raw aerial imagery input. They take a patch-based semantic segmentation approach, so they firstly divide large aerial imagery into small patches and then train the CNN with those patches and corresponding three-channel map patches.

III. PROPOSED APPROACH

A. *System's basic architecture*

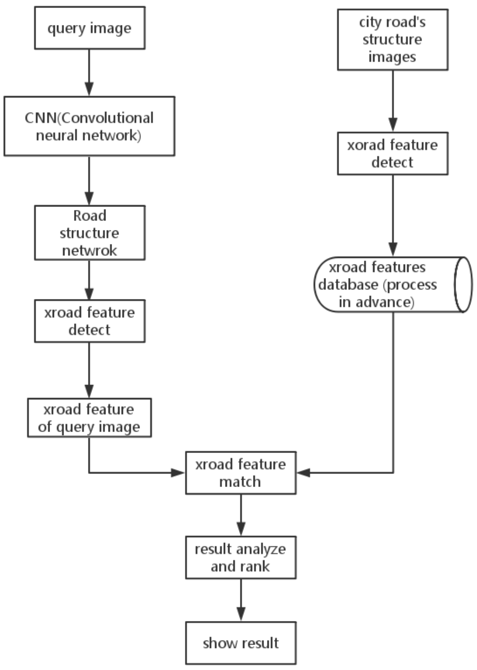
query image
CNN(Convolutional neural network)
Road structure netwrok
xroad feature detect
xroad feature of query image
city road's structure images
xorad feature detect
xroad features database (process in advance)
xroad feature match
result analyze and rank
show result

Fig. 1. System's basic architecture

As we can see, firstly the roads' structure of database from Google map can be directly used with Google map API. For every road structure we generate their xroad feature (XF), which means crossing road feature. Then for the query image, we apply CNN to find their road structure. We match the xroad features of query image and the database's xroad features, rank it with the similarity algorithm. Finally show the best several image and coordinate of the result.

The CNN road structure detector take only 1 or 2 seconds to analyze the query image, So main time would be cost in the next steps.

B. *Basic flow of roadmap matching based on xroad feature.*

Firstly we find the cross point in the map structure network, we use an algorithm to find the pixel coordinate in which in a cross points. Then, we cluster the coordinate found, and determine one coordinate which can represent the cross point. Lastly we use Breadth-first search (BFS) to find the relationship between cross points, to see whether they are connected, and generate the xroad feature to do matching with the database's xroad features.

Road structure match flows as follows:

1. cross point detect
2. cross point cluster
3. cross point network generate
4. xroad feature generate
5. xroad feature matching with xroad feature database

Here is the analysis flow for a query image.

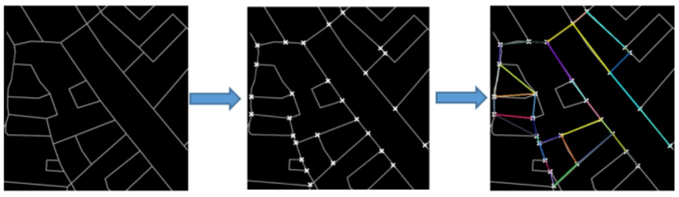

Fig. 2. Analysis flow for a query image

C. *Roads structure network feature*

What we want to do is to find the relative relationship of the cross point and use it as the matching feature. And the most important thing is that we should ensure the feature is scale-invariant and rotation-invariant. So let's focus on cross point A.

In our approach, we firstly connect the cross points if they are directly connected.

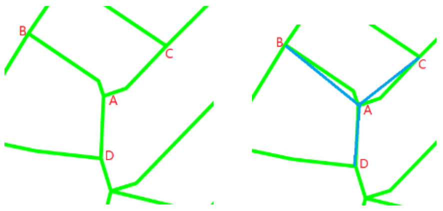

Fig. 3. Orignal cross point (left); Connect A with points which are directly connected with point A(right)

Then we calculate the angle $\angle$BAD and the proportion of the length of AB, AD, and ensure proportion larger than 1.In this case angle= $\angle$ BAD = 127° , proportion = max (AB/AD, AD/AB) = 1.4

So, this two information, the angle and the ratio of the two rays of the angle, is the feature of an angle. For every two rays in a cross point, we calculate all the angle feature, and make them as a list, and sorting in ascending order by the angle size, this list is the xroad feature of the cross point.

We should notice this, for a cross point, the number of angle feature equal to C_n^2, in which n is the number of rays of the angle, for example, in Fig. 3, the number of angle feature is 3.

D. *Robustness*

The robustness is very import in this aerial image matching system. Because it's likely that the result of CNN analysis would not be exactly the same as the standard map structure, maybe add some roads or miss some roads. We have to make it can be found when some road is error. And also the query image's rotation and the scale we cannot know, the feature should satisfy the scale-invariant and rotation-invariant.

1) *Resize and rotation*

Because the angle and proportion would not change, even if you change the rotation and resize the picture, obviously, the xroad feature would not change as well.

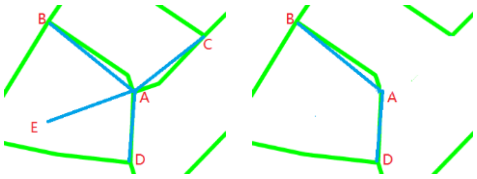

Fig. 4. Add a hypothesis cross point E and road AE (left) After remove one road (right)

2) Get one more cross point

What if we get one more cross point which in fact is a wrong cross point? In this case, if the E point is connect to A point mistakenly, then we get 3 more angle feature in the xroad feature, the number of xroad feature increase to 6, in which we still have 50% correct information considering the angle feature in xroad feature is independent . With this 50% correct information, we can at least know whether it may match or not.

3) Loss one line

In this case, obviously the number of angle feature of xroad feature would be decrease to 1 and we still has 33% correct information.

E. Xroad feature matching process

In the matching process, we choose every cross point Q from query image. And for every cross point F in the feature database, we calculate the similarity between P1 and P2. The calculation of similarity of two cross point is show in below:

$$S(Q,F) = \frac{\sum_{i=0}^{q}\sum_{j=0}^{f} dis(Q_i, F_j)}{C_q^2}$$

In this formula, q, f equals to the number of xroad features that cross point Q, F has. Q_i, F_j are the i-th xorad feature in Q, j-th xroad feature in F.

$$dis(Q_i, F_j) = \max(0, 100 - K\sqrt{N(Q_{i1} - F_{j1})^2 + M(Q_{i2} - F_{j2})^2})$$

$dis(Q_i, F_j)$ is to measure the distance of two xroad feature Q_i, F_j , K , N and M are the constant number to adjust the precision.

We can calculate and accumulate the match point for everyone in the database, then we sort the match point from high to low, choose the some of the result and show. Because the even though every road's structure network is not the same, but when the size of database become larger, it's easy to overlap for angles. So, this method is mainly use to preselect the candidate, then apply the traditional method like Scale-invariant feature transform (SIFT) [12] or Accelerated-kaze features (AKAZE) [10] on the original image and original dataset to do the final search.

IV. System Implementation

A. Cross point detection

In this part, we introduce an algorithm to auto detect the cross point. The input is road structure network like pic x. Then the output would be the coordinate of cross point. Even though in the image we can easily find the cross point, but for computer it's a hard work.

Fig. 5. A cross point's enlargement picture

In this step, I use a rotatable line to detect the cross point. And make a virtual square (of course you can use a circle, using square is just for easier to code) around A, then for each pixel B in the square's edge we confirm that if all the pixel in the line AB have color.

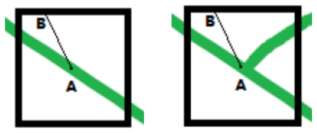

Fig. 6. Rotate line AB in a stratght line and a cross point

We can easily rotate clockwise the line AB and count the number of the times when it leave the green line. If the times is more than 3, which means there is a 3-cross point.

B. Cross point cluster

In the detect step, we can get lots of point which is satisfied with the last step's condition. Because for one cross point we there are 10 or more point can be found. So we have to cluster them into several class and pick a delegate for each cross point.

But in this problem we cannot find a 'k' to do k-means or other algorithm which need the number of clusters. This time, we use Canopy clustering algorithm [11]. This algorithm is simple and powerful.

Then for each class, we calculate the average coordinate as cross point coordinate in this class.

C. Cross point network generation

Now we get the coordinate of each cross point, then we should judge whether two cross point is connected or not. In this problem we can use BFS (Breadth-first search) to solve.

Firstly we build a virtual map which has the same size of query image, then give each cross point's coordinate an ID, and set other coordinate to zero. Then we apply the BFS like this.

```
Initialize virtual matrix M as process above
Add all the cross point's coordinate (x, y) in to the queue Q
While Q is not empty:
    X, Y = Q.pop ()
    For x, y around X, Y
        If M(x, y) is 0 and image(x, y) has color:
            Q.push(x, y)
            M(x, y) = M(X, Y)
        Else if M(x, y) is the ID of other cross point:
            Cross point M(X, Y) is connected with cross point M(x, y)
```

After the BFS, we can easily to get cross point network, and this algorithm is very fast, time complexity is O(n) in which n is the pixel number of colored point.

V. Experimental results

A. Dataset

In this paper the CNN part is incomplete (in fact we have a unperfected CNN to do the structure detect, but it still not good enough), so I used roadmap from Google Maps to demonstrate the feasibility of matching based on Road structure.

Example of apply CNN on query image:

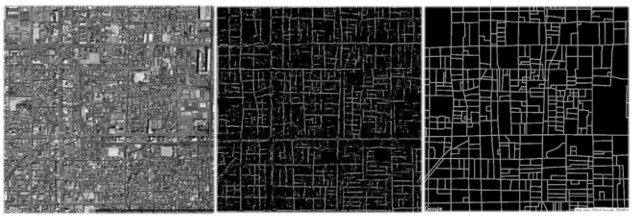

Fig. 7. From left to right: the original image, CNN's predict structure,standard road structure from Google map

To be used as file and data in image database and also for evaluating the performance, we need a large scale collections of pairs of aerial photos and corresponding structural feature labels.

Although there are several publicly available aerial photo databases, they are collected abroad, and most of them consist of photos taken in rural areas, thus there is not sufficient dataset especially for cities.

For this reason, we developed an automatic data collection system which collects and downloads publicly available data via Web API.

In the first experiment, we use the image that has the same scale with the database as query dataset. The second experiment we will use the a bigger scale dataset as query dataset, which is 2 times larger than the image in database, which means, for every query image, there are 4 image in the database can match the query image. Then we can know how powerful robustness in different scale this approach has according to the comparison of this two experiment. And next I also introduce the relationship of matching time and the number of the cross point in the query image.

B. Xroad matching speed

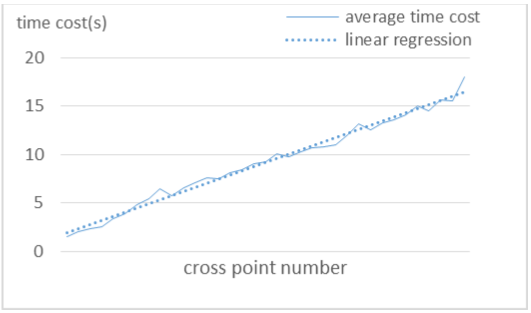

Fig. 8. Relationship between average time cost and cross point number

For each query we calculate the time cost, Fig. 8 shows the relationship between average time cost and cross point number. We can see it's almost a linear line, the dotted line show the result of linear regression of the data. Obviously, number of the cross point and the time cost are the linear relationship.

C. Robustness against rotation and resize

In this part, we download 1840 pictures which can cover the whole Kyoto as database, and random choose 466 as query set Q1, and we apply random resize (0.5 ~ 1.5) and random rotation (0 ~ 360) to this 1840 pictures and random choose 400 as query set Q2.

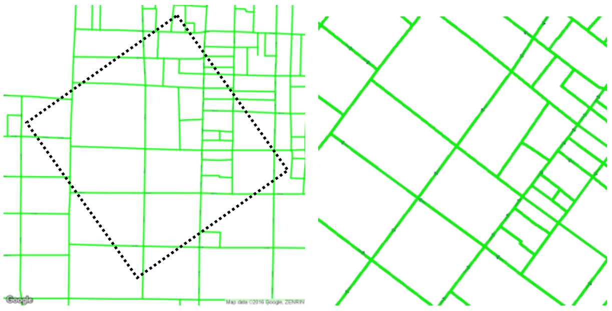

Fig. 9. Random resize and random rotation

Then we classify the query image to serval class according to the cross point they has divided by 5, for example, if one query image has 3 cross point, we put it the 0 group, and if the cross point number is 14, we put it in to the 10 group. In this time, we set threshold to 10, which means if there is one output in the first 10 result output is corresponding with the query image, we consider it's a correct.

Finally we calculate the correctness for each group, try to find the relationship between number of cross point and the correctness.

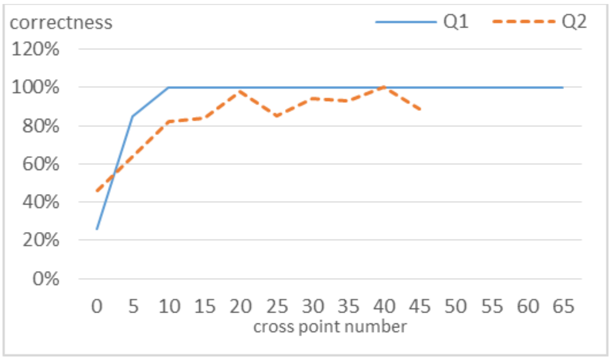

Fig. 10. the relationship between number of cross point and the correctness

In Fig. 10, the horizontal axis stand for interval of the cross point number, the vertical axis stand for the correct rate for the interval. You can see that, they got low correct rate at the beginning, the interval 1~10, this is because the fewer cross point get, the less the information we can get.

Because after resize and rotation, some cross point is missed, so the max value of cross point in Q2 is smaller than Q1.

According to the Fig. 8, we can find that when the number of cross point is bigger than 10, it has high probability (around 85%) to find the correct answer.

Let’s see the average data of Q1 and Q2.

TABLE I. AVERAGE DATA OF Q1 AND Q2

	Q1	Q2
Average Correctness	93.56%	78.95%
Average Cross point number	22.66	16.71
Query set size	466	361

Although the correctness of Q2 is not as good as Q1, but this is mainly because the quantity of Q2 is not as good as Q1, a large amount of the data in Q2 have few cross point.

Even though the rotation and resize operation has bad influence for the result, but this approach still can maintain a high correctness if the query has enough cross point.

We can easily get this result according to the data above: If the query image has enough cross point, then we have a very high probability (85%~90%) to find the correct answer.

In the Q1’s line, we can also find that, if the query image is the same scale as database, this approach performs very well.

D. *Robustness against noise*

In this experiment, we use a larger scale image as query set, the scale of query set is 2 times bigger than database. We should notice that, with 2 times scale of database, then the noise cross point also become more, even though the noise cross point is around the target area. So, in fact, in this section, we would test the robustness of noise cross point, for each query image, we have at least 75% noise information.

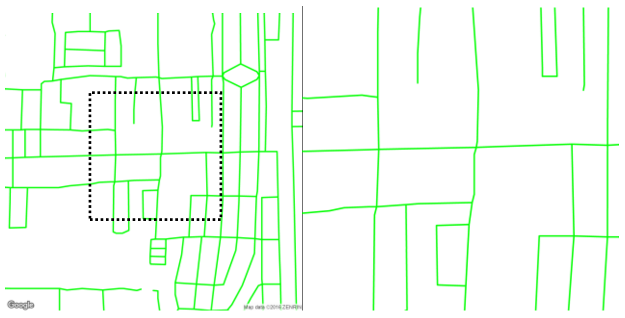

Fig. 11. Compare the query image(left) with the database image(right)

Although the scale of query image and database image is different, but the rotation are the same. In order to test the robustness of the combination of the rotation and noise, we random choose 238 picture as Q3. We also apply random resize and rotation on this larger image, and generate 196 image as Q4, which is the query data in this experiment.

Table 2 shows that the average data of Q3, Q4 with the threshold 10.

TABLE II. AVERAGE DATA OF Q3 AND Q4

	Q3	Q4
Average Correctness	89.50%	93.37%
Average Cross point number	86.31	58.28
Query set size	238	196

We can see that, Q4 have higher correctness then Q3, even though the cross point number is more than Q4. Which mean when the point number is large enough, like 60, it would get good result, but, if the points has large noise information, it would get bad result.

E. *Preselection accuracy and overall query time*

In this part, we use the most complex dataset Q4. Unlike the former experiment, this time, we measure the distribution of correct answer ranking after preselection, we make a statistics on the right answer’s position in the preselection list.

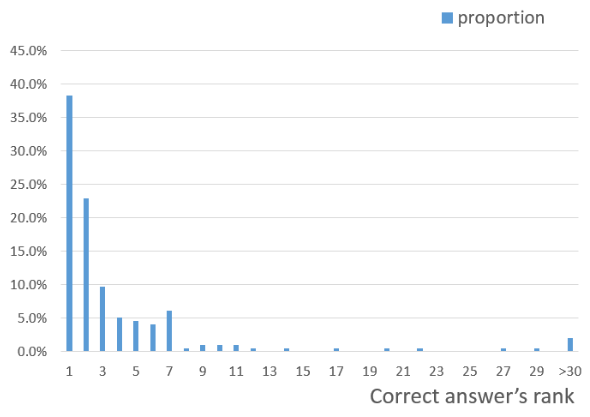

Fig. 12. Distribution of correct answer ranking after preselection

Fig. 13 show the distribution of correct answer ranking after preselection, horizontal axis is the correct answer’s rank in the preselection order, and the vertical axis is the proportion of the rank in the whole query. Then we choose different thresholds to calculate the accuracy in different threshold.

TABLE III. RELATIONSHIP OF THRESHOLD AND ACCURACY

Threshold	**accuracy**
1	38.3%
5	76%
10	93.4%
20	95.9%
30	98%

It’s obviously that with high threshold we got high accuracy. But high accuracy results in high time cost in the final matching step.

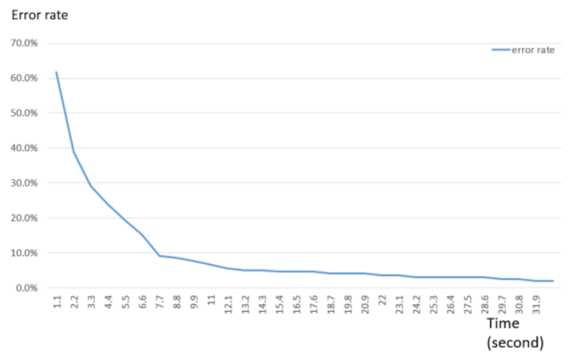

Fig. 13. Relationship between final matching time cost and error rate

As for the total time cost, the CNN part's average time cost is 1.8 seconds, and in this query data, the average cost point number is 58, which means with the average time cost figure, we got 6.5 seconds, so the total preselection time is 8.3 seconds. As for the total time of the whole process, we should plus the preselection time and the final matching time.

TABLE IV. COMPARISON WITH NAÏVE METHOD

		Preselection time	Final match time cost	Total Time cost (seconds)	correctness	Speed up ratio
Naïve method (directly use AKAZE feature to do retrieval)		0	2296	2296	100%	1
Xroad feature	Threshold = 10	8.3	11	19.3	93.4%	112.55
	Threshold = 20	8.3	22	30.3	95.9%	73.12
	Threshold = 30	8.3	31.9	40.2	98%	57.11

Table 4 shows the total time cost with different threshold compared with the most naïve method. We can see that with different threshold, we got different time cost and speed up ratio.

F. *Discussion*

In this part, we show that the results of our approach, and we test rotation and resize robustness, and also show correctness with different threshold.

In the first and second experiments we also find the most important influencing factor for this approach is the number of the cross point in query image. If the number is larger than 15, we can find the right answer with high probability even if the query image has many noise cross point.

In the third experiment, we show the relationship between the correctness and the threshold and the total time cost, which let us know, we can set different for different requirement. We can set different threshold to satisfy our requirement. If you need high correctness, you can set threshold to 30, and got correctness of 98%, with 40 seconds. If you need high speed, you can set the threshold to 10, or lower, it would run very fast and maintain a 90% correctness.

These three experiments show the influence of query images' quantity and the threshold.

VI. CONCLUSION AND FUTURE WORK

We have addressed the task of aerial city image retrial, the difficult of which is the database size, and the similar element and shape of the aerial images. We firstly use CNN to generate the basic road structure of the query image, and analyze the structure, find out the features hidden in the structure, and use a robustness matching approach to find the preselection of target image.

In this paper, we outlined our proposed approach, the basic system's architecture, the robustness theory of cross point relative relationship, and the detail implementation of each step. We also presented the experimental results of robustness and show the how stable the approach has.

As we said at the beginning, our CNN is not good enough to provide high quantity analysis for query image, in fact, I think we need Deep Convolutional Neural Network (DCNN) to do this analysis would better then CNN.

We believe that GPU (Graphics Processing Unit), is particularly well suited for the most step of our approach, like matching step and cross point detect step, they are basically separate computation. We hope to demonstrate accelerated performance results through the use of GPGPU (General-purpose computing on graphics processing units) as well.

REFERENCES

[1] Shekhar R, Jawahar C V. Word Image Retrieval Using Bag of Visual Words[C] Iapr International Workshop on Document Analysis Systems. IEEE Computer Society, 2012:297-301.

[2] Murthy V S V S, Vamsidhar E, Kumar J N V R S, et al. Content Based Image Retrieval using Hierarchical and K-Means Clustering Techniques[J]. International Journal of Engineering Science & Technology, 2010, 2(3).

[3] Niu X M, Jiao Y H. An Overview of Perceptual Hashing[J]. Acta Electronica Sinica, 2008, 36(7):1405-1411.

[4] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," in [Advances in neural information processing systems]

[5] Toshev, A. and Szegedy, C., "Deeppose: Human pose estimation via deep neural networks," in [Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on], 1653–1660 (June 2014).

[6] Farabet, C., Couprie, C., Najman, L., and LeCun, Y., "Learning hierarchical features for scene labeling," Pattern Analysis and Machine Intelligence, IEEE Transactions on 35(8), 1915–1929 (2013).

[7] Haralick R M, Shanmugam K, Dinstein I. Textural Features for Image Classification[J]. Systems Man & Cybernetics IEEE Transactions on, 1973, smc-3(6):610-621.

[8] Mnih, V. and Hinton, G., "Learning to label aerial images from noisy data," in [Proceedings of the 29th Annual International Conference on Machine Learning (ICML 2012)], (June 2012).

[9] Saito S, Aoki Y. Building and road detection from large aerial imagery[C] Image Processing: Machine Vision Applications VIII. 2015:1814–1821.

[10] Lowe, David G. (1999). "Object recognition from local scale-invariant features". Proceedings of the International Conference on Computer Vision. pp. 1150–1157. doi:10.1109/ICCV.1999.790410.

[11] Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. Pablo F. Alcantarilla, Jesús Nuevo and Adrien Bartoli. In British Machine Vision Conference (BMVC), Bristol, UK, September 2013

[12] McCallum, A.; Nigam, K.; and Ungar L.H. (2000) "Efficient Clustering of High Dimensional Data Sets with Application to Reference Matching", Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 169-178 doi:10.1145/347090.347123