JFP **21** (2): 129–133, 2011. © Cambridge University Press 2011
doi:10.1017/S0956796810000341 First published online 24 February 2011

FUNCTIONAL PEARL
The Hough transform

MAARTEN FOKKINGA
Department EEMCS, University of Twente, Enschede, The Netherlands
(*e-mail:* m.m.fokkinga@utwente.nl)

1 Introduction

Suppose you are given a number of points in a plane and want to have those lines that each contain a large number of the given points. The Hough transform is a computerized procedure for that task. It was invented by Paul Hough (1962), originally to find the trajectories of subatomic particles in a bubble chamber, and it has even been patented. Nowadays, adaptations of the Hough transform are used, among others, for identification of transformed instances of a predefined figure, instead of just a line, in a digital picture. There are plenty of explanations on the Internet (use search key "Hough transform" and "generalized Hough transform"), some with nice applets to demonstrate the working (add search key "applet" or "demo"). Recently, Hart (2009) has looked back at the invention. We show how the original procedure could have been derived. The derivation has the following notable properties:

- The "transform" is a mapping of the plane to another space in such a way that manipulations in the plane can be done equivalently in the other space, and vice versa. Hart (2009) describes its invention as: *one of those inexplicable yet genuine "aha!" insights: Mapping a zero-dimensional point to a one-dimensional straight line—which by increasing the dimensionality seems to make the problem more complicated—[…]*. This step falls out quite naturally in the course of our derivation.
- We exploit the *addition of functions* ($f \mathbin{\hat{+}} g$ is the function that maps x to $fx + gx$), and in particular the *fold-with-*$\hat{+}$: when applied to a collection of functions $f, g, \ldots$ it yields the function that maps x to $fx + gx + \cdots$.

In order to consider only a finite number of lines, the Hough transform uses a *discretization* of the space. The test for a line containing a point then needs to be relaxed; a line "contains" all points that are sufficiently close to it. Equivalently, the lines can be thought of as having finite thickness. For this to work in practice, the discretization should be fairly fine, but, for reasons of efficiency, not too fine. In addition, in a practical setting, there is *uncertainty* about the given points: the location of the points may be inaccurate, and some intended points may not be given (loss) and some given points may not be intended (noise). Dealing with discretization and uncertainty is beyond the scope of this note, as is further refinement in order

to improve computational efficiency. Our aim is to present the principle underlying the Hough transform in an idealized setting.

2 The derivation

We are given a fixed set of points and, consistently, p will vary over this set. We want to have one or more lines F, each of which contains the largest number of the given points ("F" is mnemonic for Figure, to which line can be easily generalized). Abstracting from executability on a computer, a specification is easy to give:

> Consider all lines F.
> Assign to each line: the number of given points that are contained in F. (1)
> Deliver the lines whose assigned number is maximal.

A succinct formulation (explained below) of this specification reads

$$\arg\max\,(\lambda F.\ \ \#\{p \mid p \in F\}) \tag{2}$$

The first phrase of Equation (1), "consider all lines," is formalized by leaving out the domain of F: thus F ranges over all lines. The assignment of "the number of given points that are contained in F" to line F is expressed as function $\lambda F.\ \#\{p \mid p \in F\}$. Well-known operation $\arg\max$, defined by $\arg\max f = \{x \mid \forall y.\ fx \geqslant fy\}$, chooses those F-values for which $\#\{p \mid p \in F\}$ is maximal.

In order to represent lines in the data types available to a computer, we assume a line to be identified by a collection of numeric parameters. For example, two well-known ways of characterizing a line in the x, y-plane are

$$F(a,b) = \{x, y \mid y = ax + b\}$$
$$F(\rho,\varphi) = \{x, y \mid \rho = x\cos\varphi + y\sin\varphi\}$$

In the latter characterization, ρ is the distance of the line to the origin and φ is the angle between the line and the y-axis. In the sequel, we shall use q to denote a parameter that uniquely identifies a line Fq according to a fixed representation; for example, q is (a,b) or it is (ρ,φ). (Notice that we now write "Fq" for a line parameterized by q, whereas above merely "F" itself denotes a line.) Thus, we rewrite Equation (2), using the parameter identification of lines:

$$\arg\max\,(\lambda q.\ \ \#\{p \mid p \in Fq\}) \tag{3}$$

(To get a real equality between Equations (2) and (3), mapping $q \mapsto Fq$ should be applied to every outcome of the latter in order to get exactly the outcome of the former.) Next, we replace the size operator $\#$ by one of its defining expressions: $\#\{x \mid \ldots x \ldots\}$ is a sum of as many 1's, as there are x-values for which $\ldots x \ldots$ holds. A sum can be expressed as *fold-with-*$+$. For an associative operation $\oplus$ with a neutral element, we write the *fold-with-*$\oplus$ as $\oplus/$—it can be refined to Haskell's *foldl* as well as *foldr*. Furthermore, we abbreviate "1 if $p \in Fq$ else 0" to $\langle p \in Fp\rangle$, and we write $\{\!|p \bullet \ldots p \ldots|\!\}$ for "the bag of all values $\ldots p \ldots$ where p ranges over the given set of points." Thus, line (3) equals

$$\arg\max\,(\lambda q.\ \ +/\{\!|p \bullet \langle p \in Fq\rangle|\!\}) \tag{4}$$

https://doi.org/10.1017/S0956796810000341 Published online by Cambridge University Press

Here, we see a single function $\lambda q.\ +/\{\!|\ldots|\!\}$ whose outcome for each q is a sum of various values. By the very definition of "addition of functions," this can be written as a single sum of various little functions (recall that $\hat{+}/\{\!|f,g,\ldots|\!\} = f \mathbin{\hat{+}} g \mathbin{\hat{+}} \cdots = \lambda x.\ fx + gx + \cdots$):

$$\arg\max\,(\hat{+}/\{\!|p \bullet (\lambda q.\langle p \in Fq\rangle)|\!\}) \tag{5}$$

Now, for arbitrary p, function $(\lambda q.\ \ldots)$ can be rewritten in view of the surrounding addition $\hat{+}/$: we distinguish between the parameters that yield a zero result (the neutral element for $+$) and a nonzero result. The function yields a nonzero result (namely 1) for parameter q precisely when $p \in Fq$; so we define $Gp = \{q \mid p \in Fq\}$ and then

$$\begin{aligned} & \lambda q.\ \langle p \in Fq\rangle \\ = & \text{ above definition of } Gp \\ & (\lambda q : Gp.\ 1) \cup (\lambda q : \text{complement of } Gp.\quad 0) \\ = & \\ & (\lambda q : Gp.\ 1)^\circ \end{aligned}$$

Operation $\_^\circ$ completes a partial function to a total one: $f^\circ x = fx$ if $x \in \mathrm{dom} f$ else 0; again, in view of the surrounding $\hat{+}/$, usage of this operation seems useful. Thus, line (5) equals

$$\arg\max\,(\hat{+}/\ \{\!|p \bullet (\lambda q : Gp.\ \ 1)^\circ|\!\}) \tag{6}$$

This ends the derivation. To see the correspondence with other explanations of the Hough transform, expression (6) can readily be formulated in an imperative fashion. Remember the imperative realization of folds: for numbers a_i, the result of $+/\ \{\!|a_1, a_2, \ldots|\!\}$ can be accumulated in number variable A as follows:

initialize A to 0;
for each i do: increment A by a_i

Similarly, for functions $f_i : Q \to \mathbb{N}$, the result of $\hat{+}/\ \{\!|f_1, f_2, \ldots|\!\}$ can be accumulated in function variable $A : Q \to \mathbb{N}$ (in pseudocode: **array** $A[Q]$ **of** $\mathbb{N}$) as follows:

initialize A at each q to 0;
for each i do: increment A at each q by $f_i q$

Therefore, exploiting also that zeros have no contribution to the final sum, it turns out that Equation (6) can be written as follows, using a so-called *accumulator* A:

initialize A at each q to 0;
for each p do: increment A at each $q \in Gp$ by 1; (7)
deliver the q's for which A at q is maximal.

3 Discussion

3.1 The crux

The crux of the procedure is, perhaps, the equivalence $p \in Fq \iff Gp \ni q$; it is, in fact, the *definition* of the Hough transform. It enables us to do manipulations from

https://doi.org/10.1017/S0956796810000341 Published online by Cambridge University Press

the point space equivalently in the parameter space. For example, "multiple points $p, \ldots, p'$ have a common Fq" equals "multiple figures $Gp, \ldots, Gp'$ have a common q" (in which case accumulator A is incremented multiple times at q). The equivalence is stressed (without mentioning an explicit formula!) in all explanations of the Hough transform I know, but it does not play a prominent role in our derivation—it is used, implicitly, in the step from Equations (5) to (6).

In contrast, in the formal derivation, the major structural change in the expressions occurs in the step from Equations (4) to (5), although the step is just an application of the definition of *fold*. This step reverses the nesting of scopes: in Equation (4), the scope of p is properly part of the scope for q, whereas in Equation (5), it is the other way around. This translates to the imperative formulations with nested iterations: in the initial specification (1), there is an outer loop for q, whereas in the final formulation (7), there is an outer loop for p.

3.2 Arg max

Operation argmax does not enter into the calculation: it is carried along at every step. Indeed, it can be replaced by anything else. A particularly useful choice is "arg top_k," thus selecting the lines F whose assigned number is among the top-k largest. In this way, those lines are found that each contain "a large" number of the given points. Notice, however, that in a practical setting where the location of the points may be inaccurate, parameters q for which accumulator A is *locally* maximal are more useful than the q for which $A[q]$ belongs to the top-k values.

3.3 Lines and figures

For the first example representation of straight lines in the x, y-plane, figure Gp in the parameter plane turns out to be a straight line as well, whereas for the second representation, figure Gp is a sine curve. The second representation has the advantage that each straight line can be represented by a finite value for ρ, φ. In both cases, the "line form" of figure Gp makes it easy to enumerate the $q \in Gp$, a subtask that occurs in Equation (7).

Nowhere in the derivation did we use that F is a straight line or that points p come from a plane. Far-reaching generalizations are possible. For example, figure F may be a predefined fixed figure, and each Fq may be a transformed (translated, scaled, rotated) instance of F, characterized by parameter q. Needless to say that in such a case, q will be a conglomerate of several values (not just the pair x, y or ρ, φ) so that enumerating the $q \in Gp$ increases the computational complexity considerably.

Acknowledgments

A previous version of this paper was presented at a symposium on January 22, 2010, on the occasion of the retirement of Lambert Meertens as professor at the Utrecht University. I acknowledge the elaborated and useful comments of the reviewers; they have led to a considerable improvement of the presentation and content.

References

Hart, P. E. (2009). How the Hough transform was invented [DSP History], *Signal Process. Mag. IEEE*, 26 (6): 18–22.

Hough, P. V. C. (December 8, 2010). Method and means for recognizing complex patterns [online]. US Patent 3,069,654. Available at: `http://www.freepatentsonline.com/3069654.pdf`. Interesting excerpts appear in (Hart, 2009).

https://doi.org/10.1017/S0956796810000341 Published online by Cambridge University Press