Topino: A Graphical Tool for Quantitative Assessment of Molecular Stream Separations

Sven Kochmann, Nikita A. Ivanov, Kevin S. Lucas, Sergey N. Krylov*

Department of Chemistry and Centre for Research on Biomolecular Interactions, York University, Toronto, Ontario M3J 1P3, Canada;

*Corresponding author's email: skrylov@yorku.ca

ABSTRACT: In molecular-stream separation (MSS), a stream of a multi-component mixture is separated into multiple streams of individual components. Quantitative evaluation of MSS data has been a bottleneck in MSS for decades as there was no conventional way to present the data in a reproducible and uniform fashion. The roots of the problem were in the multi-dimensional nature of MSS data; even in the ideal case of steady-state separation, the data is three-dimensional: intensity and two spatial coordinates. We recently found a way to reduce the dimensionality *via* presenting the MSS data in a polar coordinate system and convoluting the data *via* integration of intensity along the radius axis. The result of this convolution is an angulagram — a simple 2D plot presenting integrated intensity vs angle. Not only does an angulagram simplify the visual assessment, but it also allows the determination of three quantitative parameters characterizing the quality of MSS: stream width, stream linearity, and stream deflection. Reliably converting an MSS image into an angulagram and accurately determining the stream parameters requires an advanced and user-friendly software tool. In this technical note, we introduce such a tool: the open-source software Topino available at https://github.com/Schallaven/topino. Topino is a stand-alone program with a modern graphical user interface that allows processing an MSS image in a fast (<2 min) and straightforward way. The robustness and ruggedness of Topino were confirmed by comparing the results obtained by three users. Topino removes the analytical bottleneck in MSS and will be an indispensable tool for MSS users with varying levels of experience.

INTRODUCTION

In molecular-stream separation (MSS), a stream of a multi-component mixture is separated into multiple streams of individual components (Figure 1). MSS can be achieved by chromatographic or electrophoretic means; the methods are called continuous annular chromatography and continuous flow electrophoresis, respectively, the latter is also called free-flow electrophoresis.[1–3] MSS has great potential for continuous flow synthesis which requires continuous and seamless downstream separation.[4–7] Methodology and instrumentation for MSS were advanced significantly during decades of research.[3,8–17] However, the realization of MSS potential has been hindered by a lack of an approach for uniform and reproducible quantitative assessment of MSS data, which is the subject of this work.

Uniform quantitative assessment of MSS requires: (*i*) consistent and robust acquisition of MSS images (MSS raw data), (*ii*) the availability of quantitative parameters characterizing the quality of MSS, and (*iii*) robust and rugged way of extracting these parameters from the images. We have been addressing the above requirements in a systematic way during the last few years. First, we developed an image processing and analysis system that is cost-effective and applicable to different geometries of MSS device.[18,19] Furthermore, this imaging system is compatible with analyzing fluorophores and chromophores including those absorbing only in the UV range.[20,21] Second, we introduced a manner of convoluting MSS images into simple 2D plots called *angulagram* (Figure 1).[22,23] Convolution is done by using a polar coordinate system with its origin at the inlet and subsequent integration of signal along the radius. In addition, we introduced a set of quantitative parameters for stream characterization (deflection, width, and linearity) that can be extracted from angulagrams for the quantitative assessment of MSS quality along with stream resolution.[20] This novel way of convoluting and assessing MSS images has been successfully adopted by another research group.[24]

The third requirement — a robust and rugged way of extracting stream parameters from the images — was not addressed so far. This procedure involves a set of precision-

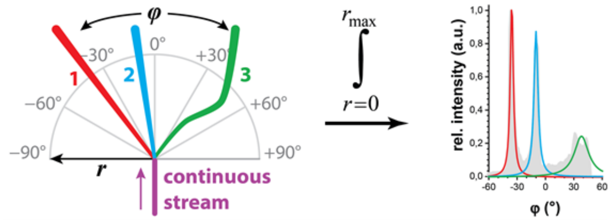


Figure 1. Molecular stream separation example and evaluation principle using angulagrams. Instead of a classical Cartesian coordinate system, a polar coordinate system is used to describe data (left). In a polar coordinate system, (ideal) streams run along the *r*-axis. Therefore, the data can be convoluted by integration over r into a clear 2D plot with peaks (right).

demanding tasks making the development of a robust and rugged parameter-extraction method difficult. We developed a reference method, which served as a proof-of-principle for the algorithmization of the procedures.[23] The reference method consists of different tools interacting in a non-ideal workflow and requiring manual transfer of data. The results depend on the user's experience and skills, which is a potential source of misinterpretations of the experimental data and, consequently, misconceptions. An ideal parameter-extraction method should provide the results in a robust and rugged fashion independent of its user. This independence on the user skills and experience requires a straightforward analysis workflow that can compensate for small variations potentially introduced by users. Such a workflow can only be implemented with a comprehensive software package consisting of a single and complete program that allows users to perform the following tasks: (*i*) process images, (*ii*) define a polar coordinate system, (*iii*) create angulagrams, and (*iv*) extract stream parameters. This program and its features need to be precise yet easily accessible to facilitate a robust and rugged quantitative assessment of MSS. A graphical user interface (GUI) is the most accessible input paradigm since

MSS evaluation deals with images, *i.e.* graphics. Thus, the proposed program should be implemented with a GUI and, furthermore, should be distributed in a way that minimizes the need for external libraries or an additional installation. These requirements make a Python environment, which was used for the reference method,[23] not suitable.

Creating such a program has three challenges. First, angulagrams require the use and visualization of a polar coordinate system while GUIs and images are usually based on the Cartesian coordinate system. Second, an accessible and precise interface must be deployed while accessibility and precision in user interfaces often are mutually excluding. Third, a coherent, straightforward, and fast (lower minute range) workflow of creating and evaluating an angulagram from an input image must be implemented, while this workflow consists of a set of diverse and precision-demanding tasks (see above). Here, we report on overcoming these challenges and introducing an accessible program Topino that allows users to assess MSS data quantitatively by creating and evaluating angulagrams from MSS images. For development, we used a modern version of C++ in combination with the Qt framework library and Eigen 3 math library.[25,26] The result is an easy-to-distribute program with a modern and clear dark-themed GUI that runs on all current versions of major operating systems (Windows, Linux, and macOS). In this work, we discuss the major points of Topino's implementation such as the visualization of the polar coordinate system as well as the coherent and straightforward workflow. Furthermore, we demonstrate its applicability to example images from our and others' previous works and provide a brief outlook on the future development of Topino.

METHODS

Creating an Angulagram. The process of angulagram creation is described in our previous work in detail.[22] Briefly, an input image (MSS raw data) is converted to grayscale, the region-of-interest (ROI) is extracted, and the angulagram function is calculated. All these calculations are automatically done by Topino internally. The user only needs to define the ROI by creating and aligning the polar coordinate system on the image (see Results and discussions section). Additionally, the results of intermediate calculations can also be viewed and exported from Topino if needed.

Conversion to Gray Scale. A pixel in a typical input image is a combination of three colors: red, green, and blue (RGB). Ideally, an input image only has one color channel (intensity) for data analysis. Thus, typical input images (color images with 3 channels) have to be converted to an ideal input image (grayscale image with 1 channel). This conversion process is called decolorization or desaturation.[27] Typically, a linear combination of RGB values is used to generate a grayscale value, *e.g.* by calculating and using the luminance of a pixel by:

$$L(x,y) = 0.21R(x,y) + 0.72G(x,y) + 0.07B(x,y) \qquad (1)$$

where x and y are the pixel coordinates in Cartesian coordinates. $R(x,y)$, $G(x,y)$, and $B(x,y)$ are the red, green, and blue intensity values at (x,y) resulting in the luminance value $L(x,y)$, which is used as gray value at (x,y). There are several conversion methods available and the best applicable method depends strongly on the image. For instance, the luminance value uses 72% of the green channel signal and only 7% of the blue channel signal. Hence, luminance values are ideal for green fluorescence images, but basically useless for blue fluorescence images. Therefore, we implemented several different conversion methods in Topino from which the user can select. Moreover, the gray values (= signal intensities) can be inverted to ensure that low signal intensity equals low gray value and high signal intensity equals high gray value. Finally, the gray value levels can be adjusted to remove unwanted noise and background.

Extraction of ROI. The ROI is extracted as a polar coordinate-based image (short: polar image). The calculation of the polar image is done by step-by-step iteration over all radii r (with a 1-pixel step) and angle φ values (with a 0.1° step) in their respective boundaries (as defined by the ROI) and assigning the intensity from a point at $(x(\varphi,r), y(\varphi,r))$:

$$I(\varphi,r) = I(x(\varphi,r), y(\varphi,r)), \quad \varphi = \varphi_{\min}...\varphi_{\max}, \, r = r_{\min}...r_{\max} \qquad (2)$$

where $x(\varphi,r)$ and $y(\varphi,r)$ are calculated as:

$$\begin{aligned} x(\varphi,r) &= x_0 + \mathrm{round}(r \times \cos\varphi) \\ y(\varphi,r) &= y_0 - \mathrm{round}(r \times \sin\varphi) \end{aligned} \qquad (3)$$

where x_0 and y_0 are the coordinates of the origin of the polar coordinate system. Function *round*(...) is a function that rounds a real value to the nearest integer since pixel coordinates are integers. The minus sign in the second line is required since the y-axis of the computer image runs from top to bottom in contrast to a standard Cartesian coordinate system, which runs from bottom to top.

Calculating the angulagram function. The polar image is convoluted by integration over the radius to generate the angulagram function $I(\varphi)$, which is calculated by

$$I(\varphi) = \int_{r_{\min}}^{r_{\max}} I(\varphi,r)\mathrm{d}r \approx \sum_{r_{\min}}^{r_{\max}} I(\phi,r) \qquad (4)$$

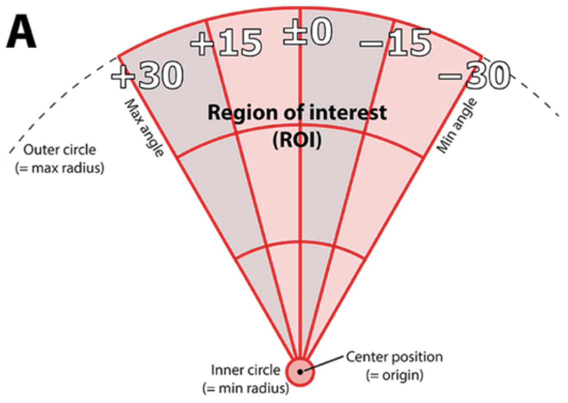
where $r_{\min}$ and $r_{\max}$ are the radii boundaries of the ROI and, thus, of the polar image. $I(\varphi,r)$ is the intensity in the polar image at the respective angle and radius. Internally discrete values are used for φ and r due to the integer representation of pixel coordinates. Therefore, the integration is basically the same as the summation over all values of r for a given φ.

Software and sample data availability. Topino is publicly available 64-bit software released under the 3-Clause BSD license. It can be used, copied, and distributed freely. Binary files for Windows and Linux (Debian package), source code, a user guide, and example files are available on the Github repository (https://github.com/Schallaven/topino). Furthermore, snapshots are available as supplementary files on ChemRxiv.[28]

The Windows version of Topino runs on both Windows 7 and Windows 10, the Linux version runs at least on Ubuntu 16 and Linux Mint 19, and the macOS version runs at least on MacOS X 10.12 (Sierra). The source code will allow other researchers and developers in the area of MSS to contribute to, implement, and extend analysis routines. Also, it is possible to compile and run the program on other than the listed above platforms.

RESULTS AND DISCUSSION

Polar Coordinate System. In order to evaluate an input image (MSS raw data), the inlet position and the ROI have to be defined. Since MSS evaluation with angulagrams uses polar coordinates, the ROI needs to be defined by a polar coordinate system with the origin at the inlet position. In Topino, the polar coordinate system is represented by a small inner circle (covering the actual inlet) and a larger outer circle (Figure 2). Both circles have the exact same center position that defines the inlet position and, in turn, the origin of the coordinate system. The visible part of the outer circle defines the ROI by its 0° angle direction, min/max angle, and min/max radius



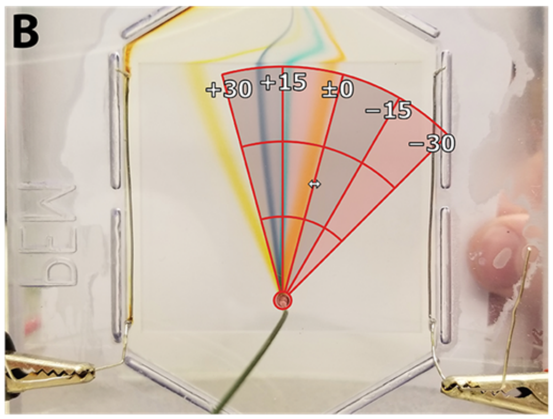


Figure 2. Schematic of the polar coordinate system (**A**) and an example as shown in Topino (**B**).

(Figure 2A). All these elements can be manipulated by simply clicking in or on them and dragging with the mouse (Figure 2B). For instance, the position of the origin can be changed by clicking into and dragging the area of the smaller circle. Likewise, the minimum radius or the 0° angle direction can be changed by clicking and dragging the border of the inner circle or the 0° angle line, respectively. For advanced users, Topino also includes options to input precise values for these parameters individually. Only the ROI area is used to generate the angulagram, *i.e.* all pixels inside the smaller circle as well as all pixels outside of the whole polar coordinate system representation are ignored (Figure 2A).

Conclusively, the visualization and implementation of the polar coordinate system in Topino allow users to define the ROI, remove background such as the non-signal pixels of the physical inlet as seen on the example image in Figure 2B. Furthermore, the ability to move and rotate the coordinate system and the ROI freely avoids potential data-changing preprocessing steps such as cutting or rotating the image. Furthermore, the graphical representation of the polar coordinate system is fast, accessible, and precise since it can be viewed and manipulated directly on the input image as well as by inputting number values if needed.

Angulagram-Creation Workflow. Our reference method for angulagram creation was published as Python programs that use a command-line interface (CLI).[18−23] A typical command line to process an input image *input.jpg* with our Python program *angureflexin.py* for creating an angulagram looks like

```
angureflexin.py --sat -200
       --inlet 2056 2184 --zone 1500 1800
       --levels 180 235 input.jpg
```

Here, "*--sat*" defines the (de)saturation level of −200; "*--inlet*" defines the inlet position $x = 2056$, $y = 2184$; "*--zone*" defines a rectangular zone which is 1500-pixel-wide and 1800-pixel-high with the inlet at the bottom middle, *i.e.* from (1306, 384) to (2806, 2184) in Cartesian coordinates; and "*--levels*" define the gray values between 180 and 235 used for evaluation. The respective value for each of these parameters has to be extracted using a graphics program such as *Adobe Photoshop*. When one value is determined, it has to be manually noted down in order to transfer it later to the Python program as shown in the example above. Subsequently, the next value is determined and noted down. These two steps are repeated until all required parameters are determined. These steps result in a non-straightforward workflow (Figure 3A).

Furthermore, the reference-method workflow is disconnected due to the manually noting down of values as well as the change in input paradigm from GUI (graphics program) to CLI (Python programs). This disconnection is even more apparent when errors occur. For instance, it is easy to transpose, forget, or mistype digits or signs when noting down values by hand. Such errors might not be visible immediately but become apparent further downstream in the workflow. For correction, the user has to track the source of error upstream or simply repeat the whole workflow again. For a novice user, such errors are hard to track down and time-consuming to fix, which discourages from the use of the program as a result. For advanced users, these errors are easier to spot and correct; however, correction still takes time and breaks the workflow significantly.

Subsequent to *angureflexin.py*, a second Python program, *evolutin.py*, is run to evaluate the resulting angulagram. This program will create a simple text file with all the determined stream parameters for further processing. In contrast to *angureflexin.py*, this second Python program does not require any more parameters that have to be pre-determined. All in all, the complete workflow (Figure 3A) based on the reference method takes around 10 min per input image for a trained person if no error occurs.

In contrast, Topino was optimized and tailored towards a short and straightforward general workflow (Figure 3B). After starting Topino, the user is presented with an empty

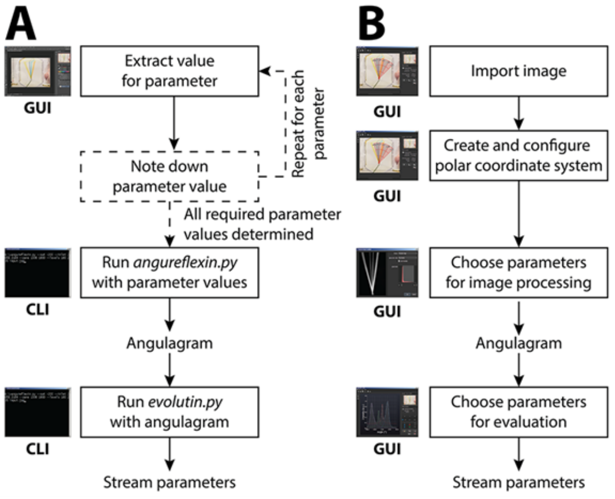


Figure 3. Workflow for angulagram creation and evaluation using the reference method[21] (**A**) and Topino (**B**). The dashed lines depict points of disconnection in **A** (see text for more details).

workspace. First, an image has to be imported. Second, the polar coordinate system has to be created and configured. Third, the image processing parameters such as desaturation mode and color level boundaries for the image need to be chosen to ensure the correct generation of an angulagram. Fourth, the angulagram is generated and evaluated by fitting the peaks to extract the respective stream parameters. The resulting stream parameters can be exported or simply copied to other applications for presentation or further processing. The whole optimized workflow takes less than 2 min for a user that has gone through this workflow once or twice in Topino. Furthermore, this workflow is straightforward and connected since all values are generated and processed inside a single GUI environment removing the need for manually noting down anything (Figure 3B).

Comparison of Evaluation Results with the Reference Method. Topino generates results close to those generated with the reference method when evaluating the same image. For example, a quick re-evaluation of the input image of Figure 2A in Ref. 22 with Topino lead to stream deflections of −10.8°, −0.8°, +6.6°, and +20.5° (Ref. 22: −12.5°, −1.6°, +6.4°, and +21.2°), stream widths of 8.3°, 3.1°, 3.8°, and 7.3° (Ref. 22: 8.0°, 2.5°, 3.2°, and 6.2°) as well as stream linearity of $L^2 \geq 0.95$ (Ref. 22: $L^2 \geq 0.95$). Similar results are expected due to the same underlying basic algorithm. The differences (*e.g.* in-stream deflection of the first stream) are mainly due to minor improvements and additional features in Topino such as being able to select a lower boundary for the radius of the ROI (to remove the physical inlet pixels from evaluation) and more grayscale conversion methods that are better applicable to this input image (here, *max intensity* with inverted color levels from 60 to 210 was used).

As a second example, a quick re-evaluation of the data in Figure 5A in Ref. 19 (original experiment from Jezierski *et al.*[29]) with Topino leads to stream deflections of −7.7°, −3.2°, and +2.9° (Ref. 19: −7.5°, −3.5°, and +2.7°), stream widths of 4.0°, 5.1°, and 3.2° (Ref. 19: 4.6°, 3.5°, 3.0°) as well as stream linearity of $L^2 \geq 0.98$ (Ref. 19: $L^2 \geq 0.98$). The grayscale conversion method *max intensity* with full-color level range (0 to 255) was used. In the input image, there is a barely visible fourth stream (Rhodamine 6G). This barely visible stream can be evaluated with Topino by optimizing the parameters of the grayscale conversion method (here, *max intensity* with color levels from 0 to 60 was used). For this fourth stream, the deflection and width were +8.0° (Ref. 19: +8.0°) and 4.6° (Ref. 19: could not be evaluated due to only small shoulder peak signal), respectively. Again, the overall results are expectedly similar, and the differences are due to improvements and additional features in Topino, which, in this case, allow the full analysis of the input image. Both input images can also be found in Figure 4A and B, respectively.

Comparison of Evaluation Results Obtained by Different Users. We used four different input images (Figure 4, Input image A–D) to study the impact of three different users (the first three authors of this work) on the assessment results. Based on our experimental experience (streams usually occur between −45° and +45°) and analytical-chemical standard error margins (2%), we expect the assessment of MSS to be robust and rugged if the deviations in-stream deflection, width, and linearity are ≤1.8° (2% of 90°), ≤1.8° (2% of 90°), and ≤0.02 (2% of linearity range), respectively. We used two images (A, C) from our own research[20,21] and two images (B, D) from others' research in this field[29,30], *i.e.* four different images in total, as input images in Topino. The images were acquired by different techniques (reflectometry and fluorimetry) and differed also in the quality (noise, focus, etc.) to cover practical and non-ideal situations. Each user approached the assessment of these four input images on

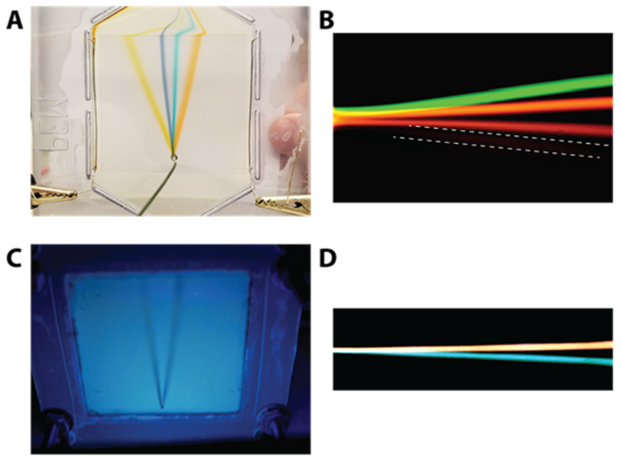


Figure 4. Input images used to compare evaluation results by different users. **A:** High-quality reflectometric image (original experiment by us) from Ref. 20. Adapted with permission of The Royal Society of Chemistry **B:** High-quality fluorescence image (original experiment by Jezierski et al.) from Ref. 29. This image contains one barely visible stream (dashed lines). Adapted with permission from Springer Nature. **C:** High-noise fluorescent image (original experiment from us) from Ref. 21. This image contains a lot of noise and non-uniform lightning. Adapted with permission from American Chemical Society. **D:** High-quality fluorescence image (original experiment by Schasfoort et al.) from Ref. 30. Adapted with permission of The Royal Society of Chemistry.

their own without any knowledge of the others' assessment approach. The skill levels between the three applicants varied: expert (= developed the software), advanced user (= knows MSS but has limited experience in MSS evaluation), and beginner (= just started in the field of MSS).

All three users were able to assess and extract stream parameters for all four images. The evaluation files of all applicants (including the input images) and statistical data are available in the Supporting Information. Across all images and users, the average standard deviation on stream deflection, width, and linearity were ±0.4°, ±0.8°, and ±0.02, respectively. These values are below the above thresholds and, thus, demonstrate the robust and rugged assessment using Topino in general. Looking at each image evaluation individually, the results slightly change as described in the following. For Input image A, the advanced user systematically found much lower widths and linearities for three of the streams leading to slightly higher deviations in the width (up to ±2.2°) as well as in linearity (up to ±0.06). We attribute this underestimation to the usage of non-optimal desaturation parameters by the advanced user due to lack of MSS evaluation experience. For Input image B, neither the beginner nor the advanced user could extract stream parameters for the barely visible stream. Here, the expert had to use two different sets of desaturation parameters to gain values for the three highly visible streams and the one barely visible stream, respectively. Both the beginner and advanced user only used one set and, therefore, were unable to evaluate the barely visible stream. We attribute this non-evaluation to the lack of MSS evaluation experience of the non-experts. Input image C contains much noise as it was taken under non-uniform lightning conditions making it challenging to choose good desaturation and fitting parameters, in particular for an inexperienced user. Here, only the expert is expected to extract reasonable values for stream parameters. Nonetheless, all three users were able to extract stream parameters in Topino with deviations near the ideal thresholds (±0.9, ±1.8°, and ±0.02 for deflection, width, and linearity, respectively). These results demonstrate that the

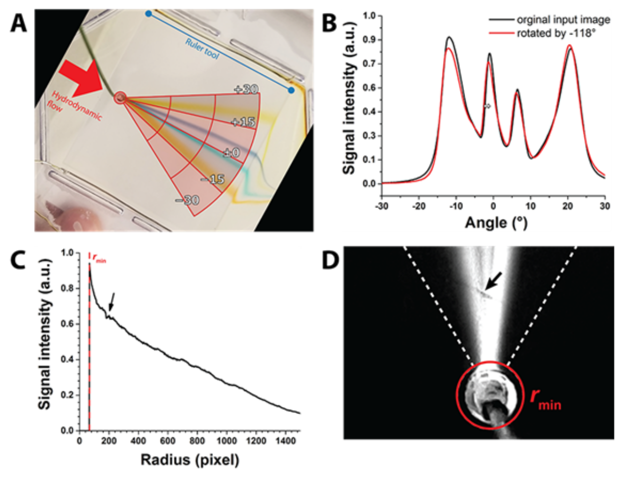


Figure 5. Evaluation example using some advanced features of Topino. Rotated input image (by −118°) can be precisely evaluated using the ruler tool to align the polar coordinate system with the hydrodynamic flow (**A**) leading to basically the same angulagram as the original input image (**B**). Radialgrams reveal debris (arrow in **C** and **D**) and other issues of the input image quality.

usage of Topino minimizes the negative impact of technical and personal variations. Input image D could be evaluated by all users with optimal deviations (±0.1, ±0.7°, and ±0.01 for deflection, width, and linearity, respectively).

Overall, the low standard deviations demonstrate that the assessment of MSS with Topino is robust and rugged towards users of different skill levels. Expectably, non-optimal evaluation conditions and lack of MSS evaluation experience result in increased deviations. This increase is minimized by the usage of Topino due to its accessible and coherent workflow.

Compensating and Spotting Variations and Errors with Advanced User Features. There are more features and functions in Topino in addition to the above-discussed ones. For instance, there is a ruler tool beside the inlet tool allowing defining lines on an image. These lines can be snapped to density points on the image and, thus, used to precisely align the parameters of the polar coordinate system to features on the image. As an example, we rotated the input image from Figure 2A in Ref. 22 by −118° in order to create a case of non-ideal input image (Figure 5A). In such a case, it is difficult for the user to set the direction of the 0° angle perfectly to the direction of the hydrodynamic flow (which is the reference for 0°). However, by using the ruler tool, the user can create and snap a line on the boundary of the separation zone and use this line as reference angle line for the polar coordinate system. This procedure leads to basically the same angulagram as with the non-rotated input image and comparable settings (Figure 5B).

Furthermore, a radialgram can be generated and displayed in Topino. A radialgram integrates the polar image over the angle instead of the radius, which can be used to follow the progression and dispersion of streams through the ROI. A typical radialgram is shaped as a half-peak that follows progressive loss of signal intensity due to molecular diffusion of streams (Figure 5C). Any deviations from this expected shape can point to debris found on the MSS device (sudden narrow drops), non-uniformity in lightning or exposure (overall curve shape change), or errors in pre-processing of the image (random peaks or drops) (Figure 5D).

Conclusively, these advanced functions are not needed for everyday assessment of MSS but are power features for advanced users that need more precision and information for difficult-to-assess border cases.

Limitations and Future Development of Topino. While Topino increases the accessibility of MSS evaluation, it is less flexible and versatile than our previously developed Python programs (reference method). For instance, automation and batch evaluation is not straightforward to implement and, therefore, not available at this time. However, we plan to continue the development of Topino to add such features and more in the upcoming months. In addition to the batch function to semi-automatically process a set of files, we would like, for instance, to add the ability to dissect complex peaks as described in our previous publication.[22] For this development, we want to encourage and invite everyone who is interested to contribute ideas, code, and bug reports to our public software repository at https://github.com/Schallaven/topino.

CONCLUSION

In this work, we introduced and presented the open-source software Topino. Topino is a stand-alone program with a modern graphical user interface that allows processing an MSS image in a fast (<2 min) and straightforward way. The robustness and ruggedness of Topino were confirmed by comparing the results obtained by several users. Topino removes the analytical bottleneck in MSS and will be an indispensable tool for MSS users with varying levels of experience. Having all requirements satisfied for uniform quantitative assessment of MSS, we foresee that the release of Topino will stimulate the usage and development of advanced MSS analysis using angulagrams and, in turn, aid MSS development itself.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on ChemRxiv (**DOI:** 10.26434/chemrxiv.13515086).

AUTHOR INFORMATION

Corresponding Author

*E-mail: skrylov@yorku.ca.

ORCID

Sven Kochmann:	0000-0001-7423-4609
Nikita A. Ivanov:	0000-0002-0842-6626
Kevin S. Lucas:	0000-0001-7389-3952
Sergey N. Krylov:	0000-0003-3270-2130

Author Contributions

Sven Kochmann planned and developed the software. *Nikita A. Ivanov* helped with the development of the graphical user interface, bug fixing, and wrote the user guide. *Kevin S. Lucas* helped with bug fixing as well as suggestions regarding the user interface and user guide. *Sergey N. Krylov* supervised the project and development and provided funding. All authors contributed to writing the manuscript. All authors read and approved the final manuscript.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

This work was supported by the Natural Sciences and Engineering Research Council of Canada (Grant SPGP 521331-2018). We thank *Anne Clinio*, *Jake Vanderplas*, *Dr. Janine George*, *Marcus Schmid*, *Dr. C. Titus Brown*, and *Dr. Victoria Stodden* for discussions and help on selecting the right software license for Topino.[31–33]

REFERENCES

(1) Hilbrig, F.; Freitag, R. *J. Chromatogr. B* **2003**, *790*, 1−15.

(2) Novo, P.; Janasek, D. *Anal. Chim. Acta* **2017**, *991*, 9−29.

(3) Johnson, A. C.; Bowser, M. T. *Lab Chip* **2018**, *18*, 27−40.

(4) Agostino, F. J.; Cherney, L. T.; Galievsky, V.; Krylov, S. N. *Angew. Chem. Int. Ed.* **2013**, *52*, 7256−7260.

(5) Jezierski, S.; Tehsmer, V.; Nagl, S.; Belder, D. *Chem. Commun.* **2013**, *49*, 11644−11646.

(6) Castro, E. R.; Manz, A. *J. Chromatogr. A.* **2015**, *1382*, 66−85.

(7) Pfeiffer, S. A.; Rudisch, B. M.; Glaeser, P.; Spanka, M.; Nitschke, F.; Robitzki, A. A.; Schneider, C.; Nagl, S.; Belder, D. *Anal. Bioanal. Chem.* **2018**, *410*, 853−862.

(8) Islinger, M.; Wildgruber, R.; Völkl, A. *Electrophoresis* **2018**, *39*, 2288−2299.

(9) Preuss, J.-A.; Nguyen, G. N.; Berk, V.; Bahnemann, J. *Electrophoresis* **2020**, *accepted*. DOI: 10.1002/elps.202000149.

(10) Jender, M.; Novo, P.; Maehler, D.; Münchberg, U.; Janasek, D.; Freier, E. *Anal. Chem.* **2020**, *92*, 6764−6769.

(11) Zhou, W.; Xia, L.; Xiao, X.; Li, G.; Pu, Q. *Electrophoresis* **2019**, *40*, 2165−2171.

(12) Staubach, S.; Reiter, C.; Weber, G.; Giebel, B. *Cytotherapy* **2019**, *21*, 57−58.

(13) Rudisch, B.; Pfeiffer, S. A.; Geissler, D.; Speckmeier, E.; Robitzki, A. A.; Zeitler, K.; Belder, D. *Anal. Chem.* **2019**, *91*, 6689−6694.

(14) Zitzmann, F. D.; Jahnke, H.-G.; Pfeiffer, S. A.; Frank, R.; Nitschke, F.; Mauritz, L.; Abel, B.; Belder, D.; Robitzki, A. A. *Anal. Chem.* **2017**, *89*, 13550−13558.

(15) Courtney, M.; Thompson, E.; Glawdel, T.; Ren, C. L. *Anal. Chem.* **2020**, *92*, 7317−7324.

(16) Herzog, C.; Poehler, E.; Peretzki, A. J.; Borisov, S. M.; Aigner, D.; Mayr, T.; Nagl, S. *Lab Chip* **2016**, *16*, 1565−1572.

(17) Kristoff, C. J.; Bwanali, L.; Veltri, L. M.; Gautam, G. P.; Rutto, P. K.; Newton, E. O.; Holland, L. A. *Anal. Chem.* **2020**, *92*, 49−66.

(18) Kochmann, S.; Krylov, S. N. *Lab Chip* **2017**, *17*, 256−266.

(19) Kochmann, S. *Schallaven/ffeimg: Image Processing and Analysis System for Development and Use of Free Flow Electrophoresis Chips.* Github/Zenodo **2017**. DOI: 10.5281/zenodo.1001785.

(20) Ivanov, N. A.; Liu, Y.; Kochmann, S.; Krylov, S. N. *Lab Chip* **2019**, *19*, 2156−2160.

(21) Ivanov, N. A.; Kochmann, S.; Krylov, S. N. *Anal. Chem.* **2020**, *92*, 2907−2910.

(22) Kochmann, S.; Krylov, S. N. *Anal. Chem.* **2018**, *90*, 9504−9509.

(23) Kochmann, S. *Schallaven/angulagram: Quantitative Characterization of Molecular-Stream Separation.* Github/Zenodo **2019**. DOI: 10.5281/zenodo.2592588.

(24) Lu, N.; Sticker, D.; Kretschmann, A.; Petersen, N. J.; Kutter, J. P. *Anal. Bioanal. Chem.* **2020**, *412*, 3559−3571.

(25) The Qt Company. *Qt: Cross-platform Application and UI Framework.* **2020**. https://www.qt.io/

(26) Guennbaud, G.; Jacob, B. *Eigen v3.* **2010**. https://eigen.tuxfamily.org/

(27) Smith, K.; Landes, P.-E.; Thollot, J.; Myszkowski, K. *Comput. Graph. Forum* **2008**, *27*, 193−200.

(28) Kochmann, S.; Ivanov, N. A.; Krylov, S. N. *ChemRxiv* **2021**. DOI: 10.26434/chemrxiv.13515086.

(29) Jezierski, S.; Gitlin, L.; Nagl. S; Belder, D. *Anal. Bioanal. Chem.* **2011**, *401*, 2651−2656.

(30) Kohlheyer, D.; Besselink, G. A. J.; Schlautmann, S.; Schasfoort, R. B. M. *Lab Chip* **2006**, 6, 374−380.

(31) Stodden, V. *Int. J. Commun. Law Pol.* **2009**, 13, 22−46.

(32) Morin, A.; Urban, J.; Sliz, P. *PLoS Comput. Biol.* **2012**, 8(7): e1002598.

(33) Stodden, V. *Comput. Sci. Eng.* **2009**, 11, 35−40.