Resurrect3D: An Open and Customizable Platform for Visualizing and Analyzing Cultural Heritage Artifacts

Joshua Romphf*
Digital Scholarship
River Campus Libraries

Elias Neuman-Donihue †
Department of Computer Science
Department of History

Gregory Heyworth‡
Department of English

Yuhao Zhu§
Department of Computer Science

University of Rochester

ABSTRACT

Art and culture, at their best, lie in the act of discovery and exploration. This paper describes Resurrect3D, an open visualization platform for both casual users and domain experts to explore cultural artifacts. To that end, Resurrect3D takes two steps. First, it provides an interactive cultural heritage toolbox, providing not only commonly used tools in cultural heritage such as relighting and material editing, but also the ability for users to create an interactive "story": a saved session with annotations and visualizations others can later replay. Second, Resurrect3D exposes a set of programming interfaces to *extend the toolbox*. Domain experts can develop custom tools that perform artifact-specific visualization and analysis.

Index Terms: Visualization systems, cultural heritage, toolkits, 3D visualization, API, customization, custom shading, portability

1 INTRODUCTION

Art and culture, at their best, lie in the act of discovery: seeing what is hidden and rejecting the fallacy that what we see is all there is. Almost every great piece of art, artifact, or site has something hidden within it, invisible to the naked eye. Paintings have pentimenti — those rough drafts that the artist regrets and paints over; manuscripts have palimpsests or scratch-outs — text that is covered up or overwritten; and buildings and archeological sites are a world of unexpected labyrinths.

The discovery aspect of culture and art is inadequately addressed by museums today. Even with a guided tour, the museum experience is centered around receiving information rather than discovering new information. 3D digitization and visualization, however, are transforming this landscape, allowing user-centric exploration in both casual consumption and scientific research of cultural heritage.

In this paper, we describe Resurrect3D, an open platform for visualizing, analyzing, and ultimately freely exploring difficult-to-study artifacts. Resurrect3D is built around three key **design principles**.

Targeting Cultural Heritage. While using a generic system architecture and supporting general 3D visualization, Resurrect3D is designed specifically with cultural heritage in mind. For instance, we support processing data from a range of data acquisition pipelines such as LiDAR scanning, multispectral imaging, and photogrammetry; similarly, we provide well-optimized tools to annotate, analyze, and visualize 3D objects. These features allow domain experts to not only interact with the artifacts but also "see the invisibles" and analyze artifacts using advanced algorithms.

Portability. Our experience of developing visualization platforms for cultural heritage tells us that the end users are extremely diverse. Resurrect3D must be available across different computing platforms ranging from desktop PCs to smartphones and Virtual Reality devices. To this end, we made a judicious design decision to base the entire system on the Web technology stack (e.g., NodeJS, WebGL, WebXR), which is naturally platform-independent. The Web application will automatically get updated whenever the user launches the application, i.e., requesting the application through a URL, allowing for automatic security/privacy improvements.

Customizability. Resurrect3D also aims to provide a clean interface to allow domain experts to develop custom tools to visualize and analyze cultural artifacts. For instance, when a painting with pentimenti is captured through multispectral imaging, experts could implement a custom Principle Component Analysis tool to reveal certain surface details that are visible only with specific combinations of spectral bands, and then implement a custom shader to visualize different bands simultaneously in order to see the steps of the pentimenti's coming into being. In contrast, existing visualization platforms for cultural heritage generally do not provide the customizability, handicapping domain experts.

Resurrect3D is completely open source, allowing institutions to control the dissemination of digitized surrogates of their collection materials. Resurrect3D is live at: `https://resurrect3d.lib.rochester.edu/`, and the code is publicly available at `https://github.com/rochester-rcl/resurrect3d` (some features will be merged from the dev branches soon).

We have used Resurrect3D for a range of pedagogical, research, and entertainment applications. For instance, in the Lazarus project [7] we imaged and visualized, in Resurrect3D, the New York Public Library's Hunt-Lenox Globe [2], a minuscule (only 5 inches in diameter) globe depicting the Americas and one of the oldest terrestrial globes in existence. Custom relighting and image analyses implemented in Resurrect3D allow us to reveal surface details not apparent on the globe. Resurrect3D is also used by the Ward project [8] to visualize numerous biological specimens from the Ward's Natural Science Establishment [10]. Annotation and metering tools in Resurrect3D allows students and researchers to better understand the physiology of Ward's rare specimens.

This paper describes the system design of Resurrect3D. We focus on various design decisions we made in making Resurrect3D accessible to casual users and domain experts, and highlight how the customizability can enable artifact-specific visualization and analysis.

2 RELATED WORK

Generic 3D Visualization. Arguably the most popular Web-based 3D visualization platform is SketchFab [27], which is used by several cultural heritage institutions for sharing 3D models of their collection materials. SketchFab, however, is not without limitations. First, it is not open source, and requires a subscription-based payment for uploading and sharing works. Second, and more importantly, SketchFab is a generic visualization platform not designed with cultural heritage in mind and lacks field-specific functionality. In contrast, Resurrect3D provides a suite of tools that are more commonly used

*e-mail: jromphf@library.rochester.edu
†e-mail: eneumand@u.rochester.edu
‡e-mail: gregory.heyworth@rochester.edu
§e-mail: yzhu@rochester.edu

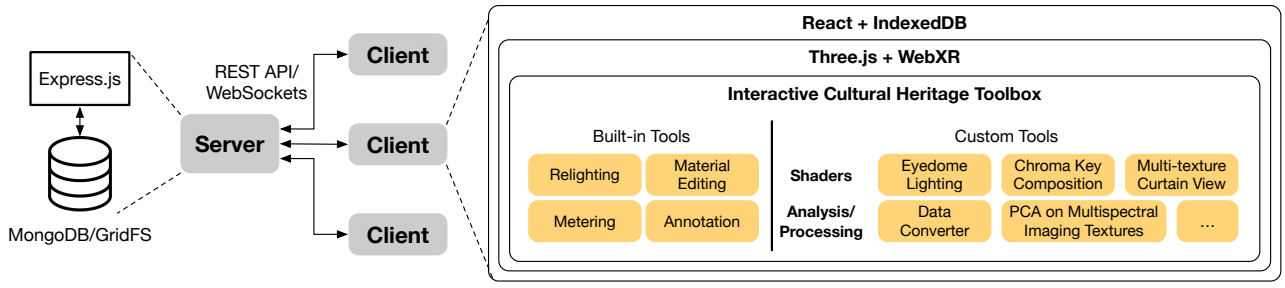

Fig. 1: Overview of the Resurrect3D system architecture.

in cultural heritage. These include interactive measurement, interactive relighting material editing to reveal surface details, image analysis, and annotation tools.

Image Viewers for Cultural Heritage There are a handful of image viewers designed for cultural heritage. They are usually highly specialized for working with Single Camera Multiple Light (SCML) data, which is obtained through Reflectance Transformation Imaging (RTI) [22] and Portable Light Dome (PLD) [21] and is useful in revealing surface details. Notable desktop tools include RTIViewer [14] and PLDviewer [3]. Web-based tools include Relight [23], Villanueva et al. [26], the Oxford RTI Viewer [4], and Pixel+ [19]. M.A.RL.I.E [20] supports not only RTI images but also shape and material representations with a flexible annotation system.

Resurrect3D supports SCML through Polynomial Texture Map (PTM) [22], which is a form of SCML data. The main difference between Resurrect3D and the SCML image viewers is that Resurrect3D supports other 3D modeling data and allows for developing custom tools to analyze 3D objects.

3D Visualization for Cultural Heritage. Smithsonian has recently developed Voyager [6], an open-source 3D platform. Much like Resurrect3D, Voyager provides material editing, relighting, measurement, and annotation tools. CyArk provides a proprietary 3D viewer for their collections [1]. The viewer is designed primarily for viewing 3D models without many of the features that Resurrect3D provides such as relighting and material editing.

Resurrect3D differs from these two systems in two main ways. First, Resurrect3D permits developing custom visualization (shading) and analysis tools, whereas neither system above has native support for customizability. We leverage the customizability and develop a set of tools such as Eyedome lighting and curtain view of multispectral textures, which are commonly used in cultural heritage visualization but are unavailable in either system. Second, both systems focus on visualizing mesh-based models whereas Resurrect3D supports a range of other data modalities commonly used in cultural heritage digitization such as multispectral imaging and SCML.

3DHOP [17] is a Web-based programming framework for 3D visualization, mostly used for cultural heritage. Through a declarative programming interface, 3DHOP provides a set of configurable knobs such as setting up camera and light parameters, picking points on the 3D model, and setting mesh transformation and visibility. These configurations are friendly to developers who are not familiar with graphics programming, but also limit the customizability to only what the configurable knobs offer. Resurrect3D, in contrast, provides customizability at a higher degree: allowing developers to directly write custom shaders and image analysis tools. Our approach provides more control over visualization and analysis, and thus allows for custom visualization such as multi-texture curtain view that is difficult to realize in 3DHOP.

Gamification Serious games for cultural heritage deliver pedagogical goals through games [12, 13, 15, 24]. While Resurrect3D itself is not a game, the visualization, interaction, and customization capabilities of the platform provide opportunities to develop series games for real cultural heritage sites and artifacts, which is our future work.

3 System Overview

The rendering system is built using the classic server-client architecture. Figure 1 shows an overview of the system.

Server. The server application is built using Express.js and Node.js for interfacing with the client requests and uses MongoDB as the backend database. Small metadata, such as annotations and session-specific settings (e.g., camera pose), are directly stored in MongoDB, while large objects (e.g., raw point clouds and texture maps) are stored using GridFS, which builds on top of MongoDB and is optimized for accessing large objects.

Clients communicate with the server in two ways, depending on the scenario. When a large chunk of data is requested, such as the initial request of a 3D model, the communication is through REST API, which is optimized for throughput. In contrast, we use WebSocket for subsequent communications that exchange small amounts of data but require real-time response, as WebSocket is optimized for latency. A typical scenario is when a teacher client sends her current state (e.g., camera pose, annotation) to student clients in the presentation mode.

Client. The client-side UI is built using the popular React library [5]. The rendering system is built on the popular three.js library [9] to leverage WebGL for GPU acceleration. The event-driven system in three.js integrates well with the asynchronous nature of React, allowing us (Resurrect3D developers) and future contributors to focus on developing functionalities specific to the cultural heritage domain. Finally, Resurrect3D also leverages WebXR to provide support for basic VR rendering of 3D objects.

We support two forms of client interaction. First, clients can perform completely independent interactions and analyses on an object. Second, clients can communicate with each other through the server, e.g., synchronizing different users to share the current state of one user's view. This can be useful in a classroom lecturing setting where students want to see what the teacher is seeing.

Finally, we also implement a basic client-side caching system using the IndexedDB API, which stores compressed model data and drastically cuts down on the load times and bandwidth required compared to always fetching the model directly from the server.

4 Interactive Cultural Heritage Toolbox

Apart from basic visualization capabilities, Resurrect3D provides an interactive cultural heritage toolbox which includes not only a set of built-in tools, but also generic programming interfaces that allow expert developers to develop custom analysis and visualization tools.

4.1 Built-in Functionalities for Cultural Heritage

4.1.1 Revealing Surface Details

A key requirement for cultural heritage visualization is to examine/reveal the surface details of an artifact to see the "invisibles". To that end, Resurrect3D provides a set of interactive tools ranging from relighting to material editing.

Fig. 2: A small wax seal from University of Rochester's Rare Books, Special Collections, and Preservation department. This example shows a combination of re-lighting and material editing (right) best increases the visibility of the seal's finer details compare to relighting alone (middle) and the oroginal rendering without any digital surface-revealing techniques (left).

Relighting. A well-known technique used by historians and archaeologists is to point a light source to a specific area of the surface to bring out the details. Resurrect3D allows users to digitally emulate this process by specifying various properties of the light source (e.g., color, intensity) and to move the light source around the scene to relight specific regions.

The key prerequisite of relighting is precise surface normals. Resurrect3D supports three different ways to obtain a normal map.

- Directly import an existing normal map.
- Infer the normal map from the texture map through Sobel filtering. This is useful when a direct normal map is unavailable.
- Generate a normal map from a Polynomial Texture Mapping (PTM) file [22]. This enables RTI for better relighting.

Interactive Material Editing. Another useful method that cultural heritage experts often use for revealing surface details is through editing the surface material of an artifact, such as the normal scale, metalness, and roughness values. For instance, changing the metalness essentially changes the albedo of the material, allowing fine-grained geometry to come out easily under certain light sources.

Critically, Resurrect3D allows users to arbitrarily combine interactive material editing and relighting to best bring out the details. The Figure above compares how a proper combination of relighting and material provides the best visual details.

4.1.2 Interactive Annotation, Metering, and Storytelling

Annotation. Cultural heritage experts usually enhance artifacts by adding metadata (e.g., texts, audio, images, video), which can later be disseminated for research or pedagogical purposes. Resurrect3D allows users to add annotations in a positional way, i.e., associating annotations with a particular position on the 3D model.

Positional annotation is implemented through a React callback that passes the click position to three.js, which translates the position to the coordinates in the camera space, which, in turn, get reprojected to the world space. This allows the annotation to be associated with the particular position of the model, invariant with the camera pose.

Metering. Digitizing artifacts brings an inherent benefit: accurate metering of the geometric properties of an artifact, such as the distance between two points, the surface area, or the volume. This metering capability is important especially for objects of extreme scales (large or small) or brittle under any physical measurement.

Resurrect3D provides an interactive measurement tool with multiple units. This is achieved using the same underlying implementation of position annotation: translating any selection from the user to coordinates in the camera space, which are then translated to the world space of the model, in which the actual measurement calculation, be it distance or area, is carried out.

Interactive Storytelling. Resurrect3D facilitates cultural heritage education by allowing one user to create a story, i.e., a saved session of annotations, measurements, material and lighting settings. The story can then later be launched by another user, essentially experiencing a virtual walkthrough of the 3D model with guided tools. At any point during the walkthrough, a user is able to apply custom interactions with the model, allowing users to explore on their own to validate/critique the pre-recorded instructions.

4.2 Customizability

A key differentiating feature of Resurrect3D is that it provides great flexibility for expert users to design custom tools. The interfaces for custom tooling can be classified into two main categories: 1) custom shaders, which allow visualizations of the 3D model different from the default shaders in three.js, and 2) custom analysis tools, which allow for statistical, image, and vision processing on the model/metadata. Enabling custom analysis tools extends Resurrect3D from a visualization platform to also be an analytics platform.

4.2.1 Custom Shaders

The cultural heritage community, over the years, has designed creative shading techniques to aid in viewing artifacts. Resurrect3D provides a straightforward interface for developing custom shaders. This is accomplished by leveraging three.js' capability of directly integrating a shader written in OpenGL Shading Language (GLSL).

The design goal of the interface is to allow developing the shaders without worrying about how the shader code will be integrated into the React application. To that end, we provide a template, which defines a "loader" function that takes a three.js object and adds the custom shader to the three.js namespace. The code snippet below illustrates this template. The only code that the programmers have to provide is the shader itself, which is a plain JavaScript object containing the uniforms and the GLSL shaders, which graphics programmers should be familiar with.

```
export default function loadNewShader(
    threeInstance: Object): Promise {
  return new Promise((resolve, reject) => {
    threeInstance.NewShader = {
      //user-define custom shader below
      uniforms: {...},
      vertexShader: ...,
      fragmentShader: ...
    }
    resolve(threeInstance);
  });
}
```

To improve performance, the loader function integrates the GLSL shader code with the main React UI through JavaScript Promises. That is, the loader function returns a Promise that resolves the three.js object. We use Promises for two reasons. First, it exposes an interface that is asynchronous by nature, which allows for additional functionality such as loading custom shaders from external sources

Fig. 3: A Graffito in Oplontis Villa A in Torre Annunziata, Italy. The fish on the graffito (left) is much more visible with the EDL shader (right).

(i.e. from a separate server or a content distribution network). Second, loaders can also be chained together with other asynchronous functions in the main React component (e.g., loading mesh data from the server) or executed in parallel (using the Promise.all method).

Automatic UI Integration. A custom shader and its various knobs, once defined, have to be added to the main application UI so that users can decide when and how to apply the shader. To allow developers to focus on the shader design without having to worry about the UI design, Resurrect3D exposes two APIs for programmers to specify the knobs for each custom shader; Resurrect3D then automatically adds the knobs to the application UI.

The first API allows programmers to define a UI group for a shader; all the knobs will show up in the group.

```
shaderGroup = new ThreeGUIGroup("shaders");
```

The second API allows programmers to specify a UI component with its name, type, and the various properties of the component, of which the main property is the callback functions to be called when corresponding user interactions are triggered.

```
shaderGroup.addComponent("intensity", components.
    THREE_RANGE_SLIDER, {
  //define the callback when a user slides the bar
  callback: ...,
  // define properties of a slider
  min: 0.0,
  max: 4.0,
  step: 0.1,
});
```

We provide three custom shaders in the current design of Resurrect3D. These shaders are readily useful on their own, but also serve as examples for user extensions.

EDL. Relighting is computationally intensive if the object is to be rendered photo-realistically. In many use-cases, however, photorealism is less important as long as surface details are distinguishable. A typical technique to achieve that is Eye-Dome Lighting (EDL), which is an image-space technique to assign different colors to pixels depending on the depth information. EDL is commonly used for quick detail revealing [18, 28]. Resurrect3D implements an interactive EDL shader following the algorithm by Boucheny [16]. Figure 3 shows the effect of our EDL shader in bring out the details of the fish on a Graffito in Oplontis, Italy.

Chroma Keying. To improve the contrast and readability of an object's surface details, historians often use an idea from the visual effects industry called Chrome Keying, which replaces a specific color with another color. Resurrect3D provides a custom Chroma Key shader, which allows users to change colors of arbitrary points on the mesh and to adjust the ratio between the original color and the replacement color for finer control over the end result.

Multi-texture Curtain View. Simultaneously examining multiple bands from multispectral imaging is a common strategy to explore different elements of the object. For instance, many paintings have pentimento, a first sketch that masters make on the canvas and that later is painted over and changed; multispectral imaging captures different layers on the painting as different images, examining which allows people to see backward in time what an object once was and to view the steps of its coming into being.

Fig. 4: Curtain-view mode. A redtail hawk from Ward's collection visualized with two texture maps (left). World map by Henricus Martellus with three color bands from multispectral imaging (right).

Through a custom shader, Resurrect3D provides what we call the curtain-view mode, which samples different regions of the rendered frame from different textures. As the mouse or touch gesture moves, the boundaries of the regions move accordingly, allowing users to understand and explore the history behind an artifact. Figure 4 shows two artifacts in the curtain-view mode: a redtail hawk model from the Ward project [8] and the famous world map by Henricus Martellus with three color bands from multispectral imaging [25].

4.2.2 Custom Processing and Analysis Tools

Oftentimes experts analyze and process the digital artifact before visualization. For instance, one could perform a Principle Component Analysis (PCA) on different bands captured through multispectral imaging to generate new texture maps for visualization.

Resurrect3D leaves a generic interface for developing custom analysis tools by exposing the model and metadata of an artifact to developers. The analysis tool and the main rendering code in three.js live in the same namespace. Thus the analysis tools can easily interoperate with the rendering pipeline.

Much like the custom shaders, the analysis tools also expose a simple Promise-based interface. The code below shows the programming interface, where the analysis function takes either a Mesh and/or its metadata, so that the function has access to all underlying geometry and material data, and returns a Promise that resolves an object with the same type as the input:

```
function myAnalysisFunction(object: THREE.Mesh |
    THREE.Group): Promise {
  return new Promise((resolve, reject) => {
    // define the analysis function here
    resolve(object);
  });
}
```

As an example, we provide perhaps the simplest example of an analysis tool, which imports and converts common 3D formats into a format that is required by the visualization system. Specifically, since Resurrect3D by default supports a physically based rendering (PBR) workflow, the conversion tool converts 3D formats (e.g., OBJ, FBX, VRML) into a range of PBR maps (e.g., diffuse and displacement maps) in the JSON Object Scene Format required by three.js. Wherever possible, we make use of the WebWorker [11] to parallelize and offload processing to background threads.

5 Conclusion

More than ninety percent of the objects in museums around the world will never be seen by the public, either for the lack of display space or because of damage or fragility. We envision a future where visualization platforms turn every museum and archive into an open university. To achieve that vision, Resurrect3D provides not only

the basic visualization and interaction capabilities, but also the customizability that allow domain experts to develop artifact-specific analysis and visualization tools.

REFERENCES

[1] CyArk 3D Explorer. https://cyark.org/explore/.

[2] Hunt-Lenox Globe. https://en.wikipedia.org/wiki/Hunt-Lenox_Globe.

[3] Manual PLDviewer7.0.05. Zenodo. http://doi.org/10.5281/zenodo.3693795.

[4] Oxford Reflectance Transformation Imaging Toolset. https://github.com/ksjogo/oxrti.

[5] React. https://reactjs.org/.

[6] Smithsonian Voyager. https://smithsonian.github.io/dpo-voyager/.

[7] The Lazarus Project. https://www.lazarusprojectimaging.com/.

[8] The Ward Project. https://wardproject.org/.

[9] three.js.

[10] Ward's Science. https://www.wardsci.com/store/.

[11] WebWorker API. https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers.

[12] E. F. Anderson, L. McLoughlin, F. Liarokapis, C. Peters, P. Petridis, and S. De Freitas. Developing serious games for cultural heritage: a state-of-the-art review. *Virtual reality*, 14(4):255–275, 2010.

[13] A. Antoniou, G. Lepouras, S. Bampatzia, and H. Almpanoudi. An approach for serious game development for cultural heritage: Case study for an archaeological site and museum. *Journal on Computing and Cultural Heritage (JOCCH)*, 6(4):1–19, 2013.

[14] J. T. Barron and J. Malik. Intrinsic scene properties from a single RGB-D image, 2016.

[15] F. Bellotti, R. Berta, A. De Gloria, A. D'ursi, and V. Fiore. A serious game model for cultural heritage. *Journal on Computing and Cultural Heritage (JOCCH)*, 5(4):1–27, 2013.

[16] C. Boucheny. *Visualisation scientifique de grands volumes de données: Pour une approche perceptive*. PhD thesis, Université Joseph-Fourier-Grenoble I, 2009.

[17] S. Duchêne, C. Riant, G. Chaurasia, J. L. Moreno, P.-Y. Laffont, S. Popov, A. Bousseau, and G. Drettakis. Multiview intrinsic images of outdoors scenes with an application to relighting. *ACM Trans. Graph.*, 2015.

[18] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[19] C. Innamorati, T. Ritschel, T. Weyrich, and N. J. Mitra. Decomposing single images for layered photo retouching. vol. 36, p. 15–25. Chichester, GBR, July 2017. doi: 10.1111/cgf.13220

[20] A. Jaspe-Villanueva, M. Ahsan, R. Pintus, A. Giachetti, F. Marton, and E. Gobbetti. Web-based exploration of annotated multi-layered relightable image models. *Journal on Computing and Cultural Heritage (JOCCH)*, 14(2):1–29, 2021.

[21] G. Liu, D. Ceylan, E. Yumer, J. Yang, and J. Lien. Material editing using a physically based rendering network. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2280–2288, 2017. doi: 10.1109/ICCV.2017.248

[22] Y. Liu, A. Neophytou, S. Sengupta, and E. Sommerlade. Relighting images in the wild with a self-supervised siamese auto-encoder. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 32–40, 2021.

[23] F. Ponchio, M. Corsini, and R. Scopigno. A compact representation of relightable images for the web. In *Proceedings of the 23rd International ACM Conference on 3D Web Technology*, pp. 1–10, 2018.

[24] Z. Shu, S. Hadap, E. Shechtman, K. Sunkavalli, S. Paris, and D. Samaras. Portrait lighting transfer using a mass transport approach. *ACM Transactions on Graphics*, 37, 2017.

[25] C. Van Duzer. *Henricus Martellus's World Map at Yale (c. 1491): Multispectral Imaging, Sources, and Influence*. Springer, 2018.

[26] A. J. Villanueva, R. Pintus, A. Giachetti, and E. Gobbetti. Web-based multi-layered exploration of annotated image-based shape and material models. 2019.

[27] X. Xiao and L. Ma. Color transfer in correlated color space, 2006.

[28] C. Zheng, T.-J. Cham, and J. Cai. T2Net: Synthetic-to-realistic translation for solving single-image depth estimation tasks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 767–783, 2018.