CONTENT 2023

The Fifteenth International Conference on Creative Content Technologies

ISBN: 978-1-68558-048-3

June 26 - 30, 2023

Nice, France

CONTENT 2023 Editors

Samuel Kosolapov, Braude Academic College of Engineering, Israel

CONTENT 2023

Forward

The Fifteenth International Conference on Creative Content Technologies (CONTENT 2023), held on June 26 - 30, 2023, continued a series of events targeting advanced concepts, solutions and applications in producing, transmitting and managing various forms of content and their combination. Multi-cast and uni-cast content distribution, content localization, on-demand or following customer profiles are common challenges for content producers and distributors. Special processing challenges occur when dealing with social, graphic content, animation, speech, voice, image, audio, data, or image contents. Advanced producing and managing mechanisms and methodologies are now embedded in current and soon-to-be solutions.

Similar to the previous edition, this event attracted excellent contributions and active participation from all over the world. We were very pleased to receive top quality contributions.

We take here the opportunity to warmly thank all the members of the CONTENT 2023 technical program committee, as well as the numerous reviewers. The creation of a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to CONTENT 2023. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the CONTENT 2023 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope CONTENT 2023 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of creative content technologies. We also hope that Nice provided a pleasant environment during the conference and everyone saved some time to enjoy this beautiful city.

CONTENT 2023 General Chair

Hans-Werner Sehring, Tallence AG, Hamburg, Germany

CONTENT 2023 Steering Committee

Raouf Hamzaoui, De Montfort University - Leicester, UK
Manfred Meyer, Westfälische Hochschule, Germany
Mu-Chun Su, National Central University, Taiwan

CONTENT 2023 Publicity Chair

José Miguel Jiménez, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

CONTENT 2023

Committee

CONTENT 2023 General Chair

Hans-Werner Sehring, Tallence AG, Hamburg, Germany

CONTENT 2023 Steering Committee

Raouf Hamzaoui, De Montfort University - Leicester, UK
Manfred Meyer, Westfälische Hochschule, Germany
Mu-Chun Su, National Central University, Taiwan

CONTENT 2023 Publicity Chair

José Miguel Jiménez, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

CONTENT 2023 Technical Program Committee

Kambiz Badie, Research Institute for ICT & University of Tehran, Iran
René Berndt, Fraunhofer Austria Research GmbH, Austria
Mario Bisson, Politecnico di Milano, Italy
Christos J. Bouras, University of Patras, Greece
Vincent Charvillat, Toulouse University, France
Juan Manuel Corchado Rodríguez, Universidad de Salamanca, Spain
Rafael del Vado Vírseda, Universidad Complutense de Madrid, Spain
Sotiris Diplaris, Information Technologies Institute - Centre for Research and Technology Hellas, Greece
Jimmy Eadie, Trinity College Dublin, Ireland
Hannes Fassold, Joanneum Research - Digital, Graz, Austria
Joshua A. Fisher, Columbia College Chicago, USA
Valérie Gouet-Brunet, IGN / LaSTIG, France
Raouf Hamzaoui, De Montfort University, UK
Tzung-Pei Hong, National University of Kaohsiung, Taiwan
Verena Kantere, National Technical University of Athens, Greece / University of Ottawa, Canada
Wen-Hsing Lai, National Kaohsiung University of Science and Technology, Taiwan
Alain Lioret, Université Paris 8, France
Yong Liu, Universiti Brunei Darussalam, Brunei
Nadia Magnenat-Thalmann, University of Geneva, Switzerland & Nanyang Technological University, Singapore
Prabhat Mahanti, University of New Brunswick, Canada
Maryam Tayefeh Mahmoudi, ICT Research Institute, Iran
Manfred Meyer, Westphalian University of Applied Sciences, Bocholt, Germany

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Color Manipulation in Images by Using a Modified HSV Algorithm

Samuel Kosolapov
Department of Electronics
Braude Academic College of Engineering
Karmiel, Israel
e-mail: ksamuel@braude.ac.il

Abstract— **There are several situations in which the HSV (Hue, Saturation, Value) algorithm is preferred to execute color manipulations in a specific image. Unfortunately, whereas Value and Saturation can be calculated in any situation, Hue cannot be calculated when the specific pixel is gray. Even more, for dark regions, nearly gray regions, and for overbleached regions, the calculation of the Hue is non-reliable. Inherent to digital image noise makes the calculations of Hue in the above situations even more problematic. In an attempt to provide more control during color manipulations, an extended structure "sHSVF" was defined, in which F is a "Validity Flag". Additionally, the structure "sValidityParameters" was defined. Values of the fields in this structure make it possible to classify the specific pixel as "IS_VALID", "HAS_ZERO", "IS_TOO_GRAY", IS_OVERBLEACHED", "IS_TOO_DARK", "IS_TOO_LIGHT", and properly set the "Validity Flag" for each pixel of the image. This flag may be instrumental in color recognition and in reliably modifying the color of the pixels in accordance with the selected rules. By selecting values of the "sValidityParameters", the user of the algorithm can specify situations when the color of the specific pixel is set to the pre-defined value, marking problematic situations. Provided examples demonstrate that this approach can be used for reliable color recognition and advanced color manipulations for synthetic and real-life images.**.

Keywords-Image Processing; HSV; color manipulations; reliable color recognition.

I. INTRODUCTION

The standard inexpensive color digital camera on its output produces a sequence of bytes. In order to apply to this sequence basic imaging processing algorithms, this sequence is organized as a two-dimensional matrix of picture elements (pixels). Each pixel is a vector in the {R (red), G (green), B (blue)} space. In the inexpensive cameras values of color components are in the range {0..255}. Presentation of the color as an {R, G, B} vector is quite natural for a human observer, having three types of color receptors. However, for applications used in machine vision, this presentation is not always convenient. An alternative presentation uses {H (hue), S (saturation) V (value)} space, or {H, S, L (lightness)} space. Presentations of the pixels in the RGB, HSV, and HSL spaces are described in a number of classical image processing books [1]-[3]. Functions converting pixels in the RGB space to HSV and HSL spaces and back are well known. The value of H actually describes the color of the pixel and thus can be used to recognize the color of the pixel in a simple and convenient way for machine vision.

There is a number of alternative approaches – for example, a sophisticated approach based on a sequence of different image processing algorithms designed for the specific goal [4]. However, algorithms of that type are in most cases too heavy for real-life applications.

Unfortunately, plain and simple HSV and HSL algorithms have an inherent problem: Hue cannot be calculated if R, G, and B values are equal. When the presentation of pixel values in the range of byte {0..255] is used, classic presentation HSV and HSL became problematic. One solution is to use the zero value of S (saturation) as a marker, pointing out that the value of H cannot be calculated in that case. However, practically when an image has a noise (and images of digital cameras always have significant noise), the situation becomes even more problematic. It is clear that pixel {100,100,100} is gray, and in this situation, Hue cannot be calculated. But if a digital camera produces a noise of, say 5 units, then pixel {98, 101, 104} must be treated as problematic for the reliable Hue calculation. Despite the fact that the value of Hue can be calculated in that case, it is clear that using this value for color recognition may lead to unreliable results.

Detailed analysis of the properties of digital cameras reveals a number of additional problematic situations. For the synthetic image (an image created by software), pixel {100, 0, 0} is a legal description of a RED pixel. But for the digital camera having noise 5 units this zero value is electronically problematic. The same is valid for the value equal to 255 – this value, in most cases, means that the object is overbleached. Those and some other situations make the classic HSV/HSL approach at least problematic for real-life applications.

The upper image in Figure 1 demonstrates a real-life photo of the mandarins. Vertical and horizontal orange arrows point to the marker (cross) in the clearly overbleached region of this selected mandarin. The lower image presents the profile of the row of the marker in the upper image. The blue arrow point to the values that arrived at the maximal value of 255. Despite the human observer probably recognizing those pixels as "pixels of orange", machine vision must reject Hue calculations of those pixels as non-reliable. The green arrow is pointing to the regions in which the blue value is very low and even arrives at the minimal zero value. In the upper image, those pixels are green leaves in the shadow. In this situation, the color of those pixels cannot be reliably evaluated as by a human

 ISBN: 978-1-68558-048-3

observer, as by hue calculation by using a computer algorithm.

Earlier attempts to improve HSV/HSL algorithm were described in [5] and [6]. This article described a more elaborated approach in an attempt to provide a more reliable algorithm of color recognition and color manipulation based on a modification of the classic HSV algorithm.

To cope with the above problems, in an attempt to provide more control during color recognition and color manipulations, an extended structure “sHSVF” was defined, in which F is a "Validity Flag".

Additionally, the structure "sValidityParameters" was defined. Values of the fields in this structure make it possible to classify the specific pixel as "IS_VALID", “HAS_ZERO”, "IS_TOO_GRAY", IS_OVERBLEACHED", "IS_TOO_DARK", "IS_TOO_LIGHT", and properly set the "Validity Flag" for each pixel of the image. By selecting values of the “sValidityParameters”, the user of the modified HSV algorithm can specify situations when the color of the specific pixel is set to the pre-defined value, marking problematic situations.

Section II describes the definition of sHSVF structure (subsection ‘A’), sValidityParameter (subsection ‘B’), and flags that are used in specific situations (subsection ‘C’).

Section III presents changes in the classical HSV algorithm.

Section IV presents exemplary analyses and processing of synthetic and real-life images demonstrating the properties of a modified algorithm.

Section V shortly summarizes the results obtained.

II. STRUCTURES SHSVF, SVALIDITYPARAMETER AND FLAGS

To store {R, G, B} values of the pixel, standard sRGB structure was used without changes:

```
struct sRGB
{
        unsigned char r;
        unsigned char g;
        unsigned char b;
};
```

Standard sHSV structure was modified by using the “double” type and by adding the integer “ValifityFlag”.

A. *Structure sHSVF*

The resulting sHSV structure was defined as:

```
struct sHSVf
{
        double H;   // Hue
        double S;   // Saturation
        double V;   // Value
        int    validityFlag; // Validity flag
}; .
```

B. *Structure sValidityParameter*

Structure sValidityParameter was designed to set numerical values needed to mark problematic pixels. It was defined as:

```
struct sValidityParameter
{
        double rgbMeanVmin;
        double rgbMeanMax;

        double SaturationMin;
};
```

Usage of this structure will be described later.

C. *Definitions of FLAGS*

In order to properly mark problematic situations, the following FLAGS were defined:

```
#define IS_VALID (0)
```

This flag is set when the value of HUE can be calculated. In this situation value in the HUE map and in the Saturation map is gray in the range {0..255}.

```
#define HAS_ZERO (1)
```

This flag was set when at least one of the {R, G, B} values were zero. This situation is electronically problematic, hence those pixels on the HUE and Saturation maps were marked by a MAGENTA color.

```
#define IS_TOO_DARK     (2)
```

This flag was set if the mean value calculated as (R+G+B)/3 was lower than the value set in the rgbMeanMin parameter of the structure sValidityParameter. Those pixels in the HUE and Saturation maps were marked by a RED color.

```
#define IS_TOO_LIGHT    (3)
```

This flag was set if the mean value calculated as (R+G+B)/3 was higher than the value set in the rgbMeanMax parameter of the structure sValidityParameter. Those pixels on the HUE and Saturation maps were marked by a YELLOW color.

```
#define IS_OVERBLEACHED    (4)
```

This flag was set if at least one value in {R,G,B} is 255. Those pixels on the HUE and Saturation maps were marked by a GREEN color.

```
#define IS_TOO_GRAY     (5)
```

This flag was set if the calculated saturation value was lower than the value of SaturationMin in the structure sValidityParameterat. Those pixels on the HUE and Saturation maps were marked by a BLUE color.

Again, if none of the above flags was set, sValidityParameter is set to the IS_VALID value defined as zero. Then pixels in the Hue and Saturation maps are gray pixels, whereas the level of gray mapping Hue and Saturation values to the range of [0..255}. It must be noted that historically, in the Windows OS, values of Hue and Saturation were mapped in the {0..239} range. Some authors mapped values of Hue in the {0..360} range, however, this range cannot be presented in the standard displays designed for humans. Hence, the range [0..255} is better suited the goal of this research.

 ISBN: 978-1-68558-048-3

III. CHANGES IN THE CLASSICAL HSV ALGORITHM

Classical function RGBtoHSV which is described at [1]-[3], and C-code of which is available in the public domain was modified by adding flags defined before. A number of exemplary code fragments of the reworked function ConvertRGBtoHSVf are presented in Figure 2. This function is defined as:

```
void ConvertRGBtoHSVf(
        sRGB rgb,  sHSVf & hsvf,
        sValidityParameter param,
        int useLimits);
```

Arguments of the function are "rgb" values of the current pixel as defined in the sRGB structure; calculated values of the Hue, Saturation, Value, and Validity flag of the above pixel as defined in the sHSVF structure; "param" specifying parameters used as limits for the processing of this pixel; and flag useLimits, which can be set to FALSE or TRUE. When this flag is set to FALSE, this values of Hue and Saturation are calculated in the "classical way". The value of the "V" can be calculated in any situation. The code and comments of the code fragments presented in Figure 2 are self-explanatory.

Obviously, the reverse function ConvertHSVfToRGB is defined as:

```
void ConvertHSVfToRGB(
    sHSVf hsvf,
    sRGB& rgb);
```

Additionally, were defined functions converting source image to the Hue, Saturation, and Value maps, and to the image presenting validityFlag values as a human-readable gray map, in which different values are encoded by using different levels of gray.

To demonstrate this approach to the well-known procedure of "recoloring", the exemplary function ChangeHue was defined as:

```
void ChangeHue(
        unsigned char trueColorSource[][..][..],
        unsigned char trueColorDestination[][..][..],
        double oldHue,  double newHue,
        double hueHalfRange,
        sValidityParameter param, int useLimits);
```

It must be noted that most recoloring algorithms replace the specified value of Hue with a new one. But this approach is adequate only for synthetic images. Hence, in this function, the additional parameter "hueHalfRange" is added. Then, all pixels having valid values of Hue in the range from (oldHue – hueHalfRange) to the (oldHue + hueHalfRange) will be replaced with the "newHue" value. Naturally, this function uses validity parameters to exclude from the processing problematic pixels. By setting values of those parameters different image processing and color manipulation effects can be achieved.

IV. EXAMPLES OF ANALYSIS AND PROCESSING OF SYNTHETIC AND REAL-LIFE IMAGES

The upper left image in Figure 3 presents a synthetic image specially prepared for the tests of a modified approach and its HSVF maps. The upper part of this image is strips having different gray levels and 6 basic colors Pseudo random noise with a magnitude of 5 units was added. The effect of this noise is clearly seen in the top right map of the Validity flag. The central part of the synthetic image is a pure green ramp, whereas the lower part of this image is a sum of a gray ramp with a green ramp. The lower left image in Figure 3 represents the Hue map, whereas the lower right image represents a Saturation map. Detailed analysis of images presented in Figure 3 validated that the HSVF algorithm works as expected.

Figure 4 represents an example of color manipulation with a real-life photo. Yellow Lemons in the left image were marked in the right image by using non-natural Blue color. In this image, only one lemon was not properly illuminated by the Sun, so all lemons (except the darker part of one lemon) were successfully recognized and recolored.

The left top image in Figure 5 presents somehow problematic real-life photo. In this photo, some mandarins and leaves are in the deep shadow, whereas some mandarins are clearly overbleached. Parameters of the algorithms were selected in such a way, that only reliably validated parts of the mandarins will be recolored to the blue color (see top right image). The lower left image presents a fragment of the Hue map, whereas the lower right image represents a fragment of the Saturation map of the above photo. Problematic pixels are marked as green (overbleached) and red (too dark).

V. SUMMARY AND CONCLUSIONS

An exemplary recoloring algorithm can be fine-tuned by setting relevant parameters and flags. Examples presented in section IV demonstrated that the described approach could be used to analyze and process real-life photos. It is planned to add more parameters and flags in future research.

REFERENCES

[1] J. Russ, and F.Neal, "*The Image Processing Handbook*", CRC Press, 2017.

[2] R. Jain, R. Kasturi, and D. Schunck, "*Machine Vision*", MIT Press and McGraw Hill, Inc. 1995.

[3] J. Foley, A. van Dam, S. Feiner, and J. Hughes , "*Computer Graphics Principles and Practice*". Second Edition in C, ADDISON-WESLEY, 1997.

[4] J. Siyi and C. Heng, "*Color Recognition Algorithm Based on Color Mapping Knowledge for wooden Building Image", Scientific Programming*, Volume 2022, pp 1-15, 2022.

[5] S. Kosolapov, "*Comparison of Robust Color Recognition Algorithms*", Journal of International Scientific Publications, Materials, Methods and Technologies, Volume 15, pp 274 – 283, 2021.

[6] S. Kosolapov, "*Evaluation of Robust Color Recognition Algorithms*", Journal of International Scientific Publications, Materials, Methods & Technologies, Volume 16, pp 83-93, 2022.

 ISBN: 978-1-68558-048-3

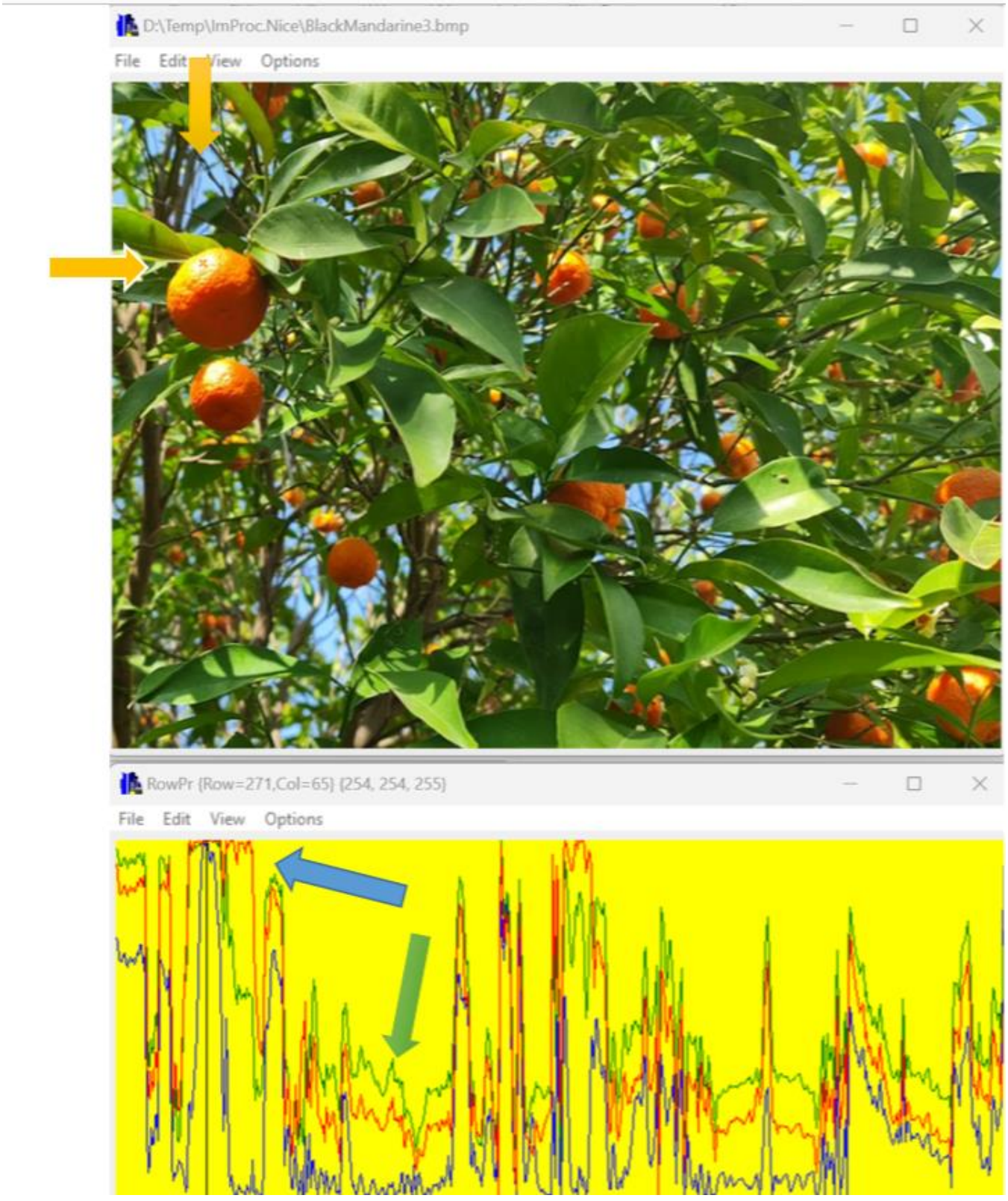

Figure 1. Real-life image of mandarines and profile of the row marked by a cross and by orange arrows. Blue arrow points to the overbleached region. Gren arrow points to the unrealistically low blue values.

 ISBN: 978-1-68558-048-3

```
if (useLimits == TRUE)
{
    if ((rgb.r == 255) || (rgb.g == 255) || (rgb.b == 255))
    {
        hsvf.validityFlag = IS_OVERBLEACHED;
        hsvf.V = 255;//  Can always be set
        hsvf.H = 0;  //  Non reliable: flag is set
        hsvf.S = 0;  //  Non reliable: flag is set
        return;
    }
```

```
double mean = 1.0 * (rgb.r + rgb.g + rgb.b) / 3.0;

if (mean < param.rgbMeanMin)
{
    hsvf.validityFlag = IS_TOO_DARK;
    hsvf.V = 0;  //  Can always be set
    hsvf.H = 0;  //  Non reliable: flag is set
    hsvf.S = 0;  //  Non reliable: flag is set
    return;
}
```

```
if (mean > param.rgbMeanMax)
{
    hsvf.validityFlag = IS_TOO_LIGHT;
    hsvf.V = 255;//  Can always be set
    hsvf.H = 0;  //  Non reliable: flag is set
    hsvf.S = 0;  //  Non reliable: flag is set
    return;
}
```

```
double delta = max - min;

if (delta == 0)  // Pixel is Gray
{
    hsvf.validityFlag = IS_TOO_GRAY;
    //  V was already calculated
    hsvf.S = 0;  //  Saturation is 0
    hsvf.H = 0;  // Hue cannot be calculated, flag is set
    return;
}
```

```
if (useLimits == TRUE)
{
    if (hsvf.S < param.SaturationMin)
    {
        //  V was already calculated
        hsvf.S = 0; // S calculation is not reliable, hence is set to 0
        hsvf.H = 0; // Hue calculation is not reliable, hence is set to 0
        hsvf.validityFlag = IS_TOO_GRAY;
        return;
    }
}
```

Figure 2. Extracts of the code from the file HSV.cpp demonstrating some situations when validity flags are set to the relevant values.

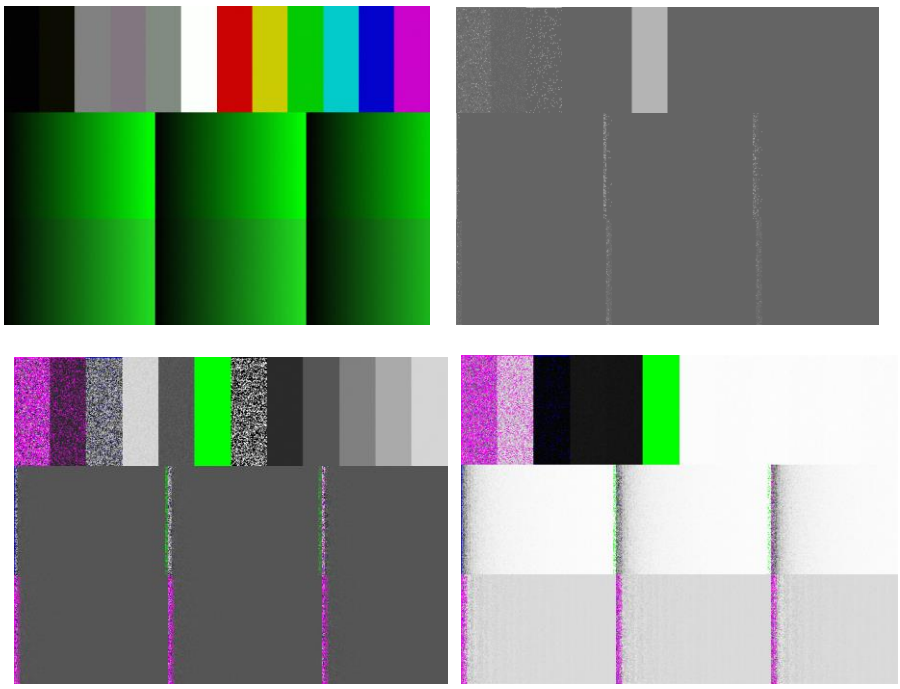

Figure 3. Upper left: Synthetic image with simulated noise. Upper right: Map of validity flags. Lower left: Hue Map. Lower right: Saturation map.

Figure 4. Left: Real-life image of lemons. Right: Example of color manipulation: Objects having Hue in the selected range of values are recolored to the blue color.

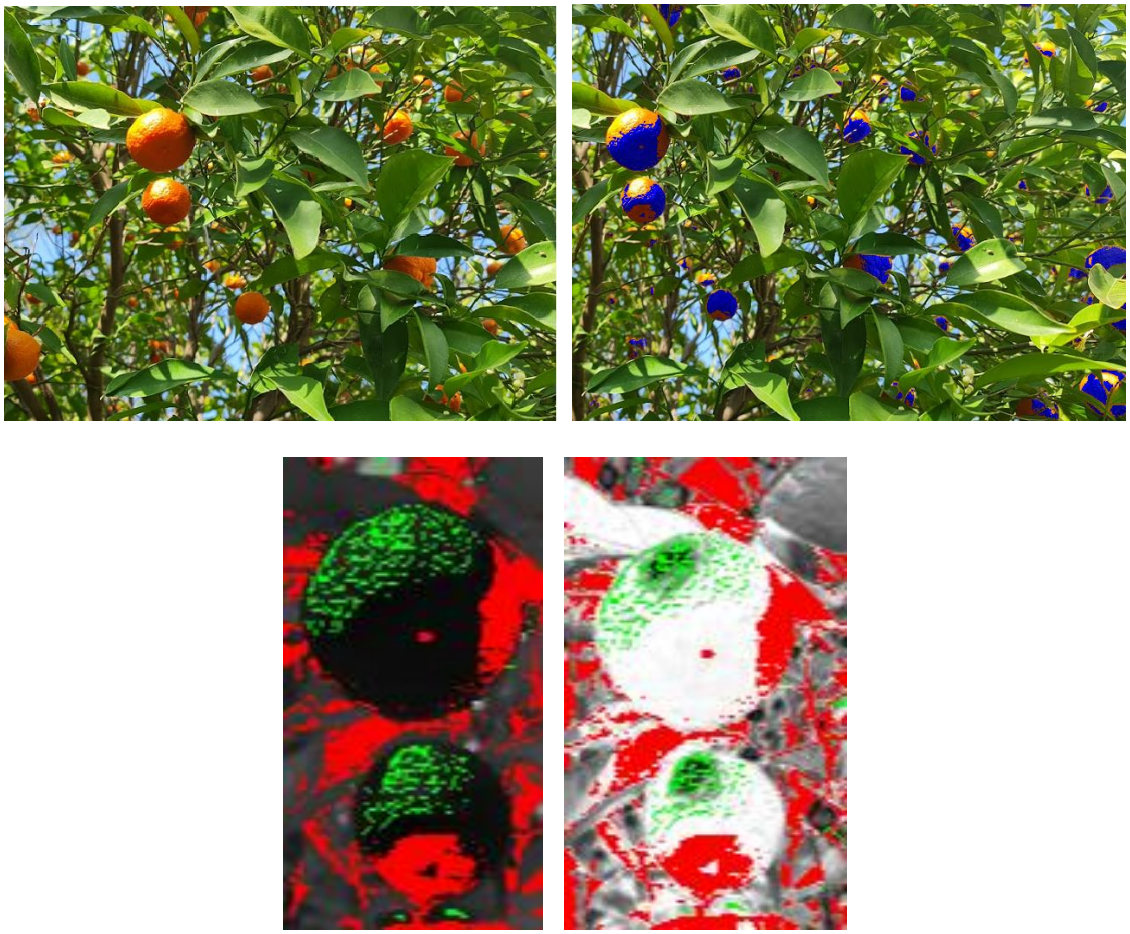

Figure 5. Top Left: Real-life image of mandarins. Top Right: Right: Example of color manipulation: Objects having reliably calculated Hue values in the selected region of the Hue values are recolored to the blue color. Bottom left: Extract from the Hue Map. Bottom right: Extract from the Saturation map. Problematic regions are marked as green (overbleached) and red (too dark)

 ISBN: 978-1-68558-048-3

Using Text Queries to Look Up Unlabeled Images: A Command-Line Search Tool Based on CLIP

Yurij Mikhalevich
Lightning AI
Dubai, United Arab Emirates
email: yurij@mikhalevi.ch

***Abstract*—This paper presents a practical, scalable implementation of an image search engine using OpenAI's Contrastive Language-Image Pre-Training (CLIP) model. The method provides a convenient Command-Line Interface (CLI) and introduces a cache layer powered by SQLite 3 relational database management system (RDBMS) that facilitates efficient repetitive image searches within extensive image databases using natural language queries. The method's effectiveness was evaluated on ImageNet-1k and CIFAR-100 datasets, yielding a 31.17% top-1 accuracy on the ImageNet-1k train set and 55.15% top-1 accuracy on the CIFAR-100 test set. The scalability study showed that indexing time scales linearly with the number of images, and image search time increases only slightly; for example, on an Apple M1 Max CPU, indexing over a million images took 26.36 times more than indexing 50,000 images, while querying the larger image set took just 2.75 times longer. This approach is particularly relevant for industries managing vast volumes of visual data, such as media and entertainment, security, and healthcare.**

***Keywords*—image search, image indexing, photo management, computer vision, natural language processing**

I. Introduction

The release of the Contrastive Language-Image Pre-Training (CLIP) model by OpenAI in 2021 has generated significant attention in the field of Natural Language Processing (NLP) and Computer Vision (CV). This model has demonstrated the ability to learn state-of-the-art image representations from a massive dataset of 400 million (image, text) pairs that were collected from the Internet. CLIP can predict the most relevant text snippet, given an image, using natural language instructions without explicitly optimizing for the task. This zero-shot ability is similar to that of GPT-2 and 3 [1]. The authors of CLIP have demonstrated that CLIP performs as well as the original ResNet50 on ImageNet in a zero-shot manner without using any of the original 1.28M labeled examples. This accomplishment overcomes significant challenges in computer vision.

In addition to these capabilities, CLIP allows researchers to perform image searches using natural language queries. This article explores this particular application of CLIP.

The exponential growth of data, particularly in the form of images, makes this research especially relevant. With the proliferation of digital devices and the Internet, more and more images are being produced and shared online every day. As a result, it is becoming increasingly challenging to find specific images manually. A robust and efficient image search solution can help users and businesses quickly find the images they need amidst this growing sea of data. Moreover, as the volume of photos being shared online continues to increase, it becomes increasingly critical to protect intellectual property and online security. An efficient image search solution can help identify and remove images that violate copyright laws or contain sensitive or inappropriate content. Therefore, with the growing amount of data and images being created, an efficient image search solution is becoming increasingly necessary and relevant.

In Section 2, the paper explores related works in the field of image search. Section 3 describes the proposed method for image search, which is using CLIP. Section 4 presents the implementation details of the proposed method. Section 5 discusses the performance of the proposed method. Section 6 presents the search quality measurement results of the proposed method. Finally, Section 7 concludes the paper and discusses future work.

II. Related Works

There exist multiple methods and solutions for image search.

Traditional image search methods rely on bag-of-features, which are sets of features that describe the contents of an image. These features can be manually specified by annotating the image with descriptive labels. For instance, Fiedler et al. [2] proposed an image tagging software that helps users enter the labels efficiently. Another way to obtain labels is to use labels injected into the photos by camera software, as explored by Tesic [3], who examined camera metadata for consumer photos.

Mobile phone software can also inject useful metadata within photos. Kim et al. [4] explored how this metadata can be leveraged to effectively manage and search photos using mobile smartphones. This metadata can include information about the location, time, or type of camera used, among others. By using such metadata, it's possible to automatically generate labels for images.

One way to automatically generate labels is by using image recognition algorithms, such as Convolutional Neural Networks (CNNs). CNNs can be specifically trained to recognize and classify images based on their contents. For example, Krizhevsky et al. [5] trained a CNN to recognize images based on their visual features. This approach can be combined

 ISBN: 978-1-68558-048-3

with other methods, such as using camera metadata or manual annotations, to generate more accurate and diverse labels for images. Lee et al. [6] proposed a scalable method for image annotation that combines manual and automatic approaches to improve the accuracy and scalability of the labeling process.

In image retrieval, search algorithms are applied to the features extracted from images to find relevant images based on a user's query. The choice of search algorithm depends on the type of labels associated with the images and the problem at hand. For instance, if the labels are GPS coordinates, a distance-based search algorithm can be used to find images related to the query location, as demonstrated by Zhang et al. [7].

On the other hand, if the labels are in text form, a text-based search algorithm can be employed to retrieve images based on textual queries. There are various techniques available for text-based search, ranging from substring matching to lemmatization-based search, as shown by Balakrishnan et al. [8], or even using word embeddings, as explored by Günther [9] and Kenter et al. [10].

In practice, combining multiple features and approaches, such as GPS coordinates, text labels, image capture date, camera model, focal distance, etc. can yield more accurate and relevant results. Ismail [11] investigated image annotation and retrieval based on multi-modal feature clustering and found that this approach can significantly improve the retrieval performance of the system.

Therefore, in summary, image retrieval is a complex process that involves the extraction of features from images, the selection of suitable search algorithms based on the type of labels and the integration of multiple features and approaches to improve retrieval accuracy.

The methods mentioned earlier do not facilitate searching for images using a text query without first identifying the features. Moreover, they do not permit searching for a concept that is not included in the image labels, even if the concept exists within the image. These are the problems that OpenAI's CLIP [1] enables us to solve.

III. Method

With CLIP's text transformer, it is possible to convert a text query to a n-dimensional vector (where n differs depending on the CLIP model used). With CLIP's image transformer, it is possible to convert an image to a n-dimensional vector. Then, we can calculate the dot product (Equation 1) of the normalized query vector and each of the normalized image vectors. After this, we sort the images by the decreasing dot product and take first k images; this gets us the k images that match the query the most.

$$\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \qquad (1)$$

While this approach works reasonably fast on a few images, it may not be scalable when dealing with a large number of images. This is particularly challenging if there is no access to GPUs to run the CLIP model. In order to address this scalability issue, the solution proposed in this paper suggests caching the image vectors. By caching the image vectors, repeated queries can be executed quickly, without having to wait for the image to be processed each time that a query is made.

Caching involves storing the computed image vectors on disk after the initial processing of images so that they can be accessed later. When a new query is received, the system retrieves the cached image vectors and computes only the query vector. This approach reduces the computational overhead associated with image querying and improves the response time for users. This is particularly useful when dealing with large image databases, where repeated queries are common. By leveraging caching, the system can handle repeated queries quickly and efficiently without having to recompute the image vectors each time.

The proposed solution also allows adding new images to the cache to avoid recomputing the whole cache when the image catalog is updated.

The main advantage of the proposed method over the methods described in Section 2 is that method does not require any metadata, labels, or annotations, which enables using it with any image catalog. Moreover, if used on labeled images, the proposed method allows searching for concepts not included in the image labels. The presence of these mechanisms enables users to utilize CLIP effectively in real-world scenarios involving image search.

IV. Implementation

The method described above is implemented in the Python utility called `rclip` [12]. `rclip` provides an easy-to-use CLI interface that allows users to search images within any directory on a computer where `rclip` is installed. Search within nested directories is also supported. To use it, the user should open the terminal, navigate to the directory containing the files that they want to search through, type `rclip <search query>`, and hit "Enter."

The solution uses OpenAI's `clip` library [1] to load the model and compute the feature vectors. `rclip` uses `ViT-B/32` version of the CLIP model. The code of the feature computing methods is shown in Figure 1 and Figure 2. The vectors produced by these methods are then used to compute the dot product (Equation 1).

`rclip` features the image vector cache implemented using `SQLite 3` RDBMS [13]. The image vector is computed and added to the index only if the cache does not already contain an entry for a given image. The vectors are cached by image paths. This allows the user to execute repeated queries over the same image catalog without waiting for the image vectors to be recomputed. Figure 3 defines the structure of the table storing the cached image vectors.

The implementation introduces a cache layer of a design described above, a module to manage (add, remove, and update) entries in the cache, a module to perform image and text vector computations using the CLIP model, and a module

 ISBN: 978-1-68558-048-3

```
def compute_image_features(self, images):
  images_preprocessed = (torch
    .stack([self._preprocess(thumb) for
        thumb in images])
    .to(self._device))

  with torch.no_grad():
    image_features = self._model.
        encode_image(images_preprocessed)
    image_features /= image_features.norm
        (dim=-1, keepdim=True)

  return image_features.cpu().numpy()
```

Figure 1. Image vector computing method

```
def compute_text_features(self, text):
  with torch.no_grad():
    text_encoded = self._model.
        encode_text(clip.tokenize(text).to
        (self._device))
    text_encoded /= text_encoded.norm(dim
        =-1, keepdim=True)

  return text_encoded.cpu().numpy()
```

Figure 2. Text vector computing method

to provide the user with a convenient command-line interface to perform image search.

When the user executes the `rclip` command, the tool first recursively processes the current directory and all its subdirectories, finds all of the images in them, and, if the image is not already present in the cache, computes their feature vectors and saves them to the SQLite cache database. This step is quick on repeated queries because the images are already cached, but if the user wants to skip the process of checking that the cache is up-to-date, they can pass the `-n` argument to `rclip` to skip the indexing step completely and speed up the `rclip` command execution even more.

Then, `rclip` computes the query vector, fetches from the cache database image vectors for all of the images located in the current directory, computes the similarity score between the query vector and each of the image vectors, sorts the scores by the decreasing order, and gets the first k images with the highest similarity scores.

Finally, `rclip` prints the paths to the images that match the query to the terminal. Users can then open the images in their favorite image viewer or editor.

The `rclip` source code is published on GitHub under the MIT license [12].

```
self._con.execute('''
  CREATE TABLE IF NOT EXISTS images (
    id INTEGER PRIMARY KEY,
    deleted BOOLEAN,
    filepath TEXT NOT NULL UNIQUE,
    modified_at DATETIME NOT NULL,
    size INTEGER NOT NULL,
    vector BLOB NOT NULL
  )
''')
```

Figure 3. Structure of the "images" table storing the cached image vectors

V. Performance

`rclip` was benchmarked using two different CLIP models, `ViT-B/32` (smaller CLIP model) and `ViT-L/14@336px` (larger CLIP model), on a NAS running `Intel(R) Celeron(R) CPU J3455 @ 1.50GHz`. Table I shows how `rclip` performs when indexing and searching through 269 photos when running on this CPU.

As Table I shows, the `ViT-L/14@336px` performance will not scale well, which makes `rclip` unusable in practical scenarios when running CLIP on low-level and mid-level consumer CPUs. This is why `rclip` uses `ViT-B/32`.

Running `rclip` indexing with `ViT-B/32` on 72,769 photos on the same NAS powered by `Intel(R) Celeron(R) CPU J3455 @ 1.50GHz` took 23 hours. Performing a query over 72,769 photos takes 56 seconds.

To give a better understanding of how `rclip` performance scales, Table II shows how `rclip` performs when indexing and searching through 50k images and 1.28m images on the `Apple M1 Max` CPU. As the Table II shows, the indexing time scales linearly with the number of images when the search time increases only slightly even when going from searching through 50 thousand images to searching through 1.28 million images.

It should be noted that real-life `rclip` application possibilities go far beyond results demonstrated in these benchmarks

TABLE I
INDEXING AND SEARCH PERFORMANCE ON `Intel(R) Celeron(R) CPU J3455 @ 1.50GHz` USING DIFFERENT CLIP MODELS

Model	Indexing Time	Search Time
`ViT-B/32`	3m56.626s	0m18.064s
`ViT-L/14@336px`	125m0.507s	3m19.742s
Difference	x31.70	x11.06

TABLE II
INDEXING AND SEARCH PERFORMANCE ON `Apple M1 Max` CPU USING `ViT-B/32`

Dataset	# of images	Indexing Time	Search Time
`ImagNet-1k` validation set	50k	19m24.750s	0m4.04s
`ImagNet-1k` train set	1.28m	8h31m26.680s	0m11.49s
Difference	x25.62	x26.35	x2.84

because `rclip` can be applied be used to execute any queries and not just the ones present in the dataset labels.

VI. Search Quality

As Table III shows, `rclip` achieves 31.17% top-1 accuracy and 44.80% top-5 accuracy rate on the `ImageNet-1k` [14] 1.28 million images train set and 55.15% top-1 and 81.34% top-5 accuracy on the `CIFAR-100` [15] 10 thousand images test set.

To get a better understanding of `rclip`'s performance, see Figure 5, Figure 6, and Figure 7 showing search results for a search performed on an unlabeled demo set of 142 images. Figure 4 gives a glimpse into the set and shows that there is a variety of different images present there.

VII. Conclusion

The development of the command-line tool, `rclip`, which employs OpenAI's CLIP model, has resulted in an efficient and user-friendly utility for image search. However, there are still possibilities for further optimizing its performance. Future plans include:

- to create a separate model for the CLIP text transformer and to load only it when users initiate a search that does not require indexing, thereby avoiding the loading of the CLIP vision transformer;
- to push the tool's scaling limits even further by introducing the sharded cache;
- to improve the tool's performance by preventing it from re-indexing files when they are renamed;
- to enable `rclip` write tags to the image file metadata for third-party software to utilize them;
- to enrich `rclip`'s search capabilities by utilizing metadata, which already exists within the images, like GPS coordinates, image capture date, camera model, tags, etc.;
- to do an in-depth comparison of `rclip` with other existing image search tools;
- to explore how well CLIP handles distorted and corrupted images.

Even with its current performance and capabilities, `rclip` is an incredibly valuable tool.

In summary, this paper introduces a practical and scalable method of searching images using natural language queries based on the CLIP model. The approach has demonstrated impressive results on the ImageNet-1k and CIFAR-100 datasets, indicating its potential applicability to a wide range of industries reliant on visual data.

TABLE III
RCLIP SEARCH QUALITY

Model	Top-1 accuracy	Top-5 accuracy
`ImageNet-1k` 1.28m	31.17%	44.80%
`CIFAR-100` 10k	55.15%	81.34%

References

[1] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," arXiv, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020.

[2] N. Fiedler, M. Bestmann, and N. Hendrich, "Imagetagger: An open source online platform for collaborative image labeling," in *RoboCup 2018: Robot World Cup XXII 22*, Springer, 2019, pp. 162–169.

[3] J. Tesic, "Metadata practices for consumer photos," *IEEE MultiMedia*, vol. 12, no. 3, pp. 86–92, 2005.

[4] J. Kim, S. Lee, J.-S. Won, and Y.-S. Moon, "Photo cube: An automatic management and search for photos using mobile smartphones," in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, IEEE, 2011, pp. 1228–1234.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[6] B. N. Lee, W.-Y. Chen, and E. Y. Chang, "A scalable service for photo annotation, sharing, and search," in *Proceedings of the 14th ACM international conference on Multimedia*, 2006, pp. 699–702.

[7] J. Zhang, A. Hallquist, E. Liang, and A. Zakhor, "Location-based image retrieval for urban environments," in *2011 18th IEEE International Conference on Image Processing*, IEEE, 2011, pp. 3677–3680.

[8] V. Balakrishnan and E. Lloyd-Yemoh, "Stemming and lemmatization: A comparison of retrieval performances," *Lecture Notes on Software Engineering*, vol. 2, no. 3, pp. 262–267, 2014.

[9] M. Günther, "Freddy: Fast word embeddings in database systems," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 1817–1819.

[10] T. Kenter and M. De Rijke, "Short text similarity with word embeddings," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 1411–1420.

[11] M. M. B. Ismail, *Image annotation and retrieval based on multi-modal feature clustering and similarity propagation*. University of Louisville, 2011.

[12] Y. Mikhalevich, *rclip*, version 1.2.5, Jan. 2023. [Online]. Available: https://github.com/yurijmikhalevich/rclip.

[13] R. D. Hipp, *SQLite*, version 3.31.1, 2020. [Online]. Available: https://www.sqlite.org/index.html.

[14] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[15] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," University of Toronto, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

 ISBN: 978-1-68558-048-3

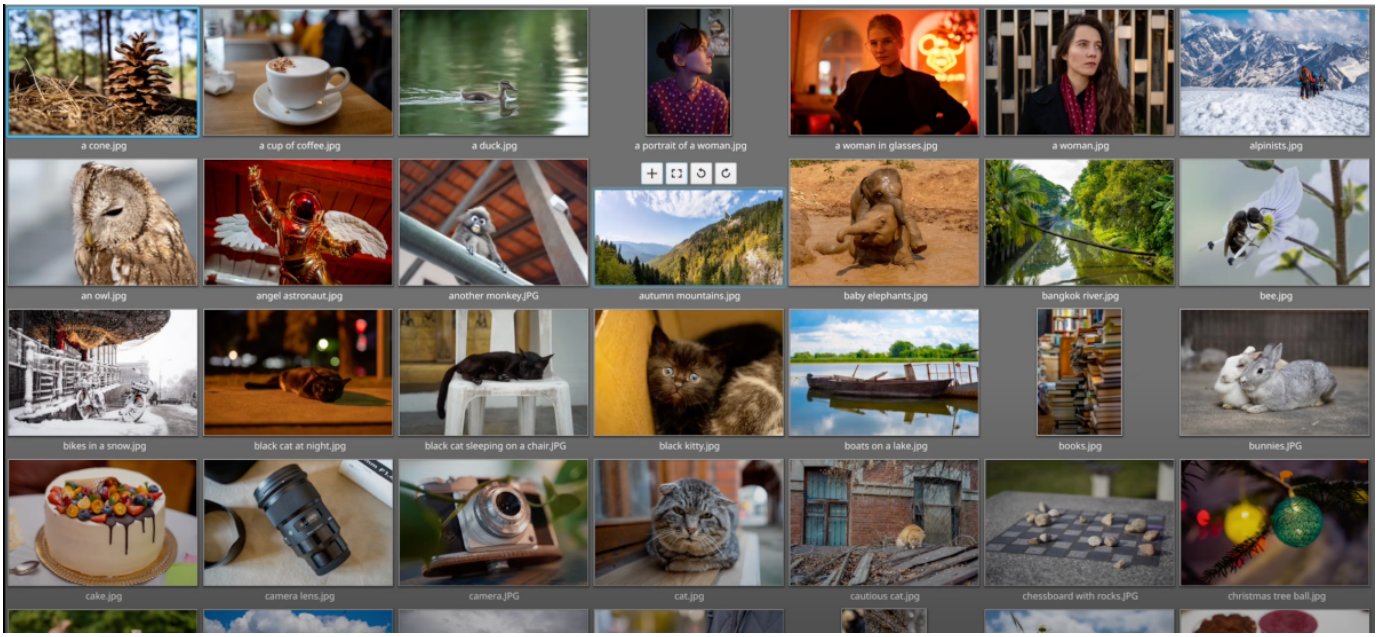

Figure 4. Demo set sample

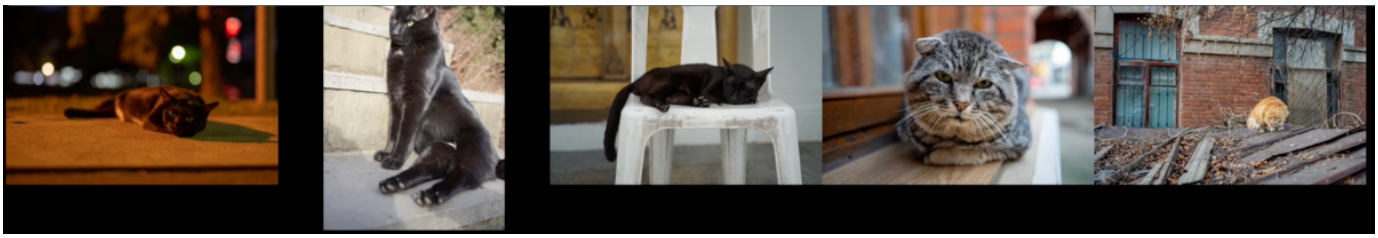

Figure 5. Search result for query “cat”

Figure 6. Search result for query “leading lines”

Figure 7. Top result for query “a kitten peeking from behind a corner”

 ISBN: 978-1-68558-048-3

Motion Reactive Sound Generation System for Immersive CAVE Environment: a Design Perspective

Stefania Palmieri
Design Department
Politecnico di Milano
Milan, Italy
e-mail: stefania.palmieri@polimi.it

Alessandro Ianniello
Design Department
Politecnico di Milano
Milan, Italy
e-mail: alessandro.ianniello@polimi.it

Mario Bisson
Design Department
Politecnico di Milano
Milan, Italy
e-mail: mario.bisson@polimi.it

Giovanni Barone
Design Department
Politecnico di Milano
Milan, Italy
e-mail: giovanni2.barone@mail.polimi.it

***Abstract*— The objective of this paper is to bring attention to the world of immersive technologies and in particular to those technologies that when put in system create the Cave Automatic Virtual Environment (CAVE) system, with the aim of including within the design of these systems, the sound component, and the interactivity connected to them. Different degrees of immersiveness and fruition systems can be identified among mixed reality and immersive reality technologies. Depending on the technologies employed, immersive digital experiences are on a spectrum ranging from Virtual Reality (VR), where the individual is totally immersed in a fictional reality, to Augmented Reality (AR), where the experience is of a hybrid reality characterized by the superimposition of digital and virtual content on the real environment. Within this spectrum, where at one extreme VR can be found, while at the other AR, CAVE systems are in a central position, definable as Augmented Virtuality. Research related to the development of such a system has, to date, privileged the visual and motion capture aspect, i.e., that which causes visual and virtual elements to be responsive to the movement of the individual, making it the main element of interaction. The purpose of the paper is to bring attention to a factor not yet central to the design of CAVE systems, namely that of the sound component, with the goal of making it a new interactive element, on par with the visual component, with the aim of increasing the degree of immersiveness of the experience. The integration of this new interaction makes it possible to create systems capable of generating sound outputs, based on the movements of the human being, and therefore motion reactive, thus relying on the same motion capture technology that underlies the visual component of CAVE, and it is believed that the inclusion of this new component can be used to increase the degree of immersiveness of the experience. the final output of the paper will be to propose design directions and tools useful to the 'integration of the sound reactive component in CAVE, which remains to date a fairly unexplored terrain.**

Keywords-audioreactive; CAVE; immersive experience; sound generation; design.

I. INTRODUCTION

In recent years, to increase the degree of immersiveness of an experience, technologies are employed and integrated that allow the user to go beyond what is tangible and visible in physical reality, adding additional layers of perception. In the world of Mixed Reality (MR), the use of different technologies can cause different types of experiences to develop, which are schematized along two different axes: immersion in the virtual environment, and perception of the real world as displayed in Figure 1.

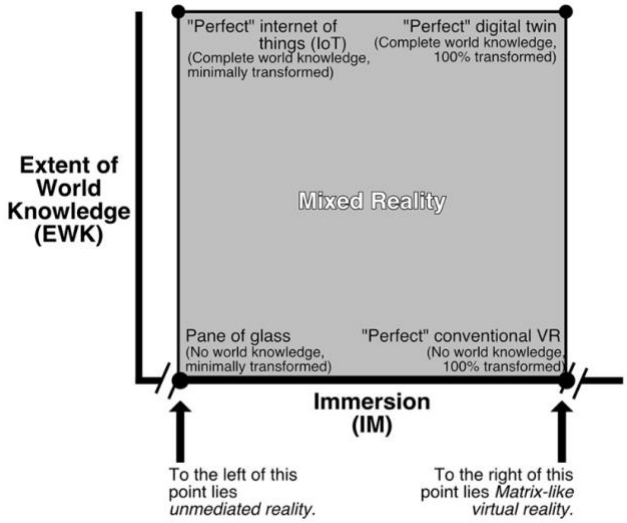

Figure 1. The graph defines Mixed Reality technologies in relation with the technological system's level of immersion, i.e., its capability of making people perceive a sense of presence, and the users's ability to perceive the real world through the technology. (Credits: Skarbez et al., 2021.)

These two dimensions create a broad spectrum of immersive experiences [1].

The different technologies available on the market determine different types of experiences: products such as

HoloLens enable MR based on a high perception of the external environment, which is enriched by virtual elements. Other technologies, such as Oculus, on the other hand, make a highly immersive reality experienceable, but with a minimal level of environmental perception [1].

Also found in the MR spectrum are CAVEs, integrated systems of different technologies that enable the creation of Augmented Virtuality (AV) experiences. These systems provide a high degree of spatial perception and have the potential to achieve an equally high level of immersiveness when implemented through the organic integration of new technologies [2].

The goal of the research is to increase the degree of immersiveness of these systems through the integration of sound generation machines (synths, drum machines, samplers), managed through motion tracking cams, already installed in CAVE systems. By combining these two technologies, it is possible to integrate an additional sensory component into the immersive experience, activating new types of interactions, in order to create mixed realities.

These new interactions can be starting points for thriving experience design scenarios by professionals involved in User Experience (UX) Design and Interaction Design, applicable in the world of entertainment, show business, and music production [3].

Finally, the objective of the paper is to theorize design guidelines for CAVE environments that implement Motion Reactive Sound Generation (MRSG), with the aim, following experimental phases, of creating a design framework that facilitates a systematic and extensive technological implementation of CAVE technology.

From a process of literature review and case studies analysis, information was extracted to generate the research question and consequently the thesis. Section I of the paper makes explicit the methodology used to carry out the research; Section II is devoted to defining the concept of immersiveness and evaluating technologies that can enable immersive experiences. Section III is focused on CAVE systems, highlighting their enabling technologies, structure, functionality, and potential. In section IV the area of audioreactivity is addressed, exploring the ways in which this type of systems can be integrated with CAVE environments; section V focuses on the role of design in the design of multisensory experiences based on the technologies mentioned before; section VI is devoted to the description of the MRSG system and the explication of guidelines for its design. This is followed by section VII, where the outcomes, the future opportunities and the limits of the research are summarized.

II. METHODOLOGY

This contribution is based on a literature review process, conducted on Google Scholar and Web of Science platforms searching through keywords such as audio reactive technologies, multisensory experience, music and video commutation, user experience, and audio-driven generative design. The parameters to select the papers have been the presence of experimental stage, their systemic design approaches and impacts, the exploitation of visual and audio content as fundamental features for the research. It aimed at mapping the state of the art of academic research in the area of interest, highlighting the most promising processes to exploit the goals of the research. In addition to this activity, a case studies analysis was carried out, which is essential to understand the various concepts of mixed reality, and to give an idea of which technologies are currently available and their peculiarities.

To select the case studies, different parameters have been applied: each project should be focused on music experiences, with a particular attention to the use of electronic and Musical Instrument Digital Interface (MIDI) instruments and technologies, and on the relationship between music and visual contents. To evaluate each project, their coherence, information simplicity, scopes, and application of design methodologies have been considered. A further phase of case study analysis was conducted in the area of music generation and sound synthesis and on the ways in which these can be systematized with technologies for immersive experiences, selected applying similar parameters and evaluated following the same criteria. Finally, the content of this paper focuses on the construction of a theoretical framework, which aims to integrate the above technologies, to be tested in a subsequent research phase as shown in Figure 2.

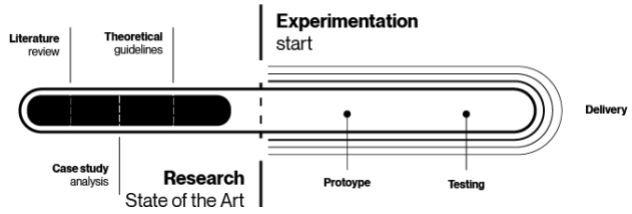

Figure 2. Review of the research process's steps highlighting its state of the art and future activities.

III. IMMERSIVE EXPERIENCE

Immersivity is a phenomenon that can be experienced when an individual is in a state of deep mental involvement [4], [5].

This mental involvement can be fostered by external stimuli, produced by technological systems; therefore, when talking about immersiveness, and especially virtual immersiveness, it is essential to also refer to the technologies that make these realities possible. Some examples of technologies are Microsoft's Hololens [6], or Meta's Oculus. These two technologies are based on AR and VR and are part of what is called Mixed Reality, as stated by Skarbez R, Smith M and Whitton MC (2021). Referring to Milgram and Kishino's [4] continuum on Mixed Reality, the different realities were placed in a diagram, to which a reference technology was assigned, as visible in Figure 3.

 ISBN: 978-1-68558-048-3

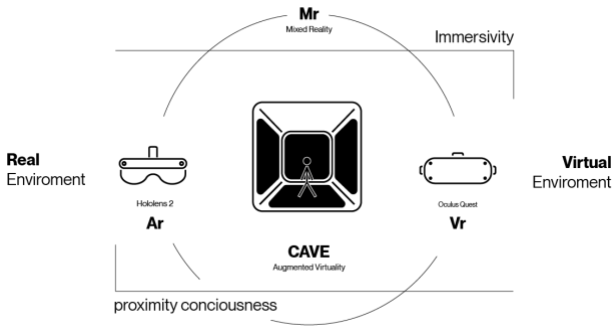

Figure 3. Review of the reality-virtuality continuum by Milgram e Kishino Milgram, P., & Kishino, F. (1994) with display's technologies examples.

As it can be observed in this scheme, the technologies that are part of mixed reality polarize on the right, where devices similar to Oculus can be found, i.e., that take advantage of immersive virtual environments, in which the user enters the virtual world and can interact with it through a control pad [6]. This technology offers a high degree of immersivity since everything can be observed at the moment of use being part of the virtual world, allowing to have a first-person Point of View (PoV) and get into the thick of the action [7].

At the opposite pole, that is, on the left side of the diagram, technologies useful for the reproduction of AR experiences can be highlighted, for example, a smartphone can be a device of AR experiences: just think of the role it plays in the application and display of Instagram filters, the positioning of digital models in space. More complex devices such as HoloLens, on the other hand, allow for greater degrees of immersiveness.

Having the visor as the only touchpoint, the user can interact with digital elements directly with their hands, making the user experience natural and intuitive. The great advantage of these technologies is the high degree of fusion with the real environment, which allows the user to have perception of his or her position in space and consequently to be able to interact with elements and individuals belonging to physical reality.

IV. CAVE SYSTEM

In a central position on the spectrum between the pole of augmented reality and that of virtual reality are CAVE systems, which belong to AV environments [8]; in these systems, the user's experience takes place within walls (the number of which varies from three to four), on which a virtual world is projected, equipped with a motion tracking system that becomes an intermediary between the virtual world and reality [8]. The advantage of these systems is the ability to move freely in space and to be able to interact with the virtual world thanks to these systems, which involve fewer constraints than those associated with wearable devices [8]. Before they were developed, and thus in earlier versions of CAVE technology, mediating devices were in fact used, which allowed interaction with virtual elements.

A CAVE has basic characterizing components and can be described, in essence, as a cube within which the user is placed. The cube shape is used to roughly simulate that of a sphere, so as to replicate the kind of view one has in the real world. The perceptual-visual component of the CAVE is made possible thanks to projectors, which can be placed either inside or outside the structure, (outside in case the walls are transparent, and the projectors point at them) thanks to which the faces of the cube are transformed into screens that envelop the viewer, allowing him to perceive himself immersed in the virtual world [9].

The interactive component, on the other hand, is made possible, as previously stated, by products, such as Microsoft's Kinect, which make motion tracking possible. These systems, placed in dialogue with a computer, make it possible to track the position of the individual in space and capture his or her movements with a level of sophistication that allows discernment of the movement of small portions of the body, such as head or even, in the most advanced devices, eyes (eye-tracking) [10]. All of this information, is subsequently translated into data that the computer uses to enable authentic interaction between the user and the virtual world, creating, for example, dynamic and responsive scenarios influenced by the user's movements.

V. AUDIOREACTIVITY

One component that often takes a back seat to the visual compartment is the world of audio. There are many articles that discuss the benefits that the sound dimension can bring within immersive systems, such as Adaptive Music Composition for Games or music mandala mindfulness [11], [12] in which they explain how the reactive audio component can increase immersiveness in virtual video game worlds in no small part.

Other articles such as Designing generative sound for responsive 3D digital environment interaction [12], on the other hand, talk about the kind of interaction that generative audio can offer within immersive 3D spaces, and the importance of the user having awareness of the spatial dimension around him or her and how generative audio, which responds in real time to movements, can be remarkably immersive [13].

By audioreactive it is mostly meant the whole world that has to do with visual elements that are generated or changed to the rhythm of music [14]. More specifically, thanks to various software, such as for example Touchdesigner, which translates sound into digital inputs, which are processed to go into the mathematical parameters with which shapes, colours, etc. are described [15], [16].

This new form of art is becoming more and more established in the world of entertainment and, in particular, in the field of live music, but it is rooted in the history of Nourathar music by Mary Hallock-Greenewalt: it evolves primarily with the goal of creating synaesthetic musical experiences, which has been pushed to the limit in experiences dedicated exclusively to this world, in which the visuals are not background of the music, but are mixed becoming one work. Examples of such installations may be the works of Jon Weinel [17] or the work of the Berlin-based artist couple Quadrature.

 ISBN: 978-1-68558-048-3

VI. MRSG SYSTEM

Starting from audioreactive systems and their operative functions, through this research the aim is to hypothesize a change in the process of application of such technologies: if in the field of VJing music is an input to create visualizations, through the hypothesized system the scope is to create music using visual sources, and to be precise cameras that integrates motion capture technology to control sound generating machines [18] through the MIDI communication system. Taking inspiration from research such as EyesWeb: Toward Gesture and Affect Recognition in Interactive Dance and Music Systems [19] it is intended to integrate this system within a CAVE equipped with three motion capture sensors, crossed so that a mix of synaesthetic interactions can be created in a sound mixed reality situation, so that the user can control and interact, through his or her own movement [20], [21], with both the visual part of the virtual world and the sound part, thanks to virtual musical instruments that will produce generative music.

An initial hypothesis, which is shown in Figure 4, envisioned that the person inside the cave could actually with his or her movement make the synthesizer play by producing notes with body movement, but in a multi-user scenario the risk of creating sound confusion dictated the need to investigate new possible solutions.

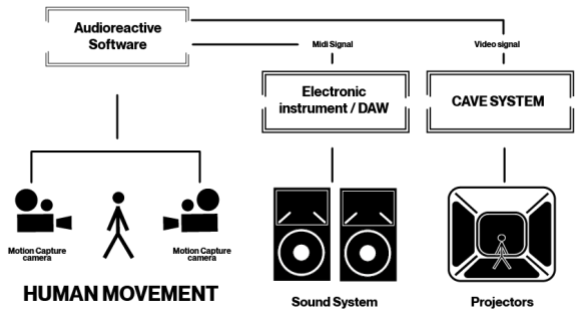

Figure 4. The scheme highlights the routing of the signals starting from the human movements, which become input for the audioreactive software. The software elaborates the input, dividing it into two different kinds of signals (MIDI and video), which finally are elaborated by the electronic instrument and the CAVE system.

Therefore, it was thought to provide a more guided and constrained experience, allowing users to manage simple parameters such as opening or closing filters, sound waveform and rhythm. The experience can be structured in two ways: using a motion sensing system given by the CAVE's cameras, with which to manage the different parameters of the virtual musical instrument, which can be assigned to each moving part of the body: limbs and head a parameter such as Low-Frequency Oscillators (LFO) speed, filter cut-off and sequencer speed, or even tuning the note itself; or, dividing the CAVE space into responsive zones dedicated to each of the instruments. In this case, as moving within the space, the user can give more or less prominence to the instrument is stepping on, while moving the hands on the Z axis will allow to modulate parameters and rhythms of melody playback. Through experimentation in the field, the interaction modes of this immersive, multisensory model will be defined in detail. The experimentation that is already being structured includes an initial phase of technical realization of the software that will act as a bridge between Mixed Reality and a Digital Audio Workstation (DAW), thus transforming the interactive system between CAVE and a human being into a true immersive MIDI controller.

VII. DISCUSSION

The contribution proposed a critical analysis and mapping of the state of the art of research in the field of audioreactive applied to immersive AV contexts, such as CAVE, with the ultimate goal of defining an implementation model replicable in other contexts with the given pre-requisites. To reach this aim a process of literature and of case study analysis were exploited.

From a design perspective, these types of immersive environments can constitute a tool that takes on several functions: on the one hand, they are spaces that have the potential to increase communication and dissemination capabilities, enabling new forms of interactive dialogue; they are functional for study and experimentation phases during the development stages of a project, facilitating the immersive visualization of certain formal elements; and finally, they allow for the application of a co-design approach to design activities, enabling the presence of multiple people interacting within the immersive environment.

Through the integration of MRSG's technologies, these potentials can be expanded to all those areas where the sound component constitutes a design element of central importance.

The research presented here involves the development of successive phases aimed at testing different configurations and applications of the MRSG system with the aim of understanding its technical limitations and systemic peculiarities; in the subsequent prototyping phase the communication between the motion capture system and the sound generation machines will have to be tested through the creation of the prototype of a dedicated platform. At this stage it will be necessary to define the interaction between the human being and the system, which will have to be tested during a UX testing phase, culminating in the delivery of a questionnaire and targeted interviews. The last step will be devoted to formalizing a systemic technology integration model, which will make replication possible.

Following this process, it will be possible to take the hypothesized and tested model to various scenarios, going to replicate the hardware and software system, contextualizing it in the target environment and according to specific requirements.

The application of these technologies and systems to immersive environments can enhance the level of multisensory and cognitive immersion that a user should be able to experience, even though at the time of paper's writing it seems that those kinds of application are still under researched and underdeveloped, due to the complexity of the needed systems and their integration within the same space.

 ISBN: 978-1-68558-048-3

For sure, the main issue and challenge is going to be encountered during the development stage of the project, when the different hardware should be linked to each other through to exploitation of ad-hoc software.

REFERENCES

[1] R. Skarbez, M. Smith, and M.C. Whitton, "Revisiting Milgram and Kishino's reality-virtuality continuum", Frontiers in Virtual Reality, vol. 2, pp. 1-8, 2021, doi: https://doi.org/10.3389/frvir.2021.647997

[2] H. Regenbrecht, T. Lum, P. Kohler, C. Ott, M. Wagner, W. Wilke, and E. Mueller, "Using augmented virtuality for remote collaboration", in Presence, vol. 13(3), pp. 338-354, 2004.

[3] C. Flavián, S. Ibáñez-Sánchez, and C. Orús, "The impact of Virtual, Augmented and Mixed Reality technologies on the customer experience", in Journal of business research, vol. 100, pp. 547-560, 2019, doi: https://doi.org/10.1016/j.jbusres.2018.10.050

[4] P. Milgram and F. Kishino, "A taxonomy of Mixed Reality visual displays", Proc. IEICE TRANSACTIONS on Information and Systems, 1994, pp. 1321-1329.

[5] S. Agrewal, A.M.D. Simon, S. Bech, K.B. Bærentsen, and S. Forchammer, "Defining immersion: literature review and implications for research on audiovisual experiences", in Journal of the Audio Engineering Society, vol. 68(6), pp. 404-417, 2020, doi: https://doi.org/10.17743/jaes.2020.0039

[6] S. Park, S. Bokijonov, and Y. Choi, "Review of Microsoft Hololens applications over the past five years", in Applied sciences, vol. 11(16), pp. 1-26, 2021, doi: https://doi.org/10.3390/app11167259

[7] K. Kilteni, I. Bergstrom, I., and M. Slater, "Drumming in immersive Virtual Reality: the body shapes the way we play", in IEEE transactions on visualization and computer graphics, vol. 19(4), pp. 597-605, 2013, doi: https://doi.org/ 10.1109/TVCG.2013.29

[8] M.A. Muhanna, "Virtual reality and the CAVE: Taxonomy, interaction challenges and research directions", in Journal of King Saud University-Computer and Information Sciences, vol. 27(3), pp. 344-361, 2015, doi: https://doi.org/10.1016/j.jksuci.2014.03.023

[9] D.A. Guttentag, "Virtual Reality: Applications and implications for tourism", in Tourism management, vol. 31(5), pp. 637-651, 2010, doi: https://doi.org/10.1016/j.tourman.2009.07.003

[10] B. Burger, A. Puupponen, and T. Jantunen, "Synchronizing eye tracking and optical motion capture: How to bring them together", in Journal of Eye Movement Research, vol. 11(2), pp. 1-16, doi: https://doi.org/10.16910/jemr.11.2.5

[11] A. Adolfsson, J. Bernal, M. Ackerman, and J. Scott, "Musical mandala mindfulness: a generative biofeedback experience". [Online] Available from: https://www.semanticscholar.org/paper/Musical-Mandala-Mindfulness-Adolfsson-Bernal/7db5aeb974611536134939e436be168f7905a946

[12] P.E. Hutchings and J. McCormack, "Adaptive music composition for games", in IEEE Transactions on Games, vol. 12(3), pp. 270-280, 2019, doi: https://doi.org/10.48550/arXiv.1907.01154

[13] K. Beilharz, "Designing Generative Sound for Responsive 3D Digital Environment Interaction", Proc. Computer Aided Architecture Design Research in Asia (CAADRIA'04), 2004.

[14] N. Collins, "Generative music and laptop performance", in Contemporary Music Review, vol. 22(4), pp. 67-79, 2003, doi: https://doi.org/10.1080/0749446032000156919

[15] H. Brouwer, Self-supervised Audio-reactive Music Video Synthesis: Measuring and optimizing audiovisual correlation. Delft, NL: TuDelft, 2021.

[16] J.R. Weinel, "Visualising rave music in Virtual Reality: Symbolic and interactive approaches", Proc. Electronic Visualisation and the Arts, London, 2020, pp. 78-84, doi: https://doi.org/10.14236/ewic/eva2020.13

[17] N.N. Correia, "AVOL: Towards an integrated audio-visual expression", in Journal of Visual Art Practice, vol. 10(3), pp. 201-214, 2012, doi: http://dx.doi.org/10.1386/jvap.10.3.201_1

[18] J. Weinel, "Technoshamanic visions from the Underworld II", Proc. Internet Technologies and Applications (ITA), 2017, pp. 345-346.

[19] A. Eigenfeldt, "Exploring moment-form in generative music", Proc. Sound and Music Computing Conference, 2016, pp. 1-6, doi: https://doi.org/ 10.5281/zenodo.851207

[20] A. Camurri, S. Hashimoto, M. Ricchetti, A. Ricci, K. Suzuki, R. Trocca, and G. Volpe, G, "Eyesweb: Toward gesture and affect recognition in interactive dance and music systems", in Computer Music Journal, vol. 24(1), pp. 57-69, 2000.

[21] C. Garoufis, A. Zlatintsi, and P. Maragos, "A Collaborative System for Composing Music via Motion Using a Kinect Sensor and Skeletal Data", Proc. Sound and Music Computing Conference (SMC-2018), 2018, pp. 415-421, doi: https://doi.org/10.5281/zenodo.1341738

 ISBN: 978-1-68558-048-3

On the Generation of External Representations of Semantically Rich Content for API-driven Document Delivery in the Headless Approach

Hans-Werner Sehring
Tallence AG
Hamburg, Germany
e-mail: hans-werner.sehring@tallence.com

***Abstract*—Complex systems often are built on the basis of services that are composed in a loosely coupled way. Data exchange between system components takes place in an external format that conforms to a system-wide agreed schema. Content Management Systems (CMSs) are one example of a class of systems that do not process data with fixed semantics, but that manage and publish meaningful content. Such systems require a consistent interpretation of data as content on both ends in order to preserve meaning. We argue that such a consistent interpretation requires mappings between content models underlying CMSs and data models used for communication, and that these mappings, therefore, must be shared by all components of the system. In order to justify this claim, we compare the expressiveness of plain data formats with that of content modeling languages, and we study mappings between them. In this paper, we use JSON and JSON Schema as a typical examples of external data representations. We discuss content models using the example of the Minimalistic Meta Modeling Language (M³L). Our initial research shows that schemas for data exchange should be tightly linked to content models in order to not only represent content as data, but also to allow for consistent interpretations of content.**

***Keywords*—content model; data schema; schema mapping**

I. Introduction

Content Management Systems (*CMSs*) are an established tool for (in particular online) content publication. They are software systems that incorporate various functions for content creation, editing, management, (automated) document creation based on layouts, and document delivery. Over time, many CMS products started integrating additional functionality to keep up with emerging requirements. At the same time, such products became increasingly complex because many of them incorporate new functions in a monolithic way.

Since they often provide a comprehensive software infrastructure comparable to an application server, many content management solutions are built using a CMS as a platform. Custom code is integrated in to the CMS, making the overall solution an even larger monolith. This approach is often suitable for purely content-based functionality.

With the landscape of digital communication solutions becoming more complex, there is an increasing number of services that integrate data and services for other entities than structured content – media files, customer data, product data, etc. The services need to interface with CMS solutions. Systems typically require application-specific integrations (see, for example, [1]). These integrations make systems that rely on data exchange with a centralized monolithic platform overly complex since they have to deal with a variety of data exchange formats and different entity lifecycles.

Since some years, an opposite trend become popular under the name *headless CMS*. Such CMSs basically concentrate on basic content creation, editing and management functions. Content publication is provided by a delivery service that makes "pure" content accessible in the form of *Application Programming Interfaces* (*APIs*). All additional services are provided by separate software components. This includes document preparation and delivery that is implemented outside of a headless CMSs.

Systems that incorporate CMSs using APIs typically are built following microservice architectures. These consist of multiple self-contained services that provide one functionality each, with the CMS providing content as one of those services. System properties are established by service orchestration in the overall architecture.

APIs for access to content consist of service signatures and of structured content representations that are used as input and output parameters. Content representations typically focus on structured content – mainly textual content and descriptions of unstructured content. Unstructured content, be it provided by a CMS or a Digital Asset Management system, is typically transferred in some binary format.

RESTful APIs are a current de-facto standard for communication between distributed CMS components. The *JavaScript Object Notation* (*JSON*) is the usual language chosen to represent (structured) content. Section II names typical aspects of APIs defined this way.

Though the idea of using simple interfaces based on a simple data exchange format is appealing, it constitutes an "impedance mismatch" with rich content structures as employed by capable CMSs. Ideally, a CMS provides various means of structuring content. Many allow defining a *schema* or *content model*. Such a schema is, on the one hand, used to provide type safety to functions handling content, and on the other it constitutes the basis to capture the meaning of content. To make use of structure and meaning assigned to content, content structure and semantics defined by content models need to be preserved in external representations, and they are used as a basis to map content to external form.

In Section III, we introduce the Minimalistic Meta Modeling Language (M³L) as an example of a rather powerful content modeling language.

 ISBN: 978-1-68558-048-3

We use the M³L's capabilities for binding to external representations to study some aspects of interfaces for content access and interchange. In particular, we demonstrate different cases of JSON generation and parsing in Section IV and discuss general differences of custom generated JSON and such generated by content models formulated in the M³L in Section V.

We conclude the paper in Section VI.

II. Standards: RESTful APIs, JSON, and GraphQL

Approaches for (remote) APIs and their implementations are of general interest since the advent of distributed systems. After a series of technological approaches, a current de-facto standard for online interfaces has emerged from REST, JSON, and (increasingly) GraphQL.

A. RESTful APIs

Representational State Transfer (REST) was proposed by Fielding as the principle of communication in the Internet [2]. It calls for stateless servers and clients that handle state between request. In conjunction with URLs that represent services calls, the definition of so-called RESTful APIs allows defining simple APIs for Web-based services.

Such APIs consist of service call signatures composed of an HTTP method and a URL that specify the service to be used and the input parameters. The response to a service call is a regular HTTP response. A typical response format for structured data is JSON as discussed in the subsequent section.

RESTful APIs can be implemented with existing Web technologies, for example, typical software libraries available for all relevant programming languages and existing software components to build a service infrastructure.

B. Content Interchange with JSON

JSON is an object language for JavaScript, allowing to formulate JavaScript object instances, where *instance* refers to data contained in object properties, not any internal object state. It can be used for data storage, transmission, and aggregation.

JSON is typically used as a response format of RESTful interfaces. It provides a simple means of structuring data with typical collection types, and encoding of data as character strings.

Most API-based CMSs use JSON to distribute content. They typically do so by representing content in a straight-forward manner using the structuring means and primitive data types of JSON.

Internally, CMSs allow the definition of content models that describe content. Such models are the basis for describing content, for content editing, and also for JSON generation.

Document rendering presents content in visible form for consumption, for example, in the form of HTML files. In API-based CMS solutions, rendering is performed by external rendering engines (on client-side or on server-side). The rendering process is driven by *templates* that define how to layout content. Template code makes use of knowledge about the meaning of content to be represented.

Its external form in JSON is rather generic, though. JSON can be generated in an application-specific form, but basically contains structured data. The representation of content in JSON and interpretation from that format rely on consistent code on both producer's and consumer's side. Such interpretation cannot rely on JSON representations of content alone.

C. JSON Schema Languages

JSON is appealing because of its simplicity combined with reasonable expressivity. It was defined merely for the description of single records of data. Many applications call for a schema, though, that describes how classes of data are structured.

Several schema languages have been defined for JSON, most prominently *JSON Schema* [3]. Another proposal for a JSON schema language is JSound [4]. Other approaches are Joi [5] for JavaScript applications and Mongoose [6] for configurations of the database system MongoDB.

In this paper, we use JSON Schema for the discussion of schema properties.

III. A Short Introduction into the Minimalistic Meta Modeling Language (M³L)

For the discussion in this paper, we use the M³L since it proved to be a suitable language for the modeling of various aspects of content management. To this end, we briefly introduce the M³L, and we present some exemplary base models for content management and for content interchange based on RESTful APIs.

A. A Short Introduction into the M³L

In this section, we briefly introduce the M³L by highlighting those features that are central to the underlying experiments.

The basic M³L statements are:

- **`A`**: the declaration of or reference to a *concept* named *A*
- **`A`** `is a` **`B`**: refinement of a concept *B* to a concept *A*. *A* is a *specialization* of *B*, *B* is a *generalization* of *A*.
- **`A`** `is a` **`B`** `{` **`C`** `}`: containment of concepts. *C* belongs to the *content* of *A*, *A* is the *context* of *C*.
- **`A`** `|=` **`D`**: the *semantic rule* of a concept. Whenever *A* is referenced, actually *D* is bound. If *D* does no exist, it is created in the same context as *A*.
- **`A`** `|-` **`E F G.`**: the *syntactic rule* of a concept that defines how a string is produced from a concept, respectively how a concept is recognized from a string. When the representation of *A* is requested, it is produced by a concatenation of the strings produced out of *E*, *F*, and *G*. When no syntactic rule is defined, a concept is represented by its name. Vice versa, an input that constitutes the name of a concept without a syntactic rule leads to that concept being recognized.

If a concept that is referenced by one of the statements exists or if an equivalent concepts exists, then this one is bound. Otherwise, the concept is created as defined by the statement.

Existing concepts can be redefined. For example, with the definitions above, a statement

```
A is an H { C is the I }
```
redefines *A* to have another generalization *H* and *C* (in the context of *A*) to have *I* as its only generalization.

Every context constitutes a *scope*. A redefinition of a concept in a context is only applied in that context. When a redefinition of a concept takes place in another context as the original definition, we call that redefinition a *derivation*.

The concepts that are defined by such statements are *evaluated* when used. Evaluation means looking up or creating concepts and applying semantic rules.

Before a concept is referenced and before a statement is evaluated, all concepts are *narrowed down*. The narrowing of a concept is computed as follows:

1) The effective definition of a concept in some context is the set of all definitions in that context and all of its base contexts (transitive).
2) If a concept *A* has a subconcept *B*, and if all concepts defined in the context of *B* are equally defined in the context of *A*, then each occurrence of *A* is narrowed down to *B*.

Given the sample M³L statements:

```
Person { Name is a String }
PersonMary is a Person { Mary is the Name}
PersonPeter is a Person {Peter is the Name
                         42 is the Age }
```

the result of an additional statement

```
Person { Peter is the Name 42 is the Age}
```

is narrowed down to *PersonPeter* since *PersonPeter* is specialization of *Person* and its whole content matches. The statement

```
Person { Mary is the Name 42 is the Age }
```

is not narrowed down further. It does not match *PersonPeter* since *Name* has a different specialization, and it does not match *PersonMary* since it has no matching content concept called *Age* or *42*.

B. Basic Content Management and Document Rendering

The M³L is universal and has many applications. Amongst other modeling tasks, it has proven useful to describe content as lined out in, e.g., [7]. This applies both to content models as well as content items since the M³L does not distinguish model layers, such as type and instance.

For example, with a content model like:

```
Article is a Content {
 Title is a String
 Text is a FormattedString }
```

according content can be created:

```
NewsArticle123 is an Article {
 "Breaking News" is the Title
 "This is a report on ..." is the Text }
```

For textual formats, like HTML and JSON, documents can be rendered from content through syntactic rules of content as introduced in the previous subsection. On the level of the content model, syntactic rules describe document templates, on the content item level they render single document instances.

For the sample content definitions above, a JSON template may look like:

```
Article |- "{\"title\":\"" Title
           "\",\"text\":\"" Text "\"}"
```

This syntactic rule produces as JSON output for the concept *NewsArticle123* from above: `{"title":"Breaking News","text":"This is a report on ..."}`

The syntactic rule defines a JSON structure into which the concepts from the content are integrated. These may themselves evaluate to content strings of embedded JSON structures.

Please note that, e.g., *{\"title\":\"* is a valid concept name, as is *\"}*. Since new concepts are declared the first time they are referenced, and because they syntactically evaluate to their name by default, they can be used like string literals. The concept name \" is an escape sequence for the quote character (not a quote sign for identifiers).

IV. Producing JSON using the M³L

As outlined in the preceding section, the M³L can serve as an example of a an expressive content modeling language. For API-driven content distribution, structured content needs to be represented in an external form. In state-of-the-art services, this external form is JSON.

JSON Schema allows defining valid forms of JSON structures so that content can be transferred in a reliable manner. It is not expressive enough by itself, however, to recover equivalent content on the receiver's side. Custom code is required to generate JSON out of rich content structures. Appropriate code that shares the same conception of content is required to interpret JSON data.

Reference [8] points out that schema design for JSON requires careful consideration and that even finding sample instances for a given schema is a non-trivial task since semantics is scattered over a set of definitions and constraints.

JSON Schema provides various ways of defining and relating schemas. There are multiple ways of expressing equivalent schemas and equivalence cannot generally be proven [9].

One way of sharing content concepts between sender and receiver is to have a common content model and mappings to and from external representations. We exemplify this by utilizing the capabilities of the M³L for some sample constructs.

A. Defining Lexical Rules for JSON

M³L's lexical rules can produce JSON code out of concepts as sketched in Section III-B.

The M³L does not distinguish between "types" and "instances". This distinction is, however, required in classical approaches as JSON and JSON Schema.

In addition to the above sample rules that generate JSON, the lexical rules of other concepts may produce JSON Schema. See the following simple rules for the content example:

```
Article |-
"{\"type\":\"object\",\"properties\":{\""
   Title "\":{\"type\":\"string\"},"
   Text "\":{\"type\":\"string\"}}}"
```

Such a definition resulting in the production of the following JSON Schema definition:

```
{ "type": "object",
  "properties": {
    "Title": { "type": "string" },
    "Text":  { "type": "string" }}}
```

Lexical rules for both JSON and JSON Schema require to distinguish between schema and instances. Contextual definitions allow defining both layers for a concept. The decision between schema and instance has to be made explicitly which is atypical for M³L applications. For the content example:

```
SchemaRules { Article |- ... }
InstanceRules { Article |- ... }
```

In any case, a fair amount of extra code is required to state the obvious lexical rules per concept. It is approximately the same effort like providing custom mappings in software.

The effort of mapping an internal content model to its external forms is beneficial, though, to be able to recover the semantics of content. This way, schema definitions contribute to the exchange of meaningful content. In the subsequent subsections, we compare the modeling capabilities of JSON Schema and the M³L for the generation of JSON representations of content.

B. Basic Model Mapping from M³L to JSON

Simple M³L expressions that represent content instances can be expressed in a straight-forward manner as outlined by the content example. Some information is lost in the JSON representation, though. In the example above, the concept name *Article* is not communicated.

Such concept information may be reflected in dedicated properties. But more information on the content is lost if we add content types and descriptions, for example in M³L:

```
Person {
 FirstName is a String
 LastName is a String
 Address }
Address {
 Street is a String
 City is a String }
JohnSmith is a Person {
 John is the FirstName
 Smith is the LastName
 JohnSmithsAddress is the Address {
  "Main Street" is the Street
  Lincolnshire is the City } }
```

Syntactic rules may product the following JSON:

```
{ "FirstName": "John",
  "LastName": "Smith",
  "Address": { "Street": "Main Street",
               "City": "Lincolnshire" } }
```

The intended data structure can be defined by means of JSON schema that is also generated from the content concepts, for example, as follows:

```
{"title": "Person",
 "type": "object",
 "properties": {
  "FirstName": {"type": "string"},
  "LastName": {"type": "string"},
  "Address": {"$ref":"#/$defs/Address"} },
 "$defs": {
  "Address": {
   "type": "object",
   "properties": {
    "Street": {"type": "string"},
    "City": {"type": "string"} } } } }
```

Here, the concept names *Person* and *JohnSmith* are not present in JSON. The content name *JohnSmithsAddress* is also missing; the "type" name *Address* is used instead.

Note that information is distributed over two structures, instance and schema, and declared in different languages. A JSON (instance) file does not make reference to the schema it is intended to comply with. Therefore, the matching schema has to be found by distinct means. Names – concept names in the case of the M³L – are not included in JSON, but are required for schema selection (*Person* in the above example). Additional information like an envelope structure, for example,

```
{ "JohnSmith": { "Firstname": "John" ... },
  "type": "Person" }
```

or explicit properties, for example,

```
{ "$name": "JohnSmith",
  "$type": "Person",
  "Firstname": "John", ... }
```

would be required.

In order to generate two external forms – JSON and JSON Schema – out of one integrated internal content representation, two lexical rules are required as mentioned in Section IV-A. When parsing JSON on the receiver's side, the unrelated files need to be recombined in a content representation. JSON (Schema) provides no means to do so.

C. Capturing Type Variations

Variants of content are commonly found in CMSs since one schema typically does not cover all aspects content used for communication. Few CMSs cover variations explicitly in content models. The M³L, however, allows reflecting variants by means of concept refinement and by contextualization.

Consider concepts modeled after an example from [10]:

```
Address {
 "street_address" is a String
 "city" is a String
 "state" is a String
 Type }
BusinessAddress is an Address {
 Business is the Type
 Department is a String }
ResidentialAddress is an Address {
 Residential is the Type }
```

 ISBN: 978-1-68558-048-3

JSON Schema introduces the *"if"..."then"..."else"* construct for content variants. An example from [10] reflects the above M³L definitions:

```
{
 "type": "object",
 "properties": {
  "street_address": { "type": "string" },
  "city": { "type": "string" },
  "state": { "type": "string" },
  "type": {
   "enum": ["residential", "business"] }
 },
 "required": ["street_address",
              "city", "state", "type"],
 "if": {
  "type": "object",
  "properties": {
   "type": { "const": "business" }
  },
  "required": ["type"]
 },
 "then": {
  "properties": {
   "department": { "type": "string" }
  }
 },
 "unevaluatedProperties": false
}
```

In fact, M³L would (also) work the other way round: if an *Address* with an extra *Department* is given, it is derived to be a *BusinessAddress*. The *Type* attribute is not required by the M³L. This narrowing of the M³L – and similar, yet implicit behavior of typical CMS applications – makes matching JSON data to JSON Schema definitions yet more difficult.

V. Comparison of Plain JSON and M³L Constructs

In contrast to typical data schemas, content models are not only concerned with constraints on values, references, and structure, but additionally try to capture some semantics. Furthermore, while data aims at representing one consistent state of entities, content deals with varying forms and utilizations used in communication: different communication scenarios, contexts of users who perceive content, language and other localizations, etc.

This section points out some of the differences in expressiveness of data schemas and content models using the examples of JSON schema and the M³L.

A. Subtypes

Type hierarchies allow intensional descriptions of schema elements and are, therefore, found in content models. They are not ubiquitous in data models, though. JSON schema does not feature subtyping.

JSON schema does have means to express schema variants (*"if"*, *"dependentRequired"*) and to relate different schemas (*"dependentSchemas"*, *"allOf"*, *"anyOf"*, *"oneOf"*).

These can be used to model specializations of data as variants. An example is presented in Section IV-C above.

Any forms of refinements ("subtypes") in JSON weakens the constraints of a JSON Schema since not all properties can be *"required"* or *"additionalProperties"* and *"unevaluatedProperties"* must be allowed - very much as in the M³L.

The additional *Department* property from the example above can be introduced conditionally using the *"if"..."then"..."else"* construct in JSON schema. This allows representing subtypes. The information that a *BusinessAddress* is an *Address* is lost, however, both on instance and schema level. Therefore, this is not a suitable representation of refinement that conveys semantics.

B. Single and Multi-valued Relationships

It is quite common in content models to be vague about arity. For example, some pieces of content may typically have a 1:1-relationship, making it unary in the content model. But there are exceptions of n-ary cases that also need to be covered. The M³L allows to define concepts with *is a* and *is the* to take this into account.

A typical data model would define an n-ary relationship, even though in most cases the data are 1:1.

JSON itself allows to easily vary between unary and n-ary properties by simply stating either *"a":"b"* or *"a":["b","c"]*. JSON Schema, though, needs to define arity or to define variations with *"if"..."then"..."else"*.

Consider as an example a person with two addresses:

```
Person {
 FirstName is a String
 LastName is a String
 Address }
Address {
 Street is a String
 City is a String }
JohnSmith is an Employee {
 John is the FirstName
 Smith is the LastName
 JohnSmithsAddress is an Address {
  "Main Street" is the Street
  Lincolnshire is the City }
 JohnSmithsOffice is an Address {
  "High Street" is the Street
  Lincolnshire is the City }
}
```

A JSON structure reflecting this content is:

```
{
 "FirstName": "John",
 "LastName": "Smith",
 "Address": [
  { "Street": "Main Street",
    "City": "Lincolnshire" },
  { "Street": "High Street",
    "City": "Lincolnshire" }
 ]
}
```

 ISBN: 978-1-68558-048-3

Though this is a small change to the JSON structure, it has to be explicitly foreseen in JSON Schema. It is not as easy to vary between one or multiple addresses (in this example) as it is in content models like the M³L or the Java Content Repository [11].

C. Content Conversions and Computed Values

It is common for content models to not only contain content itself but also descriptive information about the content (sometimes referred to as "meta data").

For example, a simple data property like

```
{"price": 42}
```

requires additional information to be interpreted correctly (the currency in this example). In simple data models, there is an additional documentation that establishes an agreement on how applications should deal with the data. The possibility to state the unit of measurement is typically found in *Product Information Management systems*.

In these cases, the information needs to be state explicitly, as it is done in typical master data management systems:

```
{"price": {"value":42, "currency":"€"}}
```

Such a record allows a mutual understanding of the value. It prevents an easy mapping from JSON to a numeric *price* variable, though.

As a slight improvement, values should be replaced by named concepts. The M³L captures meaning by defining relevant concepts. For example, a concept like *EuroCurrency* as a refinement of a concept *Currency* would be used instead of the string value *€*.

On top of descriptive information on content, a content model may also define a limited set of computational rules in order to define consistent arithmetics.

The M³L is expressive enough to define some (symbolic) computation. Assume, for example, a concept *Integer*, concrete "instance" concepts like *100*, and concepts describing computations like *FloatDivision*, the division of numeric values.

On the basis such definitions, it is possible to state conversion rules like the following:

```
Price {
 Value is a FloatNumber
 Currency }
PriceInEuro is a Price {
 € is the Currency }
PriceInEuroCents is a Price {
 Value is an Integer
 Cents is the Currency }
|= PriceInEuro {
 Value is a FloatDivision {
  Value is the Dividend
  100 is the Divisor } }
```

These sample definitions define (on schema level) how values are converted so that all clients using this model share the same arithmetics.

VI. Summary and Outlook

We conclude with a summary and an outlook.

A. Summary

We compare rich content models – using the example of the modeling capabilities of the M³L – with typical data schemas, in particular JSON Schema. We conclude that models for meaningful content cannot adequately be expressed by data schemas alone.

JSON became a de-facto standard for content exchange. We present examples showing that the currently evolving schema language, JSON Schema, is not sufficient for content modeling in its current form.

B. Outlook

Additional research is required to identify the full expressivity required to define external representations of content for modern content management approaches. This will guide future investigations towards a suitable set of modeling capabilities for JSON Schema.

The M³L is not intended to be a data schema language. Therefore, it lacks some features of such languages. It will be an experiment, though, to define a M³L derivate that is able to serve as an alternative schema language for JSON.

Acknowledgment

The author thanks numerous colleagues, partners, and clients for fruitful discussions on various topics centered around digital communication. He thanks his employer, Tallence AG, for the support in the publication and presentation of this work.

References

[1] H.-W. Sehring, "On the integration of lifecycles and processes for the management of structured and unstructured content: a practical perspective on content management systems integration architecture," International Journal On Advances in Intelligent Systems, volume 9, numbers 3 and 4, pp. 363–376, 2016.

[2] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Doctoral dissertation, University of California, 2000.

[3] A. Wright, H. Andrews, B. Hutton, and G. Dennis, "JSON Schema: A Media Type for Describing JSON Documents," Internet Engineering Task Force, 2022.

[4] C. Andrei, D. Florescu, G. Fourny, J. Robie, and P. Velikhov, *JSound 2.0*. [Online]. Available from: http://www.jsound-spec.org/publish/en-US/JSound/2.0/html-single/JSound/index.html [retrieved: May, 2023]

[5] *The most powerful schema description language and data validator for JavaScript*. [Online]. Available from: https://joi.dev/ [retrieved: May, 2023]

[6] *Mongoose*. [Online]. Available from: https://mongoosejs.com/ [retrieved: May, 2023]

[7] H.-W. Sehring, "On Integrated Models for Coherent Content Management and Document Dissemination," Proceedings of the Thirteenth International Conference on Creative Content Technologies, CONTENT 2021, ThinkMind, 2021, pp. 6–11.

[8] L. Attouche et al., "A Tool for JSON Schema Witness Generation," Proceedings of the 24th International Conference on Extending Database Technology. OpenProceedings.org, March 2021, pp. 694–697.

[9] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, "Finding Data Compatibility Bugs with JSON Subschema Checking," Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2021. Association for Computing Machinery, July 2021, pp. 620—632.

[10] M. Droettboom, "Understanding JSON schema," Space Telescope Science Institute, 2023.

[11] D. Nuescheler et al., "Content Repository API for Java™ Technology Specification," Java Specification Request 170, version 1.0, May 2005.

 ISBN: 978-1-68558-048-3