Annales Univ. Sci. Budapest., Sect. Comp. 56 (2024) 167–184

REAL-TIME WEB-BASED VISUALIZATION OF THE RADON TRANSFORM

Zsófia Gál and Levente Lócsi

(Budapest, Hungary)

Communicated by László Szili

(Received April 29, 2024; accepted July 10, 2024)

Abstract. The Radon transform is the mathematical tool behind nowadays widely used medical imaging techniques for diagnostic purposes such as Computer Tomography (CT). In this paper we discuss two of our web-based approaches which allow one to interactively create sample images and examine their Radon transform in order to study simple examples and basic properties of the transform. We present that the transform (more precisely its support) can be calculated in an analytic way such that it allows (near) real-time visualization, orders of magnitude faster than in earlier accessible solutions. We claim that currently this may be the best approach for introductory educational and science communication purposes about the Radon transform.

1. Introduction

The Radon transform is the mathematical tool behind nowadays widely used medical imaging techniques for diagnostic purposes such as Computer Tomography (CT). It was introduced by Johann Radon in 1917 [6], more than a hundred years ago [7]. The basic idea of the transform is to calculate the integral of a given real function defined on the plane along lines of the plane,

Key words and phrases: Radon transform, real-time visualization, JavaScript.
2010 Mathematics Subject Classification: 44A12, 92C55, 97U60.
EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies – The Project is supported by the Hungarian Government and co-financed by the European Social Fund.

corresponding to the adsorption of X-rays travelling through segments of the human body from different directions. In practical applications the inverse Radon transform shall be used to reconstruct the original function (or medical image) from the measured line integral values. Radon already showed that this is mathematically possible, and it is closely related to the Fourier transform. It is worth mentioning that later the theory was elaborated so that the first CT machine could be engineered, and the Nobel Prize in Physiology or Medicine 1979 was awarded jointly to Allan M. Cormack and Godfrey N. Hounsfield for the development of computer assisted tomography.

If someone wanted to see examples for the Radon transform on their computer screens, to study it in case of some simple images to get familiar with its basic properties, one needed to write at least some short programs in Matlab (with the Image Processing Toolbox), Octave or Mathematica etc. to utilize their built-in commands for the transform and for visualization. (And/or do the calculations using pen and paper.) This is slow and cumbersome for most introductory purposes.

Our project to be presented aims for a web-based solution, where one could easily create a simple image consisting of some basic objects (rotated ellipses and rectangles, such as in the case of the well-known Shepp–Logan phantom [8] or the recently introduced Bognár lung phantom [1]) and modifications are immediately reflected on the transform. This kind of visualization already existed e.g. in case of the two-dimensional Fourier-transform (see the 'Fourifier' under https://ejectamenta.com/imaging-experiments/fourifier/ by Watts, D.), and is in-line with trends also in educational tools emphasizing quick feedback and interaction with students [2]. We see that with some trade-off concerning the details of the transform shown, suitable accuracy and (at least near) real-time interaction is achievable, orders of magnitude faster than using earlier solutions.

In this paper we summarize the basic mathematics of the Radon transform, discuss and express criticism to earlier methods in order to find motivation for our project, then we describe two of our recent approaches: a server-side, numerical, and a client-side, analytic one, with some details about the calculations. Finally, the conclusions are drawn, and further research directions are pointed out.

2. The Radon transform

In this section we summarize the basic mathematics of the Radon transform, we recall the definition, introduce the used notation, formulate the relation to the Fourier transform, and shortly mention the problem of the inverse transform.

2.1. Definition

For the mathematical definition of the transform let a line L on the $\mathbb{R}^2$ plane be represented by parameters φ and r, where $\varphi \in [0, \pi)$ is the angle of the normal of the line and $r \in \mathbb{R}$ is the distance between the line and the origin. Let S denote the class of functions with a compact support: $S := \{f \in C^\infty(\mathbb{R}^2)$: : supp f is compact$\}$. Then we can define the Radon-transform of a function $f \in S$ as the function $\mathcal{R}f(L) := \mathcal{R}f(\varphi, r) := \int_L f(x, y)\,\mathrm{d}s$, where $\int_L \cdot\,\mathrm{d}s$ denotes the integral along the line L.

Using the equation of the line with parameters φ and r (that is: $x\cos\varphi + +y\sin\varphi = r$) the Radon transform of a function f can be written in the following form:

$$\mathcal{R}f(\varphi, r) = \int\limits_{\mathbb{R}} \int\limits_{\mathbb{R}} f(x, y)\,\delta(x\cos\varphi + y\sin\varphi - r)\,\mathrm{d}x\mathrm{d}y,$$

where δ denotes the Dirac delta function. We may get another useful form of this integral using the parameterization of L as follows:

$$\mathcal{R}f(\varphi, r) = \int\limits_{\mathbb{R}} f(r\cdot\cos\varphi - s\cdot\sin\varphi, r\cdot\sin\varphi + s\cdot\cos\varphi)\,\mathrm{d}s$$

If we fix the angle φ and compute the integral along each line with normal angle φ, then we obtain a *projection* of f in direction φ, denoted by $p_\varphi(t)$, i.e.:

$$p_\varphi(t) := \mathcal{R}f(\varphi, t) = \int\limits_{\mathbb{R}} \int\limits_{\mathbb{R}} f(x, y)\,\delta(x\cos\varphi + y\sin\varphi - t)\,\mathrm{d}x\mathrm{d}y.$$

Figure 1 depicts how a projection for a fixed angle is calculated.

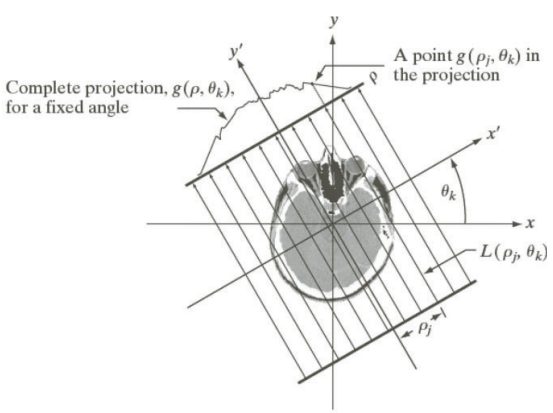

Figure 1. Projection to the line with angle θ_k. Source: [5].

In the discrete version of the transform we can think of f as an image of size $M \times N$ pixels, and its transform has a form similar to the continuous case. However, instead of considering the Dirac delta function interpolation and/or Fourier techniques are usually applied in practice, pixel extents, subpixels, accumulation matrices could be utilized.

To visualize the transform as an image we take the normal angles along the horizontal axis, and the distance from the origin along the vertical axis. This visualization of the Radon transform is also called *sinogram*, referring to the fact that the transform of a single point has the form of a sine wave. As an example Figure 2 presents the Shepp–Logan (brain) phantom [8] and its sinogram.

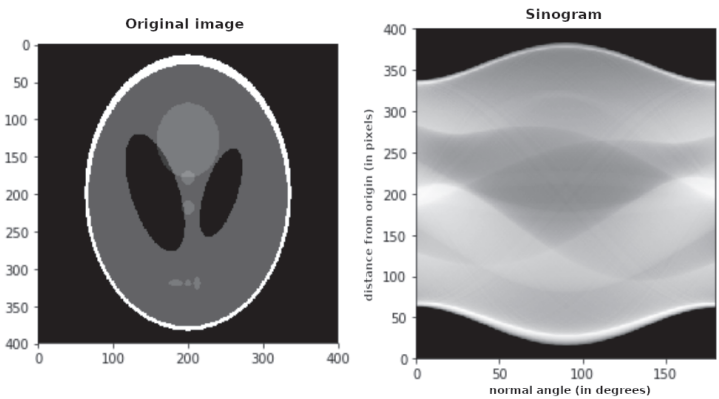

Figure 2. Sinogram of the Shepp–Logan phantom.

2.2. Relationship with the Fourier transform

An important property of the Radon transform is its connection with the Fourier transform. It can be shown, that the one-dimensional Fourier transform of a projection with angle φ is the same as the values along the line with angle φ of the two-dimensional Fourier transform:

$$(\mathcal{F}p_\varphi)(\omega) = \int_{\mathbb{R}} p_\varphi(t)\, e^{-2\pi i\omega t}\, \mathrm{d}t =$$

$$= \int_{\mathbb{R}}\int_{\mathbb{R}}\int_{\mathbb{R}} f(x,y)\, \delta(x\cos\varphi + y\sin\varphi - t)\, e^{-2\pi i\omega t}\, \mathrm{d}x\mathrm{d}y\mathrm{d}t =$$

$$= \int_{\mathbb{R}} \int_{\mathbb{R}} f(x,y) \int_{\mathbb{R}} \delta(x\cos\varphi + y\sin\varphi - t)\, e^{-2\pi i\omega t}\, \mathrm{d}t\mathrm{d}x\mathrm{d}y =$$

$$= \int_{\mathbb{R}} \int_{\mathbb{R}} f(x,y)\, e^{-2\pi i\omega(x\cos\varphi + y\sin\varphi)}\, \mathrm{d}x\mathrm{d}y =$$

$$= \int_{\mathbb{R}} \int_{\mathbb{R}} f(x,y)\, e^{-2\pi i(\omega\cos\varphi\cdot x + \omega\sin\varphi\cdot y)}\, \mathrm{d}x\mathrm{d}y.$$

This property has a role in the inverse problem.

2.3. Inverse of the transform

In most real-life problems we want to reconstruct the original function or image knowing only an approximation of its transform which is determined by measurements. Several methods exist to obtain an approximate solution to the problem.

In *backprojection* first we take one projection, and project it along the image. We do the same with all projections, and add the results to create one image, which we scale to the interval $[0, 1]$. The blurring effect can be avoided if we apply a high-pass filter on the projection before backprojecting. This method is called *filtered backprojection*.

It is also possible to use iterative methods to approximate the solution. Here an initial guess is taken, and it is improved iteratively by computing its transform and comparing it with the original transform. The books [3, 5, 9] provide further details about solving the inverse problem.

3. Visualization methods

This section provides an overview of the visualization methods about the Radon transform available in popular mathematical software. Since these are not to our satisfaction with respect to our goals, we find our motivation, and present two of our solutions.

3.1. In mathematical software

Well-known and widely used mathematical software usually provide an implementation of the Radon transform (and its inverse), and provide appropriate visualization tools. Below we'll examine Matlab, Octave, Octave-Online, Mathematica, Wolfram Alpha, with remarks on Python, C++ and Java.

In MathWorks' **Matlab** there are the commands `radon` and `iradon`. These commands are only available with the *Image Processing Toolbox* also installed.

Note that Matlab is a commercial software, with research and academic licenses available, thus it is not reasonable for everyone to have access to Matlab. Nevertheless its `radon` command works very fast, computes the transforms of a small—e.g. 400 by 400 pixel image—within 1/20 seconds, thus it may be suitable even for real-time visualization purposes too. However the creation of a basic image to transform would require one to write at least some code (even when importing an image). E.g. the lines below create an empty image (zero intensity), add a full intensity square, compute its Radon transform and display the image and the transform aside each other.

```
% Radon transform of a square

image = zeros(400,400);
image(150:250,150:250) = 1;
tr = radon(image);

% Create plots

subplot(1,2,1)
imagesc(image)
colorbar
axis square

subplot(1,2,2)
imagesc(tr)
colorbar
axis tight
```

For just a simple image one may just use `imagesc(tr)` after the transform. Creating a circle is a bit more involved, we provide two equivalent ways (in terms of the results) to do so. The first way uses loops (as in an imperative approach), and the second one uses matrix manipulation and indexing (array programming fashion).

```
% An image with a circle - 1st way

image = zeros(400,400);
for i = 1:400
    for j = 1:400
        if (i-100)^2 + (j-200)^2 < 70^2
            image(i,j) = 1;
        end
    end
end
```

```
% An image with a circle - 2nd way

image = zeros(400,400);
[i,j] = meshgrid(1:400);
image((i-100).^2 + (j-200).^2 < 70^2) = 1;
```

Figure 3 shows the above circle and its Radon transform as calculated by the Matlab system.

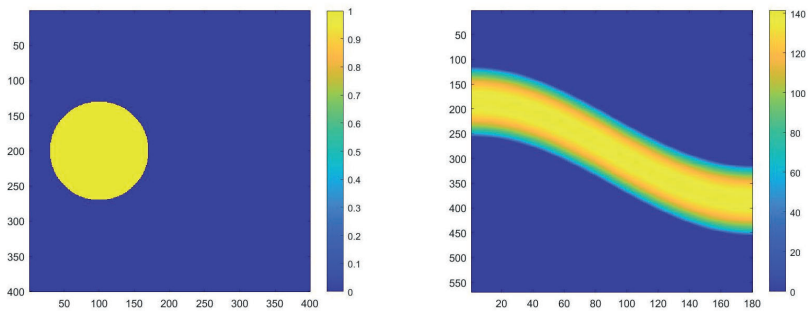

Figure 3. A circle and its Radon transform via Matlab.

If one wants to position the square(s) or circle(s) differently, or make them rectangle(s) or ellipse(s), the commands must be issued again, or rather the script modified, and run again. This makes editing the 'scene' cumbersome. Of course one may also write graphical user interface programs in Matlab to create some simple images as above, utilizing the fast Radon transform calculations, but then again this may not be published for everyone to try. Nevertheless, we find that for involved research purposes related to the Radon transform a tool like Matlab is inevitable.

The GNU **Octave** is an open-source clone of Matlab, i.e. it is freely available to everyone unlike Matlab. The above commands and programs work the same in Octave too, after loading the *Image* package with `pkg load image`. Yet another difference is the speed of calculation. The Radon transform implementation in Octave is significantly slower than that of Matlab: the calculation of the same—400 by 400 pixel—image takes about 5 seconds to complete, that is about 100 times slower. This makes this implementation unsuitable for real-time visualization, and the editing isn't any more simple either.

The online version of Octave, available at https://octave-online.net/ deserves credit too. The above commands also work here (with minor limitations). However to finish the calculation of the transform of an image of

the same size as above, one must even 'request extra time' by clicking on the appearing *Add 15 seconds* link, so it is very slow. But it has the significant advantage that no installation of any software or additional package is required, it works simply within a browser, i.e. this approach is far more easily accessible than the above mentioned software.

Wolfram's **Mathematica** also provides access to the Radon transform of images, the commands `Radon`, `InverseRadon` and also the symbolic variant `RadonTransform` are available. One may also program parameterized visualizations, the calculation speed is comparable to Matlab's (fast), but this is also a commercial software.

The online version of Mathematica's tools also allows some basic usage of the Radon transform. It is available at `https://www.wolframalpha.com/`, e.g. try `radon(square)`. More involved online usage is limited.

There is also a non-commercially available Radon transform implementation provided by the *scikit* package, written in **Python** language. The speed of this is comparable to Octave's (slow). One may find also **C++** and **Java** implementations, however these are not suitable for visualization purposes without any further implementation. There were also Java Applets implemented earlier dealing with the Radon transform, such as the ones under `https://www.rabidhamster.org/java/JavaRadon.php` (by Williams, G., 2003) or `http://rekrylov.chat.ru/tomography/`, but browsers' support for such software is increasingly dropping, so that today to try, to judge the capabilities and assess calculation speed of these is not realistic, accessibility is very limited.

Taken into account the capabilities and limitations of the above software, we conclude to summarize our goals as:

- Provide a visualization method for the Radon transform, which allows
- real-time, interactive scene editing and transform (also real-time),
- and is publicly available, easily accessible to everyone.

We see that this is not possible with the tools examined so far. Finally we arrive to the conclusion that a web-based approach is desirable, since it allows easy access for all. (We consider having an up-to-date browser on a reasonable hardware provided as a minimum.)

We argue that visualization is important also in education and science communication. After all, the Radon transform is related to Nobel prize level thoughts which are worth spreading to an audience broad as possible. Furthermore, visualization is useful to understand some of the basic properties of the transform deeper, to have some more insight compared to pure theoretical examination. The aesthetic value is also worth to consider.

3.2. Our first web-based approach: server-side, numerical

In our first approach the Radon transform of the image is computed by a server software written in Python. It uses the *scikit* library's implementation of the transform.

Using the graphical interface of the application the user can create a gray-scale image in an interactive way in a browser, by adding and rotating rectangles and ellipses with different sizes and intensity values. This facility is created using the *Fabric.js* library (http://fabricjs.com/). At a click of a button the image is sent to the server, which computes the Radon transform, and then sends the transform (again as an image) back to the client. After the transform is received and displayed, by clicking on a pixel of the sinogram we can see which line of the image the pixel is referring to (shown by a green line). The application also shows the intensity values along this line on a chart under the image. It's also possible to query only one projection with a given angle, in this case the result is shown on a separate chart. To implement these charts the JavaScript library *Chart.js* was used (https://www.chartjs.org/).

In its current state the application is only for local usage by running the server on the local computer. The implementation of the server is based on the *http.server* module of Python, which might not be appropriate for production. (See the warning on the documentation page
https://docs.python.org/3/library/http.server.html.)

The software created via this approach is thoroughly described and documented (both from the user's and the developer's point of view) in [4]. A screenshot of the application can be seen in Figure 4.

3.3. Our second web-based approach: client-side, analytical

The main lessons learned from the first approach are as follows. On one hand, the scene editing methodology with the use of *Fabric.js* is highly desirable, it works well for this purpose, and follows the latest web development standards. Therefore, we use it also in our second approach. On the other hand, the freely available numerical transformation methods (especially when client-server communication is also involved) are too slow for real-time visualization purposes. Thus we conclude that the calculation of the Radon transform should be also done on the client side (preferably implemented in JavaScript for wide accessibility), but appropriate limitations shall be made on the details of the transform shown, as a trade-off to enable real-time calculation.

We observe that for introductory visualization purposes the *support* of the Radon transform of an object carries the most information, i.e. where is the value of the transform zero or non-zero. Thus for our purposes we now set aside the need for the calculation of the exact values inside the support of the Radon transform, rather focus on finding only its boundary. The analytical

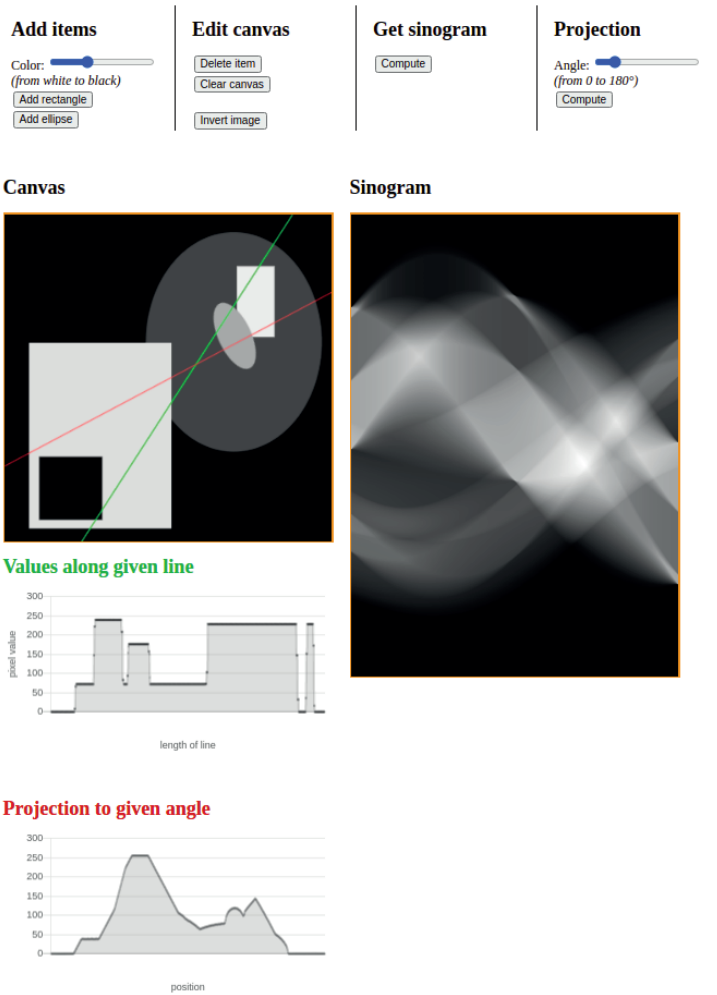

Figure 4. Screenshot of the application of our first approach (numerical, server-side calculation).

calculations in case of ellipses and rectangles are manageable and are discussed in more detail in Section 4. We conclude that finding the locations of appropriate points of interests for about 40 different angles distributed equally in $[0, \pi)$ the transform may be displayed with sufficient accuracy, and in real-time (near real-time using older hardware). Although the calculated points are just connected with straight line segments, human perception 'corrects' the visual information to observe the actual curves. Figure 5 shows the mentioned points of interest via two examples. The significant increase in calculation speed is due to the fact that instead of finding the transform's values at e.g. 400×400 points, rather the positions of about 100 to 200 points are located (independent of the screen resolution or image size). This is considered as the main idea of this approach.

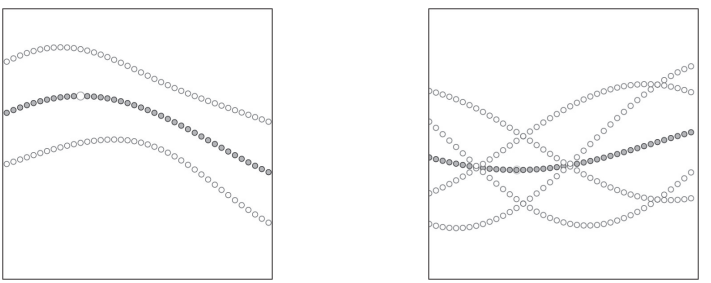

Figure 5. Points of interest calculated analytically on the boundary and inside the support of the Radon transform in case of an ellipse (left) and a rectangle (right).

The application works as follows. On the left-hand-side panel (canvas) one can edit the 'scene' or 'image': add and remove ellipses or rectangles, move, resize and rotate them. One may also clear the scene to start over. On the right-hand-side panel the Radon transform of the present objects is shown, such that every modification made on the scene is immediately reflected on the transform too. Furthermore, if a position is pointed on the transform's panel, then the corresponding line is shown on the image panel. Since intensity does not play a role in this approach, instead of a grayscale visualization, the objects may have their colors chosen from a predefined set of colors[1], and the transform will have the same color as the corresponding object for easy identification. A screenshot of the application can be seen in Figure 6. The application is available at

https://locsi.web.elte.hu/radon/

The Reader is encouraged to navigate to this page and try it out!

[1]The currently available colors are based on and named after the characters in the animated series *Sunny Bunnies* by Digital Light Studio (Minsk, Belarus, 2015).

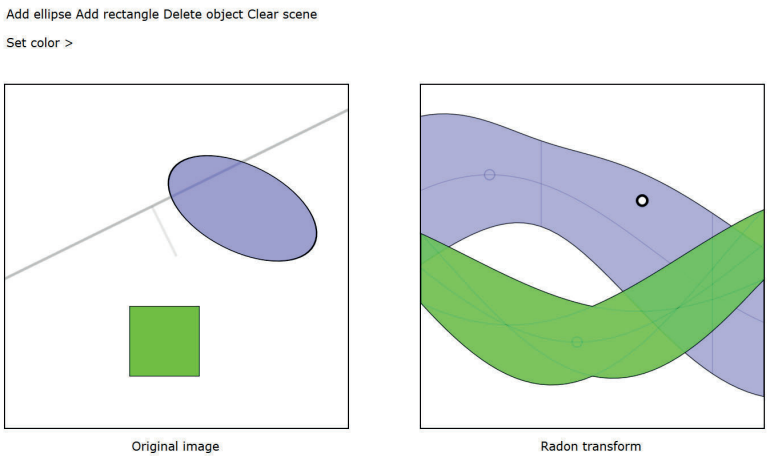

Figure 6. Screenshot of the application of our second approach (analytical, client-side calculation).

4. Analytic calculation of the support of the transform

In this section we will formulate the mathematical problem of finding the boundary of the support of the Radon transform, and deduce its solution step-by-step in case of ellipses and rectangles.

Remark that the Radon transform is 2π-periodic with respect to φ, furthermore, since $\mathcal{R}f(\varphi+\pi, r) = \mathcal{R}f(\varphi, -r)$ its domain is commonly considered to be $[0,\pi)\times\mathbb{R}$. With this in mind the transform is shown only on the region where $\varphi\in[0,\pi)$. But a point on the image may be located at any angle (in polar coordinates) in $[0,2\pi)$, and the (mathematical) atan function (to determine the angle by the Cartesian coordinates) gives values between $(-\pi/2,\pi/2)$. Considering these differences, in an actual implementation care must be taken to appropriately find the transform to display. It can be usually handled by a single conditional statement. In our implementation the `atan2` function was utilized. These minor issues will be neglected in the below discussion.

Let us first formulate the notions of the support of the Radon transform and its boundary. We will consider the function f to be compactly supported, usually a characteristic function of a simple geometric object (circle, ellipse, square, rectangle). For simplicity we may speak of the Radon transform of a circle, instead of the Radon transform of the characteristic function of a circle.

Definition 4.1. The *support of the Radon transform*

$$s\mathcal{R}f := \operatorname{supp}\mathcal{R}f = \left\{\,(\varphi,r)\in\mathbb{R}^2\,:\,\mathcal{R}f(\varphi,r)\neq 0\,\right\}.$$

The *boundary* of the support of the Radon transform is defined topologically: $\partial s\mathcal{R}f$ contain the points that are in the closure of $s\mathcal{R}f$ but not in the interior of $s\mathcal{R}f$.

Clearly for all angles $\varphi \in \mathbb{R}$, there are lines with normal angle φ that intersect a given object of compact support, and the set of corresponding $r \in \mathbb{R}$ values is bounded.

Definition 4.2. The *upper and lower boundary radius functions* for a compactly supported function f are

$$r^-(\varphi) := r^-(\varphi, f) = \inf \{\, r \in \mathbb{R} \,:\, \mathcal{R}f(\varphi, r) \neq 0 \,\},$$

$$r^+(\varphi) := r^+(\varphi, f) = \sup \{\, r \in \mathbb{R} \,:\, \mathcal{R}f(\varphi, r) \neq 0 \,\}.$$

It is easy to see that this gives us a more analytic approach for the boundary in polar coordinates:

$$\partial s\mathcal{R}f = \bigcup_{\varphi \in \mathbb{R}} \left\{\, (\varphi, r^-(\varphi, f)),\ (\varphi, r^+(\varphi, f)) \,\right\}.$$

Proposition 4.1. *For a fixed point on the plane at polar coordinates* (φ_0, r_0) (*one may consider a Dirac delta function here*), *the lines crossing this point are given by parameters* $(\varphi, r(\varphi))$, *such that* $r = r(\varphi) = r_0 \cdot \cos(\varphi - \varphi_0)$. (*Let us call such an* $r(\varphi)$ *a radius function.*)

Proposition 4.2. *Consider the characteristic function* f *of a circle of radius* $a > 0$ *centered at the origin. Then*

$$r^-(\varphi) = -a \quad \textit{and} \quad r^+(\varphi) = a.$$

This may be shortly written as $r^{\pm}(\varphi) = \pm a$ (constant).

Proposition 4.3. *Consider now the characteristic function* f *of a circle of radius* $a > 0$ *centered at the point with polar coordinates* (φ_0, r_0). *Then*

$$r^{\pm}(\varphi) = r_0 \cdot \cos(\varphi - \varphi_0) \pm a.$$

The above 3 claims are trivial, their proofs are left as an exercise. These already solve the problem in case of circles. To express the solutions for $r^-(\varphi)$ and $r^+(\varphi)$ in case of ellipses is not that simple, but not very difficult either.

Lemma 4.1. *Consider the characteristic function* f *of an ellipse located at the origin with principal semi-axes* a (*in the direction of the horizontal axis*) *and* b (*in the direction of the vertical axis*). *Then*

$$r^{\pm}(\varphi) = \pm\sqrt{b^2 \sin^2 \varphi + a^2 \cos^2 \varphi}.$$

Proof. This formula may be deduced using coordinate geometrical considerations. The parameterization of the line $L(\varphi, r)$ can be written as

$$(x, y) = (r \cdot \cos\varphi - t \cdot \sin\varphi,\ r \cdot \sin\varphi + t \cdot \cos\varphi) \qquad (t \in \mathbb{R}).$$

Substituting this into the equation $\dfrac{x^2}{a^2} + \dfrac{y^2}{b^2} = 1$ of the ellipse we get

$$\frac{1}{a^2} \cdot (r \cdot \cos\varphi - t \cdot \sin\varphi)^2 + \frac{1}{b^2} \cdot (r \cdot \sin\varphi + t \cdot \cos\varphi)^2 = 1.$$

We want to find such values of r that this equation has exactly one solution in t (the line is a tangent line of the ellipse). Collecting the terms for t, we get the quadratic equation:

$$A \cdot t^2 + B \cdot t + C = 0 \qquad \text{with} \qquad A = \left(\frac{\sin^2(\varphi)}{a^2} + \frac{\cos^2(\varphi)}{b^2}\right),$$

$$B = \left(\left(\frac{1}{b^2} - \frac{1}{a^2}\right) \cdot 2r \cdot \sin\varphi \cdot \cos\varphi\right) \text{ and } C = \left(\frac{r^2\cos^2\varphi}{a^2} + \frac{r^2\sin^2\varphi}{b^2} - 1\right).$$

By trigonometric identities the discriminant $B^2 - 4AC$ being zero simplifies to

$$\frac{r^2}{a^2b^2} = \frac{\sin^2\varphi}{a^2} + \frac{\cos^2\varphi}{b^2},$$

thus the statement of the Lemma holds. ■

Now to find the upper and lower border functions for arbitrarily translated and rotated ellipses is straightforward.

Theorem 4.4. *Let us now consider the characteristic function f of an ellipse centered at the point with polar coordinates (φ_0, r_0), with principal semi-axes a (pointing in the direction γ) and b. Then*

$$r^{\pm}(\varphi) = r_0 \cdot \cos(\varphi - \varphi_0) \pm \sqrt{b^2\sin^2(\varphi - \gamma) + a^2\cos^2(\varphi - \gamma)}.$$

Proof. Starting from the statement of the above Lemma, and taking into account the effects of translation as in our earlier Claims, it is only left to consider the effect of rotating the ellipse around its center point. Clearly this does not change the position of its center point and the added or subtracted values also stay the same, only occur at a line with angle changed against the angle of rotation. ■

Remark that in case of $a = b$ (both in the Lemma and in the Theorem) we get the result already presented for circles.

Now we turn our attention to rectangles. In this case the boundary radius functions may be expressed as minimum and maximum of the radius functions of the vertices of the rectangle. Appropriate calculation of the positions of the vertices is what makes our formulas complicated in general: we can not do the calculations directly in polar coordinates, we need the conversion to and from Cartesian coordinates.

To find the boundary radius functions, the order of the calculation should be as follows: (1) Find the values 'half diagonal length' and 'angle' describing the rectangle in a useful way when also rotation may occur. (2) Find the positions of the vertices starting from the center point of the rectangle in Cartesian coordinates, considering also the rotation of the rectangle. This is the key step in the procedure. (3) Convert the Cartesian coordinates of the vertices to polar coordinates. (4) Thus the radius function may be written for each vertex. (5) The boundary radius functions can be found as minimum and maximum of the radius functions of the vertices. These are all simple steps by themselves, but they must be combined appropriately in order to achieve the desired result.

Finally let us formulate and summarize the thoughts and mentioned calculations of the previous paragraphs in the following theorem.

Theorem 4.5. *Let us consider the characteristic function of a rectangle centered at the point with polar coordinates* (φ_0, r_0) *with semi-side lengths* a *and* b*, rotated with angle* γ*. Then*

$$r^-(\varphi) = \min\{\, r_1(\varphi),\, r_2(\varphi),\, r_3(\varphi),\, r_4(\varphi)\,\},$$
$$r^+(\varphi) = \max\{\, r_1(\varphi),\, r_2(\varphi),\, r_3(\varphi),\, r_4(\varphi)\,\},$$

with the radius functions of the vertices $r_i(\varphi) = r_i \cdot \cos(\varphi - \varphi_i) \quad (i = 1, 2, 3, 4)$*, where*

$$r_i = \sqrt{x_i^2 + y_i^2} \quad \textit{and} \quad \varphi_i = \operatorname{atan} \frac{y_i}{x_i} \qquad (i = 1, 2, 3, 4),$$

minding the different quadrants, calculated from the Cartesian coordinates of the vertices

$$(x_1, y_1) = \Big(r_0 \cdot \cos\varphi_0 + c \cdot \cos(\gamma + \mu),\ r_0 \cdot \sin\varphi_0 + c \cdot \sin(\gamma + \mu)\Big),$$
$$(x_2, y_2) = \Big(r_0 \cdot \cos\varphi_0 - c \cdot \cos(\gamma - \mu),\ r_0 \cdot \sin\varphi_0 - c \cdot \sin(\gamma - \mu)\Big),$$
$$(x_3, y_3) = \Big(r_0 \cdot \cos\varphi_0 - c \cdot \cos(\gamma + \mu),\ r_0 \cdot \sin\varphi_0 - c \cdot \sin(\gamma + \mu)\Big),$$
$$(x_4, y_4) = \Big(r_0 \cdot \cos\varphi_0 + c \cdot \cos(\gamma - \mu),\ r_0 \cdot \sin\varphi_0 + c \cdot \sin(\gamma - \mu)\Big),$$

where $c = \sqrt{a^2 + b^2}$ *the half-diagonal of the rectangle, and* $\mu = \operatorname{atan} \dfrac{b}{a}$ *the half-angle of the diagonals.*

An additional observation as a consequence of showing the transform only for $\varphi \in [0, \pi)$, i.e. on a half-period, is that the boundary radius functions satisfy an interesting 'paired half-periodic' boundary condition, namely

$$r^{\pm}(0) = -r^{\mp}(\pi), \quad \text{and} \quad (r^{\pm})'(0) = -(r^{\mp})'(\pi),$$

provided that the derivatives exist. (Among the above discussed situations they do not exist only in case of rectangles when φ is parallel or perpendicular to the sides.) Both algebraic and visual verification of this claim is possible. This observation also helps in an implementation of a proper visualization, specifically on the edges of the viewport.

5. Summary and further research

Our goal to create a tool for the Radon transform which provides real-time scene editing and transform visualization, publicly available to everyone is considered reached. Finally, we settled for a web-based, client-side, analytic calculation of the boundary of the support of the Radon transform of simple objects (ellipses and rectangles) and sets thereof. We claim that—as of today—this may be the best approach for introductory educational and science communication purposes about the Radon transform. The main milestones of the posed and met requirements are very briefly summarized in Table 1, we refer to Section 3 for the detailed discussion.

Requirement	Octave	Matlab	Server-side, numerical	Client-side, analytical
Easily accessible	no	no	almost	**yes**
Interactive editing	no	no	yes	**yes**
Real-time transform	no	yes	no	**yes**
Detailed transform	yes	yes	yes	**supp**

Table 1. Overview table of requirements.

The profound effect of these applications on the accessibility of the Radon transform is highlighted by the fact that to create e.g. a lung phantom (cross section of the body, with lungs and spine, c.f. [1]) and to examine the basic properties of its transform now becomes a child's play: see the example in Figure 7 which may now be created by anyone within minutes.

It is worth mentioning that students may get a new insight also to some basic geometric problems such as locating the common (inner or outer) tangent lines of two circles (or even ellipses).

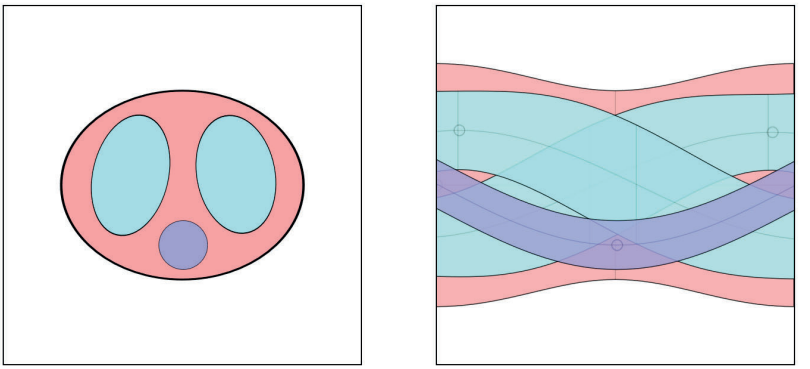

Figure 7. An interactive lung phantom on the web.

As directions of possible future research we mention the following:

- May the effect of discretization or the presence of noise be modeled following our approaches?
- Would GPU acceleration have any advantage here?
- Could we give some indication about the varying values of the Radon transform within its support while preserving the advantages of real-time, web-based calculations? The application of (approximate) level-lines may be considered. This points out a quite interesting problem by itself: locate the points of the Radon transform of an ellipse (or rectangle), where the intensity value is e.g. 50% of the maximal value.
- The web page containing the current implementation should be further developed: add a short description, links, more options, colors, dynamic layout, save & load functionality...
- Add support for the Radon transform of further objects: triangles etc.

Since the main web page for the project at https://locsi.web.elte.hu/radon/ may be subject to continuous improvement, the version currently discussed is saved for reference under https://locsi.web.elte.hu/radon/macs2020/.

Acknowledgement. This project was presented at the 13th Joint Conference on Mathematics and Computer Science (MaCS 2020) on October 3rd, 2020. We would like to thank the organizers at ELTE Eötvös Loránd University, Budapest and BBTE Babeș–Bolyai University, Cluj-Napoca for their work that made this online event possible!

References

[1] **Bognár, G.,** A no-reference image quality metric with application in low-dose human lung CT image processing, *Int. J. of Advances in Telecommunications, Electrotechnics, Signals and Systems* **5(1)** (2016), 1–7.

[2] **Bakonyi, V., T. Szabó and Z. Illés,** A real-time tool integration for lectures, *15th Int. Conf. on Emerging eLearning Technologies and Applications* (*ICETA*), Stary Smokovec (2017), 1–6.

[3] **Deans, S.R.,** *The Radon Transform and Some of Its Applications*, John Wiley & Sons (1983).

[4] **Gál, Zs.,** *Theory, Visualization and Applications of the Radon Transform* (Hungarian), MSc thesis (supervisor: Lócsi, L.), ELTE Faculty of Informatics, Department of Numerical Analysis, Budapest (2020).

[5] **Gonzalez, R.C. and R.E. Woods,** *Digital Image Processing,* Pearson, 3rd edition (2007).

[6] **Radon, J.,** Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten, *Ber. über Verh. Königlich-Sächsischen Ges. Wiss. Leipzig,* **69** (1917), 262–277.

[7] **Ramlau, R. and O. Scherzer,** The first 100 years of the Radon transform, *Inverse Problems,* **34(9)** (2018), 1–4.

[8] **Shepp, L. and B.F. Logan,** The Fourier reconstruction of a head section, *IEEE Transactions on Nuclear Science*, **21(3)** (1974), 21–43.

[9] **Zaidi, H.,** *Quantitative Analysis in Nuclear Medicine Imaging*, Springer US (2006).

Zs. Gál and L. Lócsi
ELTE Eötvös Loránd University
Budapest
Hungary
gal.zsofi@gmail.com
locsi@inf.elte.hu