



# A Service-Based Preset Recommendation System for Image Stylization Applications

F. Fregien<sup>1</sup>, F. Galandi<sup>1</sup>, M. Reimann<sup>1</sup> <sup>a</sup>, S. Pasewaldt<sup>2</sup>, J. Döllner<sup>1</sup> and M. Trapp<sup>1</sup> <sup>b</sup>

<sup>1</sup>Hasso Plattner Institute, University of Potsdam, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany

<sup>2</sup>Digital Masterpieces GmbH, August-Bebel-Str. 26-53, 14482 Potsdam, Germany

**Keywords:** Image Stylization, Recommendation System, Microservice.

**Abstract:** More and more people are using images and videos as a communication tool. Often, such visual media is edited or stylized using software applications to become more visually attractive. The data that is produced by the editing process contains useful information on how users interact with the software and data yielding respective results. In this context, this paper presents a framework that facilitates data storage, data profiling, and data analysis of image-stylization operations, image descriptors, and equivalent usage data by means of a recommendation system. The presented concept is implemented prototypical and preliminary evaluated.

## 1 INTRODUCTION


**Motivation.** In recent years, the number of smart phone users increased continuously. With the number of mobile users is growing, so is the amount of visual media data, such as images or videos. Since usually every smartphone has a camera, there exist several million professional and hobby photographers around the world. To transform, enhance, or stylize photos, there are a variety of editing applications that enable applying filters or adjust, e.g., the contrast or brightness of an image. While highly-skilled users require low-level control over all filter options, a common user can be overwhelmed by the big amount of configuration options (Figure 1). It requires a time-consuming learning phase or trial and error to achieve satisfying results. For casual users, this can be a frustrating experience as they expect pleasant results fast.

To overcome this issue, we utilize an approach known as recommendation system. Recommendation Systems (RSs) is an active research field in the area of Machine Learning (ML). It has been widely applied in e-commerce and other online services used on an every-day basis, e.g., when listening to music (Moscatto et al., 2021), watching movies (Kumar et al., 2015), buying products (Acıar et al., 2006), or

being on social media (Sperlí et al., 2018). The goal of an RS is to predict users' preference and recommend the most relevant items to users. In order to provide content-based recommendations, the system must contain information about the users and items.

Further, understanding the raw data collected by a recommendation system and forming knowledge, is advantageous for a company or developers seeking to understand its users. For example, knowing what users like and how they interact with a piece of software is vital to adjust features and release planning. Furthermore, usage data enables the possibility to learn from expert users to eventually transfer useful insights like parameter settings for less professional users. A recommendation system is a useful tool to hide complex computations and settings to increase user experience and satisfaction. Such recommendation systems enable the implementation of so-called "one-click solutions" for choosing image filters based on user preferences or the image content. To enable such systems, a data storage framework has to facilitate data profiling and analysis.

**Problem Statement & Challenges.** To summarize the above, state-of-the-art in (mobile) image-filtering apps provide a high number of available filter operations that can be combined in various configurations, yielding numerous output possibilities (Fig-

<sup>a</sup>  <https://orcid.org/0000-0003-2146-4229>


<sup>b</sup>  <https://orcid.org/0000-0003-3861-5759>



Figure 1: Different configurations of image-stylization techniques allow graphically different results. These can generate numerous variants for an input image. The variants shown here were obtained by manually adjusting the process parameters.

ure 1). According to Isenberg, each individual operation is usually highly-configurable at different level-of-control (Isenberg, 2016): from selecting a number of customizable global presets, over controlling a number of global parameter values, to local adjustments (Dürschmid et al., 2017). While preset facilitate ease-of-use, the latter two enable a low level-of-control featuring highly individual results and styles. However, they also lead to increased operating complexity and often require time-consuming trial and error methods. With respect to this, major challenges represent the design and implementation of a system that enables the data collection and analysis as well as the provisioning of recommendations based on user characteristics and specifics inputs.

**Approach & Contribution.** To address the challenges above, this paper describes the architectural design and implementation of a microservice-based framework to store and retrieve usage data as well as a prototype of a recommendation system that proposes most used image operations or operation presets. To summarize, the paper makes the following contributions. It presents the concept of a recommendation system for image-stylization applications and

it describes a prototypical implementation based on a microservice architecture and framework. The remainder of this paper is structured as follows: Section 2 reviews some terminology and background in the context of microservice-based image processing as well as basic approaches and challenges for designing recommendation systems. In Section 3, we describe the workflow and implementation of how to collect, store, and analyze usage data. In Section 5, we evaluate the runtime performance of our system by means of an application example, and present aspects of future work. Section 6 concludes this paper.

## 2 BACKGROUND

This section briefly describes related and previous work on the topic of microservices for image processing, approaches for the collection and provisioning of user or usage data, and fundamentals of recommendation systems.

**Microservices for Image Processing.** In recent years, microservices have become increasingly popular. Unlike a monolithic system, microservices are autonomous modules that perform various tasks with respect to the application logic. The advantages of microservices are (1) scalability of the components, (2) easier deployment and maintainability as well as, (3) the possibility to introduce various technologies into one system (Viggiato et al., 2018). Despite the advantages, a microservice infrastructure introduces an information barrier between the services, that can have negative effects on system latency and performance (Shadija et al., 2017).

In our work, we are extending an existing microservice platform for cloud-based visual analysis and processing, which was first presented by Richter et al. (Richter et al., 2018). The microservice platform has been improved and further developed in several subsequent works. The core of this platform is the *Image Processor* service that applies *Operations* to images. The term “operations” refers to image filters that change the visual appearance as well as transformations, such as rotating, mirroring, or cropping. To even realize complex filters that, e.g., make use of Neural Style Transfer (NST) (Gatys et al., 2016), the Image Processor uses so-called *Visual Computing Assets* (VCAs) (Dürschmid et al., 2017). They are a textual representation, which allow to describe programmatically how an operation is applied on the Graphics Processing Unit (GPU) using shader programs. The specific results of an operation can be adjusted by using *Parameters*. A set of parameters

and an assignment of parameter values compose to a *Preset*, which are used to label specific parameter configurations that are proven to be aesthetic. Operations can even be chained in a *Pipeline* to create more advanced filters. Using Visual Computing Assets (VCAs), we are able to profile and analyze which operations pipelines, presets, or parameters are used most frequently.

**Collecting, Profiling, and Analyzing Data.** Data profiling is an important part of any data-collecting mechanism. By understanding the dataset and its metadata, the underlying database schema can be improved, design flaws can be detected and the data can be better prepared for further analyzing such as machine learning tasks (Abedjan et al., 2017). Data analysis helps to extract useful information from data and has a wide range of applications. Data mining as a method of data analysis is revealing interesting patterns in the data that can help improving business workflows, finding new scientific discoveries, or enhancing user experience (Han et al., 2011). Especially in the business domain, data analysis supporting better decision-making and showing improvements in the business logic (Xia and Gong, 2014). Related to web applications, collecting usage data can help improving user experience and usability of the application (Beri and Singh, 2013).

In our use case, we focus on analyzing operations, their presets and parameters, as well as how they are arranged in a pipeline by a user. Furthermore, we collect image descriptors, which contain several metadata about an image, such as Exchangeable Image File Format (EXIF) or Global Positioning System (GPS) data (Ölvecký and Host’ovecký, 2021). With these, the recommendation system can try to find correlations between operation configurations and images. To persist the collected usage data, we use a state-of-the-art storage approach: a Graph-based Database Management System (GDMS) (Huang et al., 2002). We decide to use Neo4j, as it is open-source and self-hostable, yet it is one of the most stable and mature GDMS that provides methods to run complex queries on related data.

**Recommendation Systems.** A recommender or recommendation system is suggesting solutions or proposing hints to solve a particular problem of a user (Ricci et al., 2022). Users often do not have the capability to make respective decisions because they are hindered by time constraints or lack of operating knowledge. Therefore, recommendation systems are useful tools to manage a large number of options and are supporting the decision-making process. Due to

the advantages for the user, recommendation systems are heavily used in commercial applications to increase user satisfaction and sales (Ricci et al., 2010).

There are three major types of RSs: (1) content-based RSs analyze the content of items to recommend items that are similar to the ones the user liked in the past, (2) Collaborative Filtering RSs makes predictions based on the preferences of other users with similar tastes, and (3) Hybrid Recommendation RSs that combine collaborative filtering and content-based methods to leverage the advantages of both.

Besides the commonplace “customers also bought, used or watched” recommendation, more complex systems are developed. Novel step-wise recommendations are designed to support users accomplishing tasks that consist of several steps instead of one (Nouri et al., 2020). This scenario resembles the decision-making process of choosing VCAs to achieve a visually aesthetic result given an input image. In this context, the workflow consists of two primary steps: (1) the user has to select a VCA, and (2) the user has to find proper parameter settings. Moreover, these two steps can reoccur several times forming a VCA pipeline.

Assisting the user in the choosing process poses many challenges. Besides a recommendation system, a manageable user interface must be implemented that hides the overwhelming complexity of the rendering techniques. To face the challenge in the user experience of image filtering apps, valuable insights comprising a featured iOS App, BeCasso (Pasewaldt et al., 2016) are obtained (Klingbeil et al., 2017).

### 3 SYSTEM

**Preliminaries & Assumptions.** For modeling low-level and high-level use cases for the proposed recommendation system, we make the following assumptions. First, we avoid to transfer the input image to the system due to the following reasons: (1) to be applicable in different, possibly world-wide scenarios, with varying data protection and data privacy regulations, (2) to account for possibly limited data throughput between device and service, and (3) to avoid requiring respective storage capacities. Instead, we rely on representing the input image characteristics using common low-level image feature descriptors that are computed on-device prior to transmission. Secondly, to enable the mapping of data transmission and retrieval to a user we assume a unique user token.

**System Overview.** Figure 2 outlines the interaction between a user, a Visual Media Abstraction System

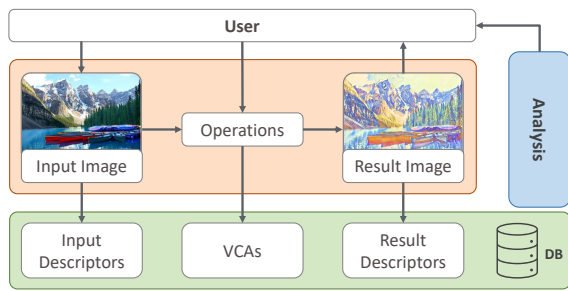


Figure 2: High-level outline of the framework for data storage and data analysis in the domain of visual computing applications (Visual Media Abstraction System (VMAS) is colored in orange and Data Storage and Analysis System (DSAS) is colored in green and blue).

(VMAS), and the Data Storage and Analysis System (DSAS), which are briefly described as follows:

**VMAS:** The VMAS computes the result image based on the input image and VCA selection. VCAs comprises all information needed to perform the computation including parameter values and additional metadata.

**DSAS:** This paper focuses on the DSAS which constitutes the framework for data profiling of image and VCA metrics. The term “framework” refers to a system that facilitates the development of analytics applications leveraging the frameworks Application Programming Interface (API) for storing and retrieving usage data.

**Usage Data Acquisition and Storage.** The user wants to edit an uploaded input image. The VMAS will provide a selection of VCAs and parameters to stylize the image according to the user’s input. On the basis of these VCAs, the VMAS computes the result image which is sent back to the user subsequently. During this process, there are several points where useful data for the later analysis is generated:

**Input Image Descriptors:** The actual input image contains useful image descriptors. These descriptors can be easily retrieved in case of general information descriptors, such as spatial resolution, filename or image type. Moreover, there are specific domain information descriptors that contain information about the image such as object recognition data.

**VCA Data:** Further, VCA data including selected parameters determine the computation of the resulting image. This data is most important for the later analysis as it tracks how the user interacts with the given VCAs and parameters to produce a certain result image.

**Output Image Descriptors:** The resulting image usually comprises the same type of descriptors as the input image. This information can be useful to understand the user’s perception of visual aesthetics. By comparing the input and output descriptors, more subtle insights can be detected about desired visual changes made by the user.

**Recommendation System.** The user wants a recommendation of suitable VCAs and parameter settings to produce a visually aesthetic result image. The recommendation is based on the analysis of the usage data, which is stored in the DSAS.

## 4 IMPLEMENTATION ASPECTS

This section covers implementation details of the fundamental software framework used (Section 4.1) and the database modeling (Section 4.2).

### 4.1 Software Framework

Figure 3 shows an overview of the proposed framework that basically consists of two microservices: the Usage Data Collector (UDC) and Usage Data Analyzer (UDA). Together these form the conceptual DSAS. Both have access to the same database, which holds the usage data at hand.

The framework has to provide an API to the raw data and general statistic routes. To achieve better (1) maintainability, (2) stability, and (3) separation of concern the data service and analytics are established as two separate services. Presuming there will be more analytics use cases in the future, only the UDA must be adapted, whereas the UDC remains unaffected. Furthermore, this separation has performance advantages in regard to handling incoming requests. Thus, the UDA can be specialized in receiving statistics requests and analyzing results. On the other hand, the UDC handles primarily usage data import requests. The general workflow of using this framework is as follows:

1. An application for image processing sends an Hypertext Transfer Protocol (HTTP) request containing image descriptors and a VCA pipeline configuration to the UDC.
2. The UDC validates the incoming usage data and stores it in the GDMS.
3. An application sends an HTTP request to the UDA in order to get recommendations, e.g., the most used operations for a specific user.

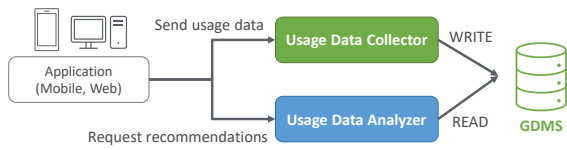


Figure 3: Overview of the services architecture and major requests.

4. The UDA runs an analysis on the usage data stored in the GDMS and sends the result as response to the application.

Note that the UDA only has to read data from the database, whereas the UDC must be able to insert data. Since this paper involves the mere collection and analysis of usage data, update and delete operations were omitted for the sake of brevity.

## 4.2 Database Modeling

This section describes the preliminaries and challenges for modeling the data base of the proposed recommendation system as well as implementation details specific to the individual services.

**Preliminaries and System Challenges.** Initially, a recommendation system is challenged by the following conceptual and technical aspects (Goyani and Neha, 2020), which can be aligned to our problem domain as follows:

**Cold Start Problem:** Recommendation techniques that rely on existing data, e.g., based on user interactions with an application, are not able to compute recommendations for a new user as no usage data about this user is collected yet. In this case, recommendations must be made based on other metrics, such as most used operations.

**Data Sparsity:** Some recommendation techniques use matrices to store and compute scores for, e.g., user-operation-relations, which indicate how much an operation is liked by a user. If many users interact with few operations only, this score matrix is sparse and can consume storage space unnecessarily.

	Neo4j	Dgraph	TigerGraph	JanusGraph
Open Source	☑	☑	✗	☑
Self hostable	☑	☑	✗	☑
Stable / Mature Ecosystem	☑	☑	☑	✗
API for JavaScript	☑	☑	✗	✗
Complex Graph / ML algorithms	☑	✗	☑	✗

Figure 4: Feature comparison of different graph-based databases with respect to their suitability for our approach.

**Scalability:** As computation performance decreases with the amount of stored data, it should be able to deploy more instances of the UDC and UDA microservices even on multiple machines in a cloud-based environment.

**Adaptability:** There are several approaches for computing recommendations (Das et al., 2017), yet we have no data indicating which approaches fit best for our use cases. Therefore, we want to be able to implement several approaches and compare them to each other, based on accuracy or performance. In this work, we are only implementing recommendations based on number or count of usage. But as a requirement, all services should be adaptable to support other approaches as well.

**Structural Flexibility:** Since the system should be expandable in terms of recommendation approaches, we require our data to have a flexible structure. Having no rigid schema, allows us to add or change properties or adapt the underlying storage structure on demand.

**Assessment of DBMS Approaches.** Prior to database implementation, we have to choose a suitable Database Management System (DBMS) approach. We assess the suitability between the following three approaches:

**Relational DB:** This classical DB approach offers only limited structural flexibility due to the usage of a static database schema. Further, it requires often expensive JOIN operations on large tables to perform query operations.

**Document-based DB (NoSQL):** While document-based approaches allow for structural flexibility due to schema less model, it is difficult to express and represent related data. All related data must be stored in a document, which means there are many duplicates which makes it hard to analyze the data.

**Graph-based DB (NoSQL):** Graph-based approaches allow for structural flexibility due to schema less model. Relations are represented as edges in a graph, which means there are no JOIN operations required to query related data. This makes querying usually more performant than in relational DBs, especially for big data amounts.

With respect to the latter approach, Figure 4 shows comparison results of four different alternatives.

**Database Schema.** Figure 5 shows the database schema developed for the prototypical implementation of our concept. It represents the following struc-

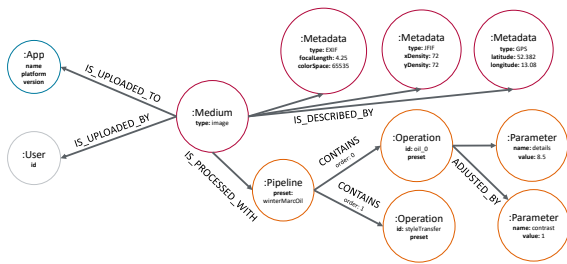


Figure 5: Database schema depicting nodes and relations for our recommendation system for image stylization approaches.

ture on a per-node basis. A medium (image) relates to an specific application that has uploaded the data and a user that initiated the editing. Further, a medium is described by to an number of metadata nodes, encoding standard attributes or descriptors. These relations can be extended by additional metadata format for future extension. Furthermore and foremost, each medium node relates with a pipeline node that processed this medium. A pipeline references the individual operations with its respective order of processing and the respective preset used. Each operation relates the to operation parameters that were manipulated by the user and its respective value.

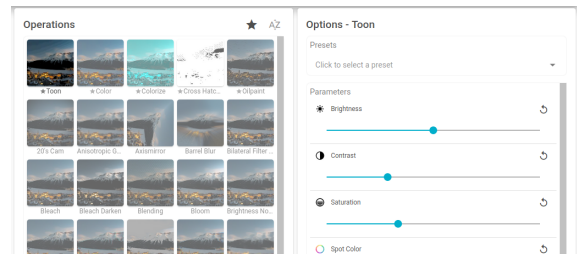
## 5 PRELIMINARY EVALUATION

This section describes and discusses the preliminary evaluation of our approach by means of application examples and performance measurements.

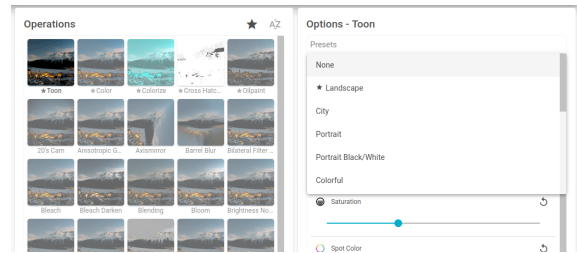
### 5.1 Application Example

In the following, the described microservice-based framework is integrated into a Web application, which is exemplary for any mobile or web application. The application requests the most used operations from the UDA to receive a sorted list of popular operations used. As shown in Figure 6, the list is used to place popular operations more prominently by ranking them before other operations and labeling them with a star symbol.

If the user wants to adapt the style of the image even further, they can use parameters or predefined presets. Again, the application requests the most used presets from the UDA to get a list of popular presets for the selected operation. Figure 6 shows how most used presets are highlighted in the application by adding the star symbols. In conclusion, a user can select a featured VCA and preset using a few clicks.



(a) Sorting after featured VCAs.



(b) Selection of featured preset.

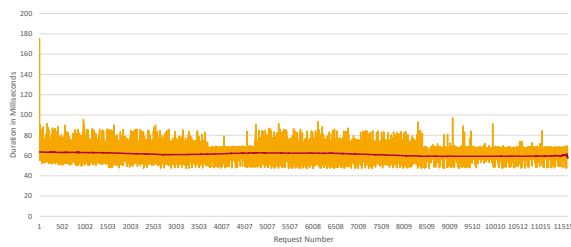
Figure 6: Integration of the prototypical implementation of the recommendation system into a web application for image stylization.

### 5.2 Performance Evaluation

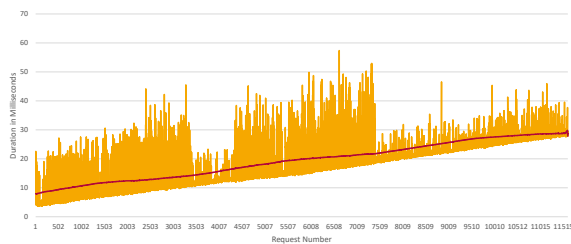
As part of a feasibility study, we tested the performance of the the UDC and UDA system components.

**Test Data & System.** For testings purposes we used a data set comprising 11 609 entries. The complete data set comprises 462 190 nodes in the database, approximately 40 nodes per entry. The nodes are distributed as follows (Figure 5): 472 application nodes, 11 594 medium respective pipeline nodes, 35 696 metadata nodes, 54 345 operation nodes, and 348 488 parameter nodes. For testing purposes, we deployed the UDC and UDA system components on a single machine with the following system specifications: Intel Core i9 10900K 3.70 GHz Central Processing Unit (CPU), 10 (20 logical) cores, and 64 GB Random Access Memory (RAM). The Node.js Version 18.6.0 runtime environment runs a Neo4j Version 4.4.7 database.

**Performance Results.** Figure 7 shows the resulting measurements of our performance experiment: the successive import of the 11 609 usage data entries into an empty Database (DB). On a per-run basis, we subsequently measured the runtime of (1) each data import-query (Figure 7) and (2) querying the Top-10 operations among all applications nodes (Figure 7). The test results show, that the duration of data import queries are almost constant at approximately 60 ms.



(a) Import usage data performance



(b) Retrieval of most-used operations performance

Figure 7: Performance measurements for database input and retrieval queries. Yellow lines show individuals timing while the line shows averaged values, respectively.

However, the runtime performance of retrieving the most-used operations increase linearly with the number of stored database nodes.

### 5.3 Future Work

In our framework, we implemented a basic recommendation system that evaluates and ranks operations and presets based on the amount they are used. This prototypical implementation can be extended by using more sophisticated state-of-the-art recommendation approaches, such as content-based, collaborative, or hybrid filtering (Das et al., 2017). This includes techniques that involve Machine Learning, e.g., for finding clusters and relations between images and VCAs and parameters that were used to stylize it.

In terms of scalability, the framework can be improved by integrating a caching or pre-computing workflow. Currently, the UDA analyzes the usage data on incoming requests. This means, the runtime performance of an analysis request increases with the amount of stored usage data. To improve the performance, the UDA could run the analysis asynchronously, e.g., by periodically pre-compute the most used operations and presets. The analysis results can then be cached and served constantly on request. This can be extended by using two database instances. The first database is used by the UDC to store all the usage data. Its performance is not degraded when an analysis is running. The second database that holds a copy of all the usage data, is able to mutate the data, e.g., by adding labels used for similarity compu-

tations. The performance of analysis is not degraded when new usage is collected at the same time. Additionally, no database transactions are required to synchronize read and write operations to guarantee consistency.

## 6 CONCLUSIONS

In conclusion, this paper presents a foundation for all kinds of data profiling tasks, as well as VCA and image data analytics. The presented framework contains a usage data storing process enabled by a Usage Data Collector microservice and the underlying Graph-based Database Management System. Moreover, the Usage Data Analyzer forms a basis for data analysis tasks and already includes a prototypical analysis for retrieving the most popular operations and presets. Finally, this work serves as an enabler for various future use cases comprising the data storage and analysis of image and VCA metrics.

## ACKNOWLEDGMENTS

This work was partially funded by the German Federal Ministry of Education and Research (BMBF) through grants 01IS18092 (“mdViPro”) and 01IS19006 (“KI-LAB-ITSE”).

## REFERENCES

- Abedjan, Z., Golab, L., and Naumann, F. (2017). Data profiling: A tutorial. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD '17*, page 1747–1751, New York, NY, USA. Association for Computing Machinery.
- Aciar, S. V., Zhang, D., Simoff, S. J., and Debenham, J. K. (2006). Recommender system based on consumer product reviews. *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, pages 719–723.
- Beri, B. and Singh, P. (2013). Web analytics: Increasing website’s usability and conversion rate. *International Journal of Computer Applications*, 72:35–38.
- Das, D., Sahoo, L., and Datta, S. (2017). A survey on recommendation system. *International Journal of Computer Applications*, 160:6–10.
- Dürschmid, T., Söchting, M., Semmo, A., Trapp, M., and Döllner, J. (2017). ProsumerFX: Mobile Design of Image Stylization Components. In *Proceedings SIGGRAPH ASIA Mobile Graphics and Interactive Applications (MGIA)*, pages 1:1–1:8, New York. ACM.
- Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks.

- 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2414–2423.
- Goyani, M. and Neha, C. (2020). A review of movie recommendation system. *ELCVIA: electronic letters on computer vision and image analysis*, 19(3):18–37.
- Han, J., Kamber, M., and Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.
- Huang, Z., Chung, W., Ong, T.-H., and Chen, H. (2002). A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '02, page 65–73, New York, NY, USA. Association for Computing Machinery.
- Isenberg, T. (2016). Interactive NPAR: What Type of Tools Should We Create? In Bénard, P. and Winnemöller, H., editors, *Non-Photorealistic Animation and Rendering*. The Eurographics Association.
- Klingbeil, M., Pasewaldt, S., Semmo, A., and Döllner, J. (2017). Challenges in user experience design of image filtering apps. In *SIGGRAPH Asia 2017 Mobile Graphics Interactive Applications*, SA '17, New York, NY, USA. Association for Computing Machinery.
- Kumar, M., Yadav, D., Singh, A. K., and Gupta, V. K. (2015). A movie recommender system: Movrec. *International Journal of Computer Applications*, 124:7–11.
- Moscato, V., Picariello, A., and Sperlí, G. (2021). An emotional recommender system for music. *IEEE Intelligent Systems*, 36:57–68.
- Nouri, E., Sim, R., Fournay, A., and White, R. W. (2020). Step-wise recommendation for complex task support. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*, CHIIR '20, page 203–212, New York, NY, USA. Association for Computing Machinery.
- Pasewaldt, S., Semmo, A., Döllner, J., and Schlegel, F. (2016). BeCasso: Artistic Image Processing and Editing on Mobile Devices. In *Proceedings of ACM SIGGRAPH ASIA Mobile Graphics and Interactive Applications (MGIA)*, pages 14:1–14:1, New York. ACM.
- Ricci, F., Rokach, L., and Shapira, B. (2022). *Recommender Systems: Techniques, Applications, and Challenges*, pages 1–35. Springer US, New York, NY.
- Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B. (2010). *Recommender Systems Handbook*. Springer-Verlag, Berlin, Heidelberg, 1st edition.
- Richter, M., Söchting, M., Semmo, A., Döllner, J., and Trapp, M. (2018). Service-based Processing and Provisioning of Image-Abstraction Techniques. In *Proceedings International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 97–106, Plzen, Czech Republic. Computer Science Research Notes (CSRN).
- Shadija, D., Rezai, M., and Hill, R. (2017). Microservices: Granularity vs. performance. In *Companion Proceedings of The 10th International Conference on Utility and Cloud Computing*, UCC '17 Companion, page 215–220, New York, NY, USA. Association for Computing Machinery.
- Sperlí, G., Amato, F., Mercorio, F., Mezzanzanica, M., Moscato, V., and Picariello, A. (2018). A social media recommender system. *Int. J. Multim. Data Eng. Manag.*, 9:36–50.
- Vigliato, M., Terra, R., Rocha, H., Valente, M. T., and Figueiredo, E. (2018). Microservices in practice: A survey study. *CoRR*, abs/1808.04836.
- Xia, B. and Gong, P. (2014). Review of business intelligence through data analysis. *Benchmarking: An International Journal*, 21:300–311.
- Ölvecký, M. and Host'ovecký, M. (2021). Digital image forensics using exif data of digital evidence. In *2021 19th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 282–286.