FAST 3D HOUGH TRANSFORM COMPUTATION

Egor. I. Ershov, Arseniy P. Terekhin
Simon M. Karpenko, Dmitry P. Nikolaev
Institute for Information
Transmission Problems, RAS
127994, 19, Bolshoy Karetny per.,
Moscow, Russia
E-mail: ershov,ars,simon,dimonstr@iitp.com

Vassili V. Postnikov
JSC Cognitive,
117312, office 709, 9, pr. 60-letiya Oktyabrya,
Moscow, Russia
E-mail: vassili.postnikov@gmail.com

KEYWORDS

Fast Hough transform, Dyadic planes, Exhaustive search, Radon transform, Fast algorithms, Image processing

ABSTRACT

We present a three-dimensional generalization of linear Hough transform allowing fast calculating of sums along all planes in discretized space. The main idea of this method is multiple calculation of two-dimensional fast Hough transforms combined with a specific method for plane parametrization. Compared to the direct summation, the method achieves significant acceleration $\big(O(n^3 \log n)$ vs $O(n^5)\big)$.

INTRODUCTION

Hough Transform (HT) was invented by Paul Hough in 1959 for the analysis of bubble chamber photographs, and patented in 1961. Later HT was modified by R. O. Duda and P. E. Hart to eliminate cases with unbounded transformation space (Hart 2009). There is a widespread opinion that despite algorithm advantages and applicability to different problems Hough transform is too slow with computational complexity being $O(n^3)$.

Fortunately, there is a fast modification of Hough transform - fast Hough transform (FHT), that is not so widely known. Complexity boundary for FHT is $O(n^2 \log n)$ for square image with linear size n, similarly to 2D fast Fourier transform. Moreover, FHT doesn't involve complex arithmetic or even multiplications and could be computed in integer domain.

FHT has a rich reinvention history - we've found four invention precedents. The first one was made in 1995 by W. A. Gotz and H. J. Druckmller (Gotz and Druckmller 1995). Further, Martin Brady reinvented FHT in 1998 (Brady 1998), and several years later in 2004 the version with in-place calculations was proposed (Karpenko et al. 2004), and finally, the last reinvention was made by M. Frederick, N. VanderHorn and A. Somani in 2005 (Frederick et al. 2005). Still, newly published HT applicability surveys do not mention FHT, e.g. (Mukhopadhyay and Chaudhuri 2015), (Hassanein et al. 2015).

FHT has become a very popular tool in image processing; a lot of applications of FHT exist, for instance: edge detection, document orientation, vanishing point detection (Nikolaev et al. 2008), detection of circles and ellipses, linear separation of two-dimensional sets (Ershov et al. 2015a). Also HT was successfully used for robust regression analysis (Ballester 1994; Goldenshluger and Zeevi 2004; Bezmaternykh et al. 2012). We believe, that this tool could improve various computer vision algorithms, e.g. visual odometry and visual localization based on feature point analysis (see Konovalenko et al. (2015); Karpenko et al. (2015)): it allows to use feature lines as well.

Despite of existence and multiple reinventions of FHT for two-dimensional image there is no analogous algorithm for three-dimensional arrays. Such an algorithm could be a very useful tool for many image processing tasks, such as color segmentation, object detection, and orientation estimation using lidar or sonar data, ultrasonic diagnostic, and so on.

In this paper we propose fast three-dimensional Hough transform (3FHT), which calculates sums over all quasi-planes in space using $O(n^3 \log n)$ operations. Here n is linear size of the data cube. We should emphasize that there are two different ways to define 3FHT: as sum along all planes of a cube (discrete Radon transform), or as sum along all lines (discrete Jon transform). From now on we will discuss only discrete analog of Radon transform in three-dimensional space. We use the terms "quasi-plane" or "dyadic plane" to designate discrete planes used in proposed algorithm in contrast to conventional discrete Bresenham planes.

The paper is separated into two chapters. In the first chapter we discuss features of the fast Hough transform for two-dimensional case (2FHT). In the second chapter we describe new 3FHT algorithm, discuss its computational complexity, and geometrical deviation of dyadic plane from its continuous counterpart.

2D FAST HOUGH TRANSFORM

This section is based on materials from (Karpenko et al. 2004) and aimed to emphasize main ideas and features of 2FHT.

Parametrization

Parametrization is one of the main issues while designing Hough transform. A simple form $ax + by + c = 0$ leads to infinite size of Hough space (Hart and Duda 1972). To overcome this problem P. Hart proposed polar parametrization, but unfortunately it doesn't allow to

Proceedings 30th European Conference on Modelling and Simulation ©ECMS Thorsten Claus, Frank Herrmann, Michael Manitz, Oliver Rose (Editors)
ISBN: 978-0-9932440-2-5 / ISBN: 978-0-9932440-3-2 (CD)

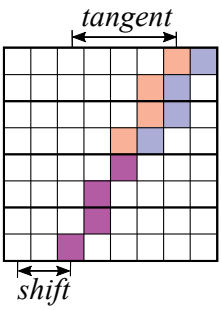

Figure 1: Parametrization and structure of dyadic pattern in two-dimensional fast Hough transform. Two lines with tangent 4 and 5.

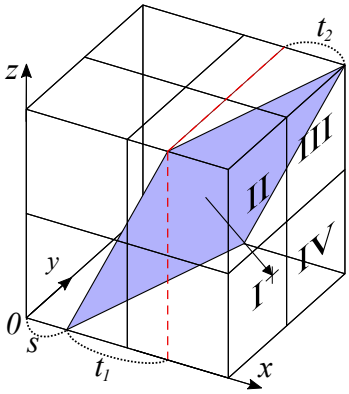

Figure 2: Example of dyadic plane and its parametrization.

construct fast computational scheme, as far as we know. Another parametrization method is shown on figure 1. A line is defined by two positive numbers: shift along one of the rectangle edges s and slope of the line t. Thus all possible lines are divided into four groups: mostly-vertical with right tangent like one on the figure 1, mostly-vertical with left tangent, mostly-horizontal with right tangent, and mostly-horizontal with left tangent. It is easy to see that each group can be transformed to another with reflection or 90° rotation. Thus the computational scheme can be described once. And so the algorithm is described for mostly-vertical lines with right tangent (see fig. 1).

Dyadic Pattern: construction and accuracy

In practice, two Brezenham lines with tangent t and $t+1$ share a lot of pixels or even line segments. Naive HT scheme calculates the same line segment sums multiple times, which leads to computational inefficiency. To overcome this problem Dyadic pattern (structure of discrete line) was proposed. For simplicity we will consider images with linear size $n = 2^p$, where $p \in \mathbb{N}$. To plot dyadic pattern with given tangent D_t one should conduct recursive procedure: at each step the image is divided in half and then initial line segment with slope t is approximated in both halves with shorter line segments having slope $\lfloor t/2 \rfloor$. One pixel shift between these subsegments is added if t is odd. Examples of dyadic patterns are illustrated on the figure 1. Note, that there is no dependence between structure of pattern and $shift$. Mnemonically this rule can be written as

$$D_t = D_{t/2}\,[t \bmod 2]\,D_{t/2} \tag{1}$$

We call such type of discrete patterns "dyadic lines" or "dyadic pattern". This construction is recursive in nature. Computation result's reuse allows for complexity reduction from $O(n^3)$ to $O(n^2 \log n)$.

In (Ershov et al. 2015b) it was shown that maximal possible dyadic line deviation from its geometrical counterpart grows with image size as $\frac{1}{6}\log_2 n$. Thus for an image size 1024×1024 the maximal deviation would be less than two pixels, which is good enough for all practical purposes.

3D FAST HOUGH TRANSFORM

In this section we describe new computational scheme for calculating sums along all dyadic planes in data cube. At first glance, it appears to be a huge computational problem. Indeed, the variety of all planes in $\mathbb{R}^n$ is three-dimensional – therefore, one can uniquely represent almost any plane using three parameters a, b and c as in the following equation:

$$\frac{x}{a} + \frac{y}{b} + \frac{z}{c} = 1 \tag{2}$$

Let us consider planes which intersect given data cube in space. Suppose its edge is of the size n. Then parameters a, b and c would span in $[-2n, 2n]$, so in discrete setting one obtains $12n^3$ planes. It means that the output size of the volumetric Hough transform should be about the same order of magnitude as its input size, which is acceptable. However, for each output cell one seemingly has to calculate the sum over corresponding plane independently. That would take $O(n^2)$ computations per cell, resulting in a huge $O(n^5)$ overall complexity.

Parametrization and Algorithm

Luckily, a considerable speedup is possible. Allowing for modest geometrical inaccuracy, one can compute three dimensional fast Hough transform in $O(n^3 \log n)$, thus spending only $\log n$ summation per output voxel. As far as we know, this result is new. The construction strongly relies on two-dimensional Hough transform described in previous chapter.

Firstly, let us describe convenient plane parametrization. All planes can be divided into twelve groups by normal vector orientation. Indeed, cube has three mutually orthogonal faces, each divided into four parts (see fig. 2). Moreover, normal vector position uniquely defines plane in space. Note that for given normal position it is possible to determine three plane traces. Any pair of which also uniquely determines plane parameters. Therefore, to parametrize the plane, it is enough to fix some point on edge with coordinate $(s, 0, 0)$ and pair of line slopes t_1, t_2. For simplicity, we will consider further type `I` planes. It has two mostly-vertical with right slope traces in xy-face (t_1) and in xz-face (t_2). Each trace is right-sloped as illustrated in fig. 2.

Let us sketch out the idea of 3FHT. You can find working MATLAB implementation on github: *https://github.com/Ershoff/FastHoughTransform3D*.

Double integral over any type I plane that intersects some fixed face F of the cube can be represented as itered integral. First, one have to integrate over all horizontal lines intersecting F. Second, these line integrals should be integrated again over a mostly-vertical lines contained in F. Let's consider this idea in detail.

At the first stage we apply 2FHT to each horizontal xy-slice of the data cube. This way we will obtain another cube (sliced-FHT cube) where voxels represent sum along corresponding lines in horizontal slice. At the second stage we apply 2FHT for each vertical xz-slice of this sliced-FHT cube. Resulting HT-cube contains sum along corresponding plane at each voxel.

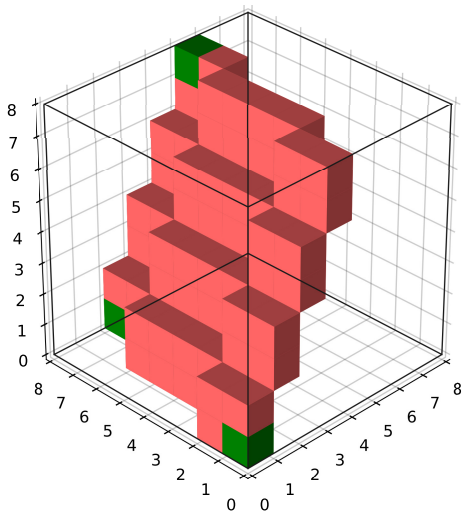

Figure 3: Example of a dyadic plane. This one has $s = 0$, $t_1 = 2, t_2 = 4$.

Pseudo-code of FHT3 is described in Algorithm 1. For simplicity, we use function $fht2\_quart()$ that performs 2FHT calculation only for the mostly-horizontal lines with right slope. Thus, $fht2\_quart()$ takes 2D square array as an input, and returns an array of the same size. To simplify pseudo-code we use ":" notation to work with array dimensions. Operation $dc(:,:,i)$ return a reference to the two-dimensional subarray of dc with $z = i$.

Three coordinates of these voxels correspond to shift s along x-axis and slopes t_1, t_2. Example of dyadic plane is shown in figure 3.

Complexity and Precision

Let us consider complexity of proposed algorithm. On the first stage we apply 2FHT to each of n horizontal xy-slices of the data cube. On the second stage we apply 2FHT to each of n vertical xz-slices of the sliced-FHT cube. So we perform two-dimensional fast Hough transform n-times sequentially for both slice types. As 2FHT requires $O(n^2 \log n)$ operations, both stages have $O(n^3 \log n)$ computational complexity. Difference between execution time of naive HT algorithm and 3FHT is illustrated on figure 4.

Algorithm 1 Pseudo-code of 3D Fast Hough Transform

```
function FHT3(dc)                          ▷ dc - data cube
    n ← size_o f_e dge(dc);                ▷ n - cube edge size
    hs ← zeros(n, n, n);                   ▷ cube filled with zeros.
    dhs ← hs;
    for i = 1 : n do
        hs(:, :, i) ← fht2_quart(m(:, :, i));
    end for
    for i = 1 : n do
        dhs(:, i, :) ← fht2_quart(hs(:, i, :));
    end for
    return dhs;
end function
```

In previous researches Ershov et al. (2015b), we succeed in proving that maximal spacial distance between corresponding dyadic pattern and ideal line has order $\frac{1}{6}\log_2 n$. Moreover we showed that the largest deviation is achieved in $t = n/3$. It is easy to see that maximal deviation in 3FHT should be twice as big as in 2FHT, i.e. $\frac{1}{3}\log_2 n$.

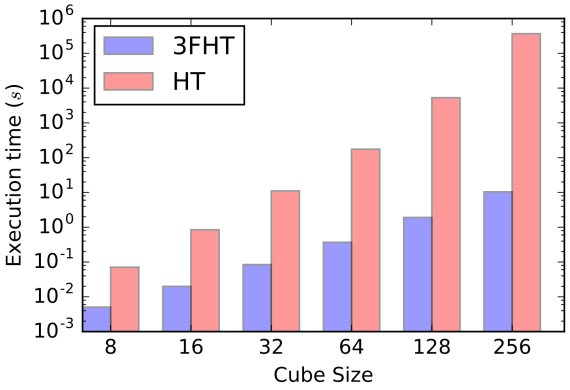

Figure 4: Comparison of computational time for naive HT algorithm and three-dimensional fast Hough transform algorithm. Execution time is given in log scale.

CONCLUSION

In this paper a new effective scheme for calculating three-dimensional Hough transform is presented. Computational complexity of proposed algorithm is reduced from $O(n^5)$ to $O(n^3 \log n)$. To achieve this result we propose novel three-dimensional dyadic pattern and plane parametrization. We state that the maximal deviation of dyadic plane from its geometrical counterpart is equal to $\frac{1}{3}\log_2 n$, where n is linear size of the data cube.

ACKNOWLEDGEMENTS

Applied scientic research is supported by Ministry of Education and Science of the Russian Federation (projects RFMEFI58214X0002)

REFERENCES

Ballester, P. 1994. "Hough transform for robust regression and automated detection." *Astronomy and Astrophysics*, 286:1011–1018.

Bezmaternykh, P.V.; T.M. Khanipov; and D.P. Nikolaev. 2012. "Linear regression task solution with fast Hough transform (in Russian)." *35th Conference and School on Information Technologies and Systems (ITaS 2012)*, 354–359.

Brady, M. 1998. "A fast discrete approximation algorithm for the Radon transform." *SIAM J. Computing*, 27(1):107–119.

Ershov, E.I.; D.P. Nikolaev; V.V. Postnikov; and A.P. Terekhin. 2015a. "Exact fast algorithm for optimal linear separation of 2d distributions." *Proceedings 29th European Conference on Modelling and Simulation (ECMS 2015)*, 469–474.

Ershov, E.I.; A.P. Terekhin; D.P. Nikolaev; V.V. Postnikov; and S.M. Karpenko, "Fast Hough transform analysis: pattern deviation from line segment." In *Eighth International Conference on Machine Vision*, 987509I 1–5 (International Society for Optics and Photonics, 2015b).

Frederick, M.; N. VanderHorn; and A. Somani. 2005. "Real-time H/W implementation of the approximate discrete radon transform." *IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP'05)*, 2:399–404.

Goldenshluger, A. and A. Zeevi. 2004. "The hough transform estimator." *The Annals of Statistics*, 32:19081932.

Gotz, W.A. and H.J. Druckmller. 1995. "A fast digital Radon transform – An efficient means for evaluating the Hough transform." *Pattern Recognition*, 28.12:1985–1992.

Hart, P. 2009. "How the Hough transform was invented [DSP history]." *Signal Processing Magazine IEEE*, 26(6):18–22.

Hart, P. and R. Duda. 1972. "Use of the Hough transformation to detect lines and curves in pictures." *Communications of the ACM*, 15:11–15.

Hassanein, A.S.; S. Mohammad; M. Sameer; and M.E. Ragab. 2015. "A survey on Hough transform, theory, techniques and applications." *arXiv preprint arXiv:1502.02160*.

Karpenko, S.M.; I.A. Konovalenko; A.B. Miller; B.M. Miller; and D.P. Nikolaev. 2015. "UAV Control on the Basis of 3D Landmark Bearing-Only Observations." *Sensors*, 15(12):29802–29820.

Karpenko, S.M.; D.P. Nikolaev; P.P. Nikolayev; and V.V. Postnikov. 2004. "Fast Hough transform with controllable robustness (in Russian)." *In Proc. of IEEE AIS'04 and CAD-2004*, 2:303–309.

Konovalenko, I.A.; A.B. Miller; B.M. Miller; and D.P. Nikolaev, "UAV navigation on the basis of the feature points detection on underlying surface." In *Proceedings of the 29th European Conference on Modeling and Simulation (ECMS 2015), Albena (Varna), Bulgaria*, 499–505 (2015).

Mukhopadhyay, P. and B.B. Chaudhuri. 2015. "A survey of Hough transform." *Pattern Recognition*, 48(3):993–1010.

Nikolaev, D.P.; S.M. Karpenko; I.P. Nikolaev; and P.P. Nikolayev. 2008. "Hough transform: underestimated tool in the computer vision field." *Proceedings of the 22th European Conference on Modelling and Simulation*, 238–246.

AUTHOR BIOGRAPHIES

EGOR ERSHOV was born in Moscow, USSR. He studied engineer science and mathematics, obtained his Master degree in 2014 from Moscow Institute of Physics and Technology. Now he is a Ph.D. student. Since 2014 he is working in Vision Systems Lab at the Institute for Information Transmission Problems RAS. His research activities are in the areas of computer vision. His e-mail address is ershov@iitp.ru.

VASILII POSTNIKOV was born in Sverdlovsk, USSR. He studied applied mathematics, obtained his Master degree in 1990 and Ph.D. degree in 2001 from Moscow Institute of Physics and Technology. Since 1993 he works at Institute for System Analysis, RAS. His research activities are in the area in the area of image analysis and video data recognition. His e-mail address is vasilli.postnikov@gmail.com.

SIMON KARPENKO was born in Moscow, USSR. He studied mathematics, obtained his Master degree in 2002 from Moscow State University. Since 2007 he works in the Vision Systems Lab. at the Institute for Information Transmission Problems RAS. His research activities are in the areas of computer vision, machine learning and pattern recognition. His e-mail address is simon@iitp.com.

ARSENIY TEREKHIN was born in Moscow, USSR. Since 2005 he worked as a programmer for a brokerage company. In 2014 he joined Vision Systems Lab. at the Institute for Information Transmission Problems, RAS. His e-mail address is ars@iitp.ru

DMITRY NIKOLAEV was born in Moscow, USSR. He studied physics and computer science, obtained his Master degree in 2000 and Ph.D. degree in 2004 from Moscow State University. Since 2007 he is a head of the Vision Systems Lab. at the Institute for Information Transmission Problems RAS. His research activities are in the areas of computer vision and image processing with focus on computation effective algorithms. His e-mail address is dimonstr@iitp.ru.