BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

ALGORITHMS FOR AUTOMATIC SPHERICAL IMAGE AND VIDEO CROPPING

ALGORITMY PRE AUTOMATICKÉ OREZANIE SFÉRICKEJ FOTOGRAFIE A VIDEA

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. MARTIN IVANČO**
AUTOR PRÁCE

SUPERVISOR **doc. Ing. MARTIN ČADÍK, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Ivančo Martin, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Algoritmy pro automatický ořez sférické fotografie a videa**
Algorithms for Automatic Spherical Image and Video Cropping
Kategorie: Zpracování obrazu

Zadání:

1. Seznamte se s problematikou automatického ořezu obrazu.
2. Vyberte a popište metody vhodné pro implementaci ořezu sférické fotografie a videa. Při výběru se soustřeďte na použitelnost výsledků z fotografického hlediska.
3. Alespoň tři zvolené metody implementujte.
4. S metodami experimentujte, posuďte jejich vlastnosti, implementované algoritmy porovnejte a diskutujte možnosti budoucího vývoje.
5. Dosažené výsledky prezentujte formou videa, plakátu, článku, apod.

Literatura:

- http://cadik.posvete.cz/

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Čadík Martin, doc. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 1. listopadu 2019

Abstract

The aim of this work is to provide a detailed overview of the research regarding spherical videos. Specifically, this work focuses on the task of creating a normal field of view video from a spherical one for viewing purposes. It provides an implementation of some of the available methods as well. So far, there have been three methods introduced by four papers tackling this challenge. All of them have brought promising results and this work examines two of them thoroughly. It also provides a baseline method leveraging well known techniques of automatic image cropping. This method serves as a good comparison tool which puts the improvements brought by the examined methods into perspective. Based on a thorough comparison via a user study this work concludes, that the best method for this task overall is a variation of the method by Pavel *et al.* [14] introduced in this work.

Abstrakt

Cieľom tejto práce je priniesť detailný pohľad na doterajší prieskum v oblasti sférických videí. Konkrétne sa táto práca zameriava na problém tvorby videa s normálnym zorným poľom zo sférického videa pre potreby zobrazovania. Prináša tiež implementáciu niektorých dostupných metód. Doteraz boli predstavené tri metódy v štyroch článkoch, ktoré riešia tento problém. Všetky priniesli zaujímavé výsledky a táto práca sa dvomi z nich zaoberá hlbšie. Táto práca tiež prináša základnú metódu využívajúcu overené metódy automatického orezu obrazu. Táto metóda je využitá na porovnanie so skúmanými metódami, u ktorých zvýrazní ich vylepšenia ale aj nedostatky. Na základe porovnania metód pomocou užívateľského experimentu táto práca usudzuje, že najlepšou zo skúmaných metód pre túto úlohu je upravená varianta metódy od Pavel *et al.* [14], predstavená v tejto práci.

Keywords

Automatic 360° cinematography, automatic image cropping, spherical video, 360° video retargeting

Kľúčové slová

Automatická 360° kinematografia, automatický orez obrazu, sférické video, usmerňovanie 360° videa

Reference

IVANČO, Martin. *Algorithms for Automatic Spherical Image and Video Cropping.* Brno, 2020. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Ing. Martin Čadík, Ph.D.

Rozšírený abstrakt

Cieľom tejto práce je priniesť detailný pohľad na doterajší prieskum v oblasti spracovania sférických videí. Konkrétnejšie sa táto práca zameriava na dostupné metódy orezu sférického videa za cieľom zobrazovania takýchto videí na bežnej 2D obrazovke. Dve takéto metódy sú následne vybrané a podrobne popísané. Detailne popísaná je aj základná metóda ktorá je založená na štandardných technikách v oblasti automatického orezu obrazu. Táto metóda slúži ako dobrý nástroj na porovnanie s metódami ktoré sú špeciálne vyvinuté pre účely orezu sférického videa. Z porovnania so základnou metódou bude potom jasne vidno v ktorých oblastiach priniesli zlepšenie a aké sú ich slabosti

Základná metóda je predstavená v troch variantách založených na troch rôznych metódach automatického orezu obrazu. Každá z variant pracuje tak, že orezáva video snímku po snímke a orezané video je následne transformované z equirektangulárnej na stereografickú projekciu, ktorá je štandardom pre zobrazovanie sférických videí. Detailný postup orezu obrazu sa u použitých metód líši, ale u každej z nich sa dá rozdeliť na dve časti – mapovanie *saliency* a hľadanie optimálneho orezového obdĺžnika pre vypočítanú saliency mapu. Tá má za cieľ zvýrazniť v obraze oblasti, ktoré priťahujú ľudskú pozornosť.

Prvá z vybraných metód určených špeciálne pre orez sférického videa je *AutoCam* od Su *et al.* [20]. Táto metóda funguje v troch fázach. V prvej fáze sa zo vstupného sférického videa vygenerujú takzvané časo-priestorové *glimpsy*. Sú to krátke videá ktoré sú akoby pohľadmi do sférického videa. Existuje preddefinovaná množina sférických súradníc rovnomerne rozložených na povrchu imaginárnej gule obklopujúcej 360° kameru ktorá je v jej strede. Pre každú z týchto súradníc sa vygeneruje niekoľko krátkych videí pokrývajúcich celú dĺžku videa pomocou transformácie z equirektangulárnej na stereografickú projekciu so stredom premietania práve na danej súradnici. Výsledkom je množina krátkych 2D videí ktoré dokopy kompletne zachytávajú okolie 360° a dalo by za z nich znovu rekonštruovať pôvodné sférické video. V druhej fáze sa glimpsy ohodnotia. Cieľom je priradiť glimpsom skóre na základe toho, aký zaujímavý alebo dôležitý je obsah ktorý zachytávajú. Pre naplnenie tohto cieľa sa využíva *data-driven* prístup založený na extrakcii *C3D features*. Výstupné skóra sa potom využijú v tretej fáze, ktorá hľadá najlepšie ohodnotenú sekvenciu glimpsov. Súradnice týchto glimpsov potom tvoria trajektóriu, ktorá sa použije na vygenerovanie finálneho videa. Okrem ohodnotenia pomocou C3D features je vysvetlený aj postup založený na mapovaní saliency.

Druhá vybraná metóda je *Automatic Importance Detection* od Pavel *et al.* [14]. Táto metóda je založená na priamej extrakcii rôznych feature máp zo snímok sférického videa uložených v equirektangulárnej projekcii. Kombinujú sa tri druhy feature máp ktoré pokrývajú oblasti z vysokou sémantickou hodnotou až po jednoduché features založené na kontraste. Prvá z nich je feature mapa zvýrazňujúca oblasti ktoré obsahujú ľudskú tvár, ktoré sú sémanticky veľmi dôležité. Druhá je vytvorená pomocou hodnôt optického toku a teda zachytáva temporálny a pohybový aspekt videa. Tretia je vygenerovaná pomocou rýchlej metódy saliency mapovania. Tieto mapy sa postupne agregujú do takzvaných *shot features*. V nich potom metóda hľadá výrazné vrcholy ktoré označujú dôležité časti videa. Najvýraznejší z vrcholov potom určuje horizontálnu sférickú súradnicu použitú pre daný záber vo videu. Týmto spôsobom sa získajú súradnice pre každý záber vo videu. Keďže každý nový záber značne zmení obsah vo videu, prípadná veľká zmena vo vybranej súradnici pre po sebe nasledujúce zábery nebude viditeľná. Okrem originálnej metódy je detailne popísaná aj varianta s plynulou zmenou súradníc. Tá je založená na lineárnej interpolácii trajektórie získaných súradníc z jednotlivých snímkov videa.

Nedielnou súčasťou tejto práce je aj implementácia všetkých týchto metód, ktorej technické detaily sú tiež popísané v tomto texte. Pomocou implementovaných metód boli následne vygenerované výstupy z datasetu sférických videí. Tieto výstupy boli porovnané pomocou užívateľského experimentu metódou *2AFC* od David *et al.* [4]. Tento experiment prebiehal prostredníctvom webovej stránky zobrazujúcej náhodnú dvojicu výstupov z dvoch rôznych metód z rovnakého vstupného videa. Respondent po zhliadnutí oboch videí vybral to ktoré považoval za zaujímavejšie a subjektívne lepšie. Výsledné dáta boli následne prevedené do stupníc z-skóre podľa *Law of Comparative Judgements* od Thurstone [22].

Z 1179 hodnotení 66 respondentov vyplynulo že metódou s najkvalitnejšími výsledkami je upravená varianta metódy od Pavel *et al.* [14] predstavená v tejto práci. Rozdiel medzi ňou a základnou metódou založenou na automatickom orezávaní od Fang *et al.* [6] je však štatisticky nevýznamný. Významný však je rozdiel v časovej efektivite – základná metóda je skoro 10 krát pomalšia ako originálna i upravená metóda od Pavel *et al.* [14], ktorá dokáže spracovať v priemere 3 snímky za sekundu. Celkovo najlepšou metódou je teda spomínaná varianta metódy od Pavel *et al.* [14], no prekvapivým zistením bolo, že žiadna z metód špeciálne navrhnutých pre túto úlohu nedokázala produkovať stabilne lepšie výsledky ako základná metóda založená na modernom automatickom oreze klasického 2D obrazu. Pre objavenie metódy ktorá by stabilne produkovala najlepšie výsledky nezávisle na obsahu vstupného videa tak bude potrebný ďalší výskum.

Algorithms for Automatic Spherical Image and Video Cropping

Declaration

Hereby I declare that this diploma thesis was prepared as an original author's work under the supervision of Mr. Martin Čadík. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Martin Ivančo
May 20, 2020

Acknowledgements

I would like to thank my supervisor Doc. Ing. Martin Čadík, Ph.D. for his continuous support and professional advice throughout the making of this work. I would also like to express my gratitude for the all of the insights on the innovative work of prof. M.S. Kristen Grauman, Ph.D. at University of Texas at Austin, and her former student, M.S. Yu-Chuan Su, Ph.D. at Google, they kindly provided me with. Finally, I would like to thank B.S. Amy Pavel, Ph.D. at Carnegie Mellon University and Apple, and prof. B.S. Maneesh Agrawala, Ph.D. at Standford University, for taking the time to answer all of my questions regarding their extensive research in the field of spherical videos and virtual reality.

Contents

Chapter 1

Introduction

The ability to capture our surroundings not only as static photographs, but moving images as well has been around for more than a century. Nowadays, it is considered a basic feature of any consumer camera, be it a stand-alone DSLR or an inbuilt camera of a smartphone, to be able to capture video. Only in the recent years, however, a new possibility opened up in the consumer space to capture our surroundings completely using a 360° cameras. These cameras produce spherical video which can then be viewed ideally using a virtual reality headset, or, much more commonly, on a standard 2D screen. The user then has a normal field of view of the surroundings, with the ability to alter its direction using some kind of controls, usually clicking and dragging a computer mouse. This forces the user to interact with the video, which can be seen as a positive aspect in some circumstances, however, most of the time this is an inconvenience for the user and simply watching the video without interaction would be preferable. On top of that, even if the user would be willing to interact with the video, it is likely that they would miss an important part of the video, forcing them to re-watch it. As these 360° cameras become more and more widespread due to the technology becoming cheaper as well as all major social platforms implementing support for uploading and viewing spherical videos, this problem is starting to gain traction.

The definition of the problem this work tackles is therefore quite simple – create a normal field of view video by cropping an input spherical video in a way that would keep the most important or interesting parts in the field of view. The solution to this problem, however, is not as simple and so far, no definitive method that would work in every possible scenario has been found.

This work aims to research, implement and compare some of the notable methods of solving this problem, that were introduced in recent years in several scientific articles. It first explains the basic principles and advances in related areas of research. These play a major role in understanding the underlying techniques used in the actual existing solutions. It then introduces a baseline method which is builds upon the techniques from related fields. This method serves as a good comparison tool to put the improvements brought by the methods focused on this specific problem into perspective. Two of these methods are then discussed in detail, with several possible modifications and ideas for improvement provided as well. Chapter 4 lays out the details of implementation of these methods. Finally, these methods are thoroughly compared in a user experiment. This paper ends with a summary of the obtained results and possible directions for future research.

Chapter 2

Related Work

To fully understand all of the principles utilized in the chosen methods, it is useful to have an overview of the related areas of research. This chapter shortly introduces each one, from automatic image cropping through video retargeting to the recent advancements in the task of automatic 360°video cinematography, which is the focus of this work.

Automatic image cropping

The task of automatic image cropping is to find an optimal crop of an image based on contrast, saliency (explained in detail in the next section), visual composition or other measures automatically. The crop can be constrained to have a certain aspect ratio, cover a given portion of the image or have a certain specific dimensions. This topic has been researched thoroughly and it's use is quite widespread, from cropping photos to make a photographers editing job easier to automation of creating representative thumbnails.

One of the first papers to tackle this topic is one by Suh *et al.* [21]. They explore the usage of visual saliency mapping and face detection to find important objects in an image and crop the image tightly around it for a good thumbnail. Since then, automatic image cropping has advanced rapidly. One of the more recent papers on this topic is by Yan *et al.* [28], and it takes a popular, machine learning driven approach to this problem. In it, they train a neural network on a large training set of photos before and after cropping by a professional photographer. This work delivers promising results, however an even more recent work by Fang *et al.* [6] shows that machine learning might not be the right choice for this task. This paper combines three models (visual composition, boundary simplicity and content preservation) to achieve much higher quality results when compared to previous attempts and in a user study it easily beats the method by Yan *et al.*

While automatic image cropping is already a highly researched topic, when it comes to video, the task becomes much more complicated as the temporal aspect of it brings new data as well as new constraints to the equation. This task is, however, also quite thoroughly researched and the current state of the art methods are explained in a further section. Although image cropping is quite different and easier task than the one tackled by this work, it still remains a good baseline for evaluating new approaches to automatic video cropping. I will therefore be using methods described in works by Suh *et al.* [21], Stentiford [17] and Fang *et al.* [6], implemented by Ambrož V. in his recent work [1].

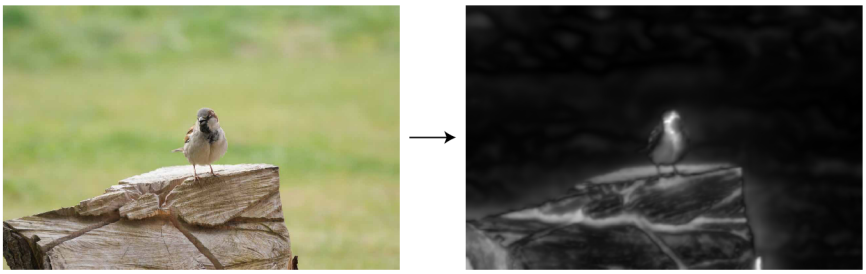


Figure 2.1: The image on the right is a saliency map of the image on the left. The method used for creating this saliency map is the one by Itti and Koch [10], and I used the implementation by Ambrož [1].

Saliency mapping

Visual saliency is a quality of an object, or, in terms of images and video, an area on an image, that makes it stand out from its surroundings. The most salient objects or areas are the ones that grab the most attention of a human eye. Naturally, this is a highly volatile quality, as it differs from person to person. However, there are trends that can be observed when tracking the focus of the eyes of a larger group of people, that show that some objects or areas of an image tend to get more attention than others. Saliency mapping or modeling is the process of creating a map (or model) of the saliency values across an image. An example of this can be seen in Figure 2.1. This is a highly researched topic and there are many approaches to creating such a map. I am going to mention 3 of them.

Itti and Koch's [10] saliency mapping algorithm is one of the most famous baselines for saliency mapping as it is based on low level features and provides acceptable results with quite low computational cost. This algorithm is also used in the automatic image cropping algorithm by Suh *et al.* [21]. There are many other models for saliency mapping for images, however, as mentioned before, when it comes to video, there is much more data to work with. One of the more recent papers by Kim W. and Kim C. [11] takes advantage of the temporal aspect of video to create saliency maps with a lot more detail than those provided by Itti and Koch's algorithm. The saliency detection technique in this paper is based on textural contrast rather than being biased towards edges and corners as most other methods are. When it comes to 360° video though, the problem becomes even more complicated as the equirectangular projection heavily used for storing them introduces a significant amount of distortion near the upper and lower edges of the frames. A recent work by Zhang *et al.* [30] tries to solve this problem by creating a saliency mapping method specifically for 360° videos. The method is based on a convolutional neural network scheme with kernels defined as spherical crowns that are rotated around the sphere. This method brings promising results and is also more thoroughly researched and used in this work.

As mentioned before, saliency is a highly volatile measure and therefore it may prove unstable and unreliable for detecting important parts of frames. However, it is a topic that has had a lot of research already done on it and it is definitely one of the most used techniques for these kinds of tasks. Therefore, it is also a strong baseline to compare with other methods, especially when it comes to the more recent techniques, like the one discovered by Zhang *et al.* [30].

Figure 2.2: Example of video retargeting using different techniques, including seam-carving [27]. The column on the left shows original frames and the other columns are retargeted for 60% of original width.

Video summarization

Summarizing a video is the process of finding interesting parts of a long clip to create a new, shorter video, that should ideally communicate the same general story and information, with removed redundancy and uninteresting parts. This topic has been researched quite thoroughly, for example in the recent work by Lee *et al.* [12], which directly focuses on 360° video for video summarization. While this task might seem similar to the one researched in this paper, it focuses on finding the interesting parts in the temporal aspect of the video, while the task of this paper is to find interesting parts in the spherical domain of a 360° video. While video summarization, especially when used on 360° video, like in the work by Lee *et al.*, might also involve some kind of automatic video cropping to output a normal field of view video, it is not the primary focus of this task.

Video retargeting

Video retargeting is in it's principle the closest to the goal of this work. It tries to find a cropping window inside a video for the purpose of viewing it on a device it wasn't made for. A good example would be viewing a cinema movie on a small screen like for example on a smartphone. There are two problems that occur in this situation. The first one is the change in the aspect ratio. Cinema cameras usually shoot in a 1.85:1 or even 2.39:1 aspect ratio while smartphones usually have a screen with standard 16:9 ratio. The movie is then presented with black bars on top and bottom of the video effectively reducing the already small display size even more. The second problem is that the creators sometimes rely on the big size and resolution of a cinema screen to show the viewers small details that could be missed on a small, low resolution display. Video retargeting tackles this problem by finding interesting parts of each frame and cropping them accordingly, as can be seen in the examples on Figure 2.2. The techniques used for this task are similar to the ones used in automatic image cropping, however, they must ensure continuity between frames and they usually take advantage of the temporal aspect of videos to gain more data.

This topic has been researched quite thoroughly in the past years and there are many existing methods achieving this goal. One of the first works that researched this topic is one by Wolf *et al.* [26]. They proposed an efficient algorithm that cropped and scaled the video according to local saliency and motion and object detectors in a way that preserved the important parts of each frame as much as possible. A more recent work by Yan *et al.* [27] introduces a method that not only crops and scales the frames of the video but also cuts unimportant parts from the frame and creates „seams“ to cover discontinuity between pixels. This work also tries to find cropping windows of videos for the purpose of viewing them on a device it wasn't made for, however, when it comes to 360° videos, finding a warped cropping window on a warped frame poses a more difficult task than just finding a rectangular crop on a rectilinear video. Although some of the techniques used in video retargeting are also used in automatic 360° video cinematography, many of them fail completely or need to be adapted for the warped image.

Virtual cinematography

Virtual cinematography tackles the problem of positioning and orientating the camera in a virtual 3D environment. It has been researched before in several works, for example one by Elson and Riedl [5]. In comparison with automatic 360° video cinematography, virtual cinematography doesn't need to tackle the problem of finding and selecting important objects in a scene, since those are known and fully under control. The camera can also be freely moved in the space, while automatic 360° video cinematography deals with a locked camera path in the space and needs to find the optimal trajectory of camera rotations on the given path. This is quite a different task and therefore it doesn't share much techniques with virtual cinematography although the tasks are similar in essence.

Automatic 360° video cinematography

As mentioned in the previous sections, a lot of research has been done on many of the related tasks to automatic 360° video cinematography, however, this task in itself is quite new and not as thoroughly researched yet. To my best knowledge, so far there have been only four papers tackling this exact task. The first one by Su *et al.* [20] proposes a method that solves this problem by creating so called spatio-temporal „glimpses“ – short five-second clips sampled from the 360° video with normal field of view at different angles and times. These glimpses are then evaluated using a data-driven approach – they use a data-set of normal field of view web videos from similar environments as those appearing in the 360° ones to learn which views are considered interesting by human videographers. According to the scoring of the glimpses, they evaluate an ideal trajectory of angles in the 360° video that ensures continuity as well as keeping the interesting parts visible. They also experiment with different methods and baselines to evaluate the effectiveness of their method. This approach brings quite satisfying results that look like they could be captured by a human, however, it does so with a relatively high computation cost – generating all the glimpses even for a short video and evaluating all of them is a demanding task for commonly available computers. The method is also quite limited in the sense that, to evaluate a video, one needs to have a dataset of other spherical and normal field of view videos from the same environment.

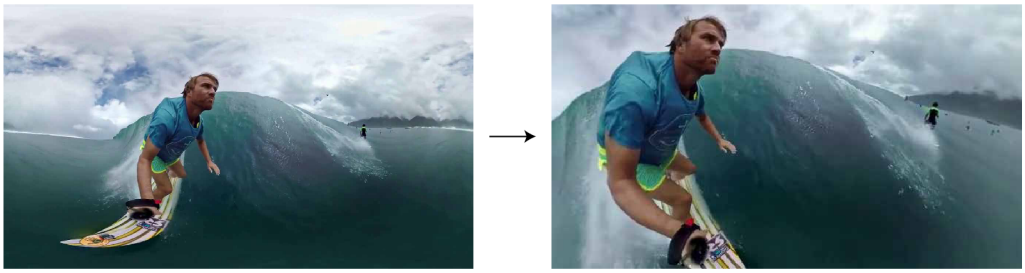


Figure 2.3: Example of a frame from spherical video in equirectangular projection (described in detail in section 3.1) on the left and a corresponding frame from one of the glimpses on the right.

In the follow-up work by Su and Grauman [19], they address two of the main drawbacks their last work had. Firstly, they add the capability of the system to generate glimpses at different fields of views, as most of the regular normal field of view cameras have the ability to change it and the videographers often use it to focus on a smaller object in detail, or on the on the other hand, capture a wider scene in its entirety. This capability, however, brings even more computational overhead for generating each glimpse for multiple fields of views as well as evaluating all of the new glimpses. The computational cost has been a problem even in the first work and they solve it in this one by dividing the approach into two parts – in the first one they sample the 360° video more sparsely with just one wide field of view and evaluate them. They then find the best trajectory in this sparsely sampled grid of evaluated glimpses and only in the second part do they create the glimpses at all of the angles and fields of views. However, they only generate and evaluate glimpses that are close to the trajectory computed by the first, sparsely sampled part of the method. This eliminates a lot of computational cost as they don't need to generate nor evaluate all of the glimpses, while barely affecting the quality of the output. Figure 2.3 illustrates the task of automatic 360° cinematography using example frame from an original spherical video and a potential corresponding selected view.

The work by Pavel *et al.* [14] is focusing on the experience of viewing 360° video in general and techniques on how to make it better. In the process, however, they also introduce a highly valuable method of automatic detection of important parts of the spherical video. Although this method was designed for choosing only the horizontal angle containing the most important details of the video, 360° videos are generally shot in a way that the interesting content is centered around the equator and rarely contain important details in the upper or lower regions. Thanks to that, this method is a worthy candidate for the task of automatic 360° video as well. This method is also much less computationally expensive than the one introduced by Su *et al.* [20], even when considering the optimized approach introduced in the follow up work. It also works on a wide range of different spherical videos by combining low-level and high-level features, such as visual saliency and face detection, without the need for a dataset of videos from similar environment.

The most recent work by Hu *et al.* [9] brings a new approach to this problem by using deep learning to select viewing angles with just the current 360° frame and the knowledge of previously selected viewing angles. They use a state-of-the-art object detector to find interesting parts of each frame, select the main focus and then shift the view in its direction while keeping the smoothness of the video. They compare this method to the one proposed by Su *et al.* [20] and find their method to achieve much better performance both

on quantitative benchmarks as well as user study. However, this method is domain-specific – it focuses on sports videos where there is usually one main object of interest and it might suffer on 360° videos where there are multiple equally important objects in the scene.

The approaches introduced by Su *et al.* [20] and Pavel *et al.* [14] will be thoroughly researched in this work. Initially, I wanted to include the method by Hu *et al.* [9], however, neither the published code, nor the paper itself detail the process of preparing data for the deep neural network, nor was I able to contact the authors. Experiments will therefore be carried out to compare the first two methods between themselves, as well as to other baselines and adapted methods detailed in the following chapter.

Chapter 3

Methods of Spherical Video Cropping

This chapter provides a thorough explanation of three methods of automatic spherical video cropping and their various modifications. The first one is a saliency baseline. It uses basic saliency mapping to find interesting objects in an equirectangular projection of the spherical video and then applies regular automatic cropping methods to find the crop. Although both the saliency mapping as well as the automatic cropping techniques are designed for regular 2D images or videos, and therefore are not expected to perform as well as the other methods that were designed specifically for spherical video cropping, they serve as a good baseline for comparison. Evaluation of saliency map using special 360° saliency detection designed for spherical images or video is also considered and explained in detail.

The second method is based on paper by Su and Grauman [19]. It splits the spherical video into spatio-temporal glimpses which it then scores using a data-driven approach. According to the scores it computes the best camera angle trajectory and based on which it can render a cropped video. This approach was the first to tackle this task specifically and brought promising results.

The third method is based on paper by Pavel *et al.* [14] and it uses a combination of face detection, optical flow and saliency to compute feature maps for each shot of the video. This method chooses the best possible direction in the space to target to crop to, while the field of view is static. It works best on edited videos, where there is less movement and multiple shots.

3.1 Saliency Baseline

As mentioned before, this method serves as a baseline for the other methods specifically designed for spherical video cropping and uses techniques that can be applied in quite a wide area of use cases. From a high level, this method can be divided into three steps. The first step is saliency mapping. In this step, saliency detection methods are applied to an equirectangular projection of the spherical frame of a 360° video to create a map of the frame which highlights the salient (interesting for a human eye) parts on it. Standard 2D saliency detection methods are considered as well as a special 360° method designed to be applied on spherical images in equirectangular projection.

The saliency map is used in the second step which is automatic cropping. The goal of this step is to find a cropping window for the original frame which contains the important

objects of the frame while leaving out details that are not interesting for the viewer. It tries to accomplish this by finding a cropping window that would contain most of the salient parts of the image, leaving out the non-salient ones. Some of them also search for other properties of the cropping window that further refine the final crop.

In a 2D scenario, a simple crop of the frame would finish the pipeline. In a 360° one, however, there is one last step that needs to be done and that is projection. The automatic cropping methods finds a cropping rectangle for the frame in standard equirectangular projection. This projection, while highly useful for storage purposes, also introduces distortions that are unnatural for human eye and is therefore not suitable for viewing. In this step, the central longitude and latitude as well as the angle of view for a stereographic projection are calculated from the cropping rectangle and then the equirectangular frame is projected into the stereographic projection which is much more suitable for viewing.

These three steps are executed on each frame of the spherical video to create a 2D cropped version. In the following subsections, each of them is thoroughly explained.

Saliency mapping

Saliency mapping uses a variety of different indicators and properties of an area or a point on an image to compute how salient it is – how strongly does it attract human eye. Since saliency mapping is not a new field of study, there has been many papers on this topic which brought a multitude of different methods. The monitored features evolved from naturally significant such as edges or other high contrast areas to much more complex and specific ones such as faces or various shapes. Saliency mapping has a wide scope of use cases, however, since early days of its research it has been used for cropping purposes – thanks to its ability to find areas of image that attract human eye and can be therefore considered important, interesting or at least containing a lot of visual information. This work employs three methods of saliency mapping which will now be shortly introduced.

Saliency mapping method introduced by Itti *et al.* [10] is a well known baseline method for saliency mapping which was inspired by the behavior and neural structure of the visual system of young primates. It is therefore independent from semantic information while it attains a relatively strong performance in many different scenarios. From a high level view, this method creates a number of scaled versions of the input image from which it then extracts feature maps which are then combined and normalized to create the final saliency map. Figure 3.1 shows an example of such saliency map as well as the saliency maps of the other two methods from the same input image.

Stentiford [17] introduces a different approach to saliency mapping. In his method the most salient regions are considered as the ones that are unique in the image. This method therefore compares small regions with others within the image. A region that does not match any, or just a few other regions is considered unique and will stand out from the background. This process can therefore assign high saliency score even to areas that do not have high contrast or sharp edges, but appear rarely in the image. It can also suppress the effect of backgrounds with high contrast, e.g. a wall with a stripe pattern, since the pattern is repeated many times in the image. This is usually a welcome behaviour in image cropping since the foreground is usually the point of interest, even when it is not the region with the highest contrast. This method however, also introduces a lot of computational overhead when compared to Itti's method.

Saliency mapping method introduced by Margolin *et al.* [13] brings a simple yet powerful principle of combining color and pattern distinctiveness. According to this method, salient

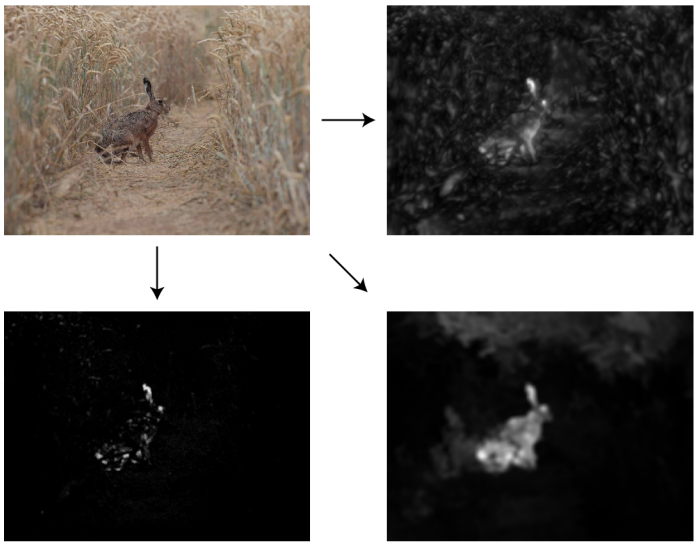


Figure 3.1: Comparison of saliency mapping methods used in this work. Top left is the original image. Top right is saliency map created using method by Itti *et al.* [10]. Bottom left is created using method by Stentiford [17]. Bottom right is created using method by Margolin [13]. All of these maps were generated using implementation by Ambrož [1].

regions are the ones that consist of pixels whose local neighborhood is distinct in both color and pattern. For pattern distinctiveness evaluation, this method uses a novel approach which only compares each small region to a single small region which is computed as an average of all in the image. This brings significant improvement in efficiency over comparing each small region to every other region. The color distinctiveness is evaluated by segmenting the image into regions and determining which of them are distinct in color. A normalized product of the pattern and color distinctiveness is considered the final saliency map – the product ensures that only the regions distinct in both pattern and color are considered salient. This method is more efficient than the one introduced by Stentiford but isn't as fast as the one by Itti.

Figure 3.1 shows a comparison of these three methods. Their implementation was adopted from a preceding work by Ambrož [1].

Automatic cropping

Automatic cropping uses the computed saliency map to find an optimal cropping rectangle for the original image that would contain most of the important (salient) areas while leaving out the non-salient ones. It does this using various algorithms, from which the ones used in this work are briefly explained below. Generally, a brute force or greedy algorithm is used for finding the optimal rectangle. Automatic cropping methods, including the ones explained here were developed for standard 2D image cropping, which means they may suffer on spherical images in equirectangular projection. However, as mentioned before, they can serve as a good baseline.

Suh *et al.* [21] introduced two methods of automatic cropping, from which the first one that is based on saliency map generated by Itti's saliency mapping method is used in this work. Suh's method tries to find an optimal balance between two conflicting conditions – that the cropping rectangle should contain as much salient parts of the image as possible

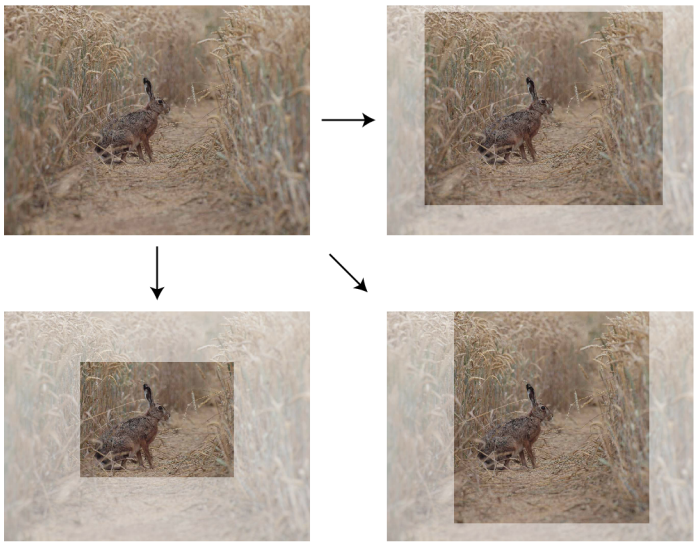


Figure 3.2: Comparison of automatic cropping methods used in this work. Top left is the original image. Top right is a crop created using method by Suh *et al.* [21]. Bottom left is created using method by Stentiford [17]. Bottom right is created using method by Fang *et al.* [6]. All of these crops were generated using implementation by Ambrož [1].

and that it should be as small as possible. It does this using three different algorithms (brute force and greedy with fixed threshold, dynamic threshold) of which the brute force is used in this work since it works reliably and even though it is slower than the greedy or dynamic one, it is still faster than both of the other methods mentioned below. As the name suggests, the brute force algorithm searches through all the possible cropping rectangles that satisfy a fixed threshold of salient pixels inside it and finds the one that has the smallest area.

Automatic cropping method used in the work by Stentiford [17] is similar to the method used by Suh *et al.* [21], however, it computes an average of the saliency values of the pixels inside each cropping rectangle. The saliency map is obtained using a method introduced in the same paper.

Fang *et al.* [6] introduced a more complex cropping method which combines three different factors – content preservation, boundary simplicity and visual composition, to determine the ideal cropping rectangle. It uses Margolin's saliency mapping method to create the saliency map according to which it selects a set of cropping rectangles that contain most of the salient parts of the image – hence content preservation model. It then ranks these candidate cropping rectangles according to boundary simplicity model (which creates an image gradient map that highlights edges and gives higher ranking to rectangles that crop through simpler regions with less edges) and visual composition model (which uses the saliency map to check whether the salient objects are placed in a visually pleasing position in the photo) and finds one with the best combined rank. Although this obviously introduces a lot more computational overhead than the other two methods, it also brings much better results.

Figure 3.2 shows a comparison of these three methods. Their implementation was adopted from a preceding work by Ambrož [1].

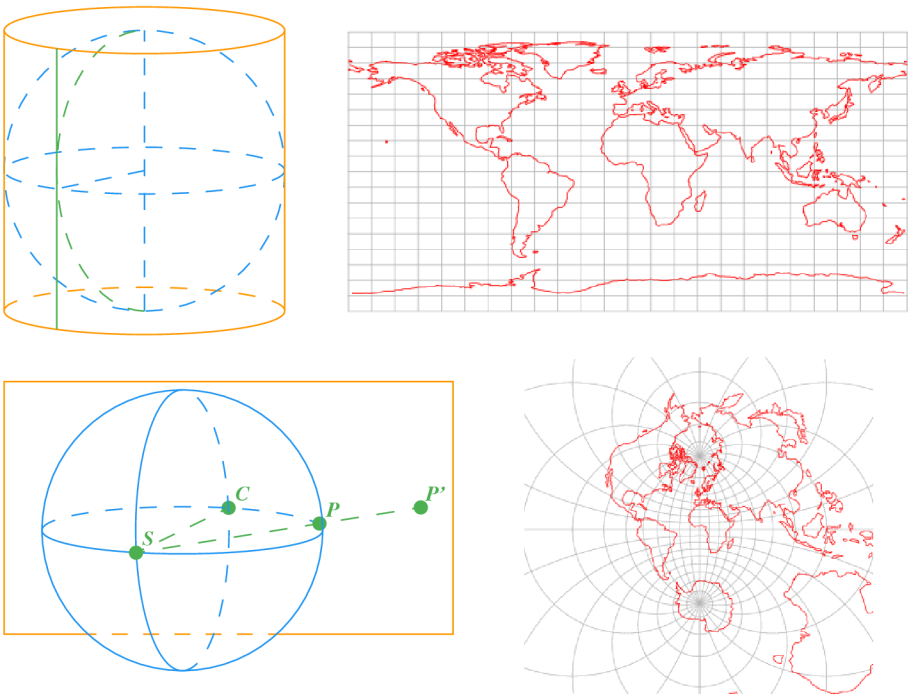


Figure 3.3: Comparison of equirectangular and stereographic projection. The equirectangular projection (top row) simply projects each meridian to a straight vertical line on a tightly wrapped cylinder which is tangent to the meridian at equator. This result in a projection that heavily distorts the top and bottom of the image as can be seen on the top right. Stereographic projection projects each point P on the sphere to the point P' on the plane which is given by the intersection of the plane and a line given by points P and S where S is a point on the sphere exactly opposite to point C which is the center of projection (ϕ_1, λ_0). As the image on bottom right shows, stereographic projection introduces the distortion gradually from the center, equally in every direction. Map projections taken from Wolfram Web Resources [24][25].

Stereographic projection

The last step that completes the saliency baseline method is the projection from equirectangular to stereographic projection [16]. Equirectangular projection is often used for storing spherical images and videos since the spherical image is projected onto a plane which is much easier to store and index than the curved surface of a sphere. The projection itself is also quite simple:

$$x = R(\lambda - \lambda_0)\cos\phi_1 \tag{3.1}$$

$$y = R\phi, \tag{3.2}$$

where R is the radius of the sphere that is projected on the plane (as illustrated in Figure 3.3) and can be calculated from the horizontal width of the plane (image) w as $R = \frac{w}{2\pi}$. This produces an image with a 2:1 aspect ratio, which is usually scaled to a 16:9 ratio which is standard for video.

Stereographic projection projects points of the sphere on a plane as well, however, it does it in a way that creates a much more natural image for viewing purposes. Equirectangular

projection, while undistorted at the horizontal center row of the image or the equator of the plane, gradually stretches the image more and more towards the top and bottom of the image. Stereographic projection distributes the inevitable distortion in a much more natural way for a human eye . It is also the projection of fish eye lenses, which are common for action sports cameras, which is an area where 360° cameras are often used as well.

In order to project from equirectangular to stereographic projection, a center point of the projection (ϕ_1, λ_0) needs to be set. It is possible to calculate it using these equations:

$$\phi_1 = \frac{x_C \cdot 2\pi}{w} \tag{3.3}$$

$$\lambda_0 = \frac{y_C \cdot 2\pi}{h}, \tag{3.4}$$

where (x_C, y_C) is the center of the cropping rectangle and w and h are the width and height of the projection plane (as illustrated in Figure 3.3), respectively. Another variable that needs to be set is the radius of the sphere R which can be calculated using a given horizontal angle of view α:

$$R = \frac{w \cdot (1 + \cos\alpha)}{4 \cdot \sin\alpha}. \tag{3.5}$$

Using R, ϕ_1 and λ_0, a point (x, y) from the projection plane can be mapped on a point (ϕ, λ) on the sphere using:

$$\phi = \arcsin\left(\cos c \sin\phi_1 + \left(\frac{y \sin c \cos\phi_1}{\rho}\right)\right) \tag{3.6}$$

$$\lambda = \begin{cases} \lambda_0 + \arctan\left(\frac{x}{-y}\right), & \text{if } \phi_1 = 90° \\ \lambda_0 + \arctan\left(\frac{x}{y}\right), & \text{if } \phi_1 = -90° \\ \lambda_0 + \arctan\left(\frac{x \sin c}{\rho \cos\phi_1 \cos c - y \sin\phi_1 \sin c}\right), & \text{otherwise} \end{cases} \tag{3.7}$$

where:

$$\rho = \sqrt{x^2 + y^2} \tag{3.8}$$

$$c = 2 \arctan\left(\frac{\rho}{2R}\right). \tag{3.9}$$

If $\rho = 0$, equations 3.6 and 3.7 are indeterminate, but (ϕ, λ) is simply equal to the center of projection (ϕ_1, λ_0).

The computed point (ϕ, λ) can than be mapped onto the equirectangular projection using the the equations 3.1 and 3.2 to find the position of the point on the original image. Since stereographic projection distorts the image in quite a different way than equirectangular does, the final projected image will not contain the exact pixels that the cropping rectangle selected. However, the position as well as the angle of view will stay the same.

3.2 AutoCam: Glimpse-Based Trajectory Selection

As mentioned in Chapter 2, the first work to tackle the task of cropping a spherical video specifically is Pano2Vid by Su *et al.* [20]. This work defines the problem of creating a normal

field of view video from a 360° one in a way that would preserve the important regions in frame while cropping out the unimportant ones and goes on to propose a solution to this problem. From a high level view, the pipeline of the proposed method is as follows. First, the 360° video is sampled to create so called *spatio-temporal glimpses*. These are short clips of the 360° video with normal field of view, sampled at equally spaced out angle distances. The glimpses represent potential views used in the output normal field of view video. Each glimpse is evaluated for its *capture-worthiness* which is a classifier that is learned using a data-driven approach. According to the acquired capture-worthiness scores, an ideal trajectory of the angles is selected and the output video can then be rendered. The follow-up work by Su and Grauman [19] introduced several new features and optimizations for this method. In the following subsections, each of the phases of this method will be explained in detail. This section also explains a modification of this method that uses standard saliency methods for evaluation of the capture-worthiness of the glimpses.

Glimpse generation

The first step of this method generates *spatio-temporal glimpses* which are short 5 second clips sampled from the original 360° video at multiple angles to get a set of normal field of view videos that cover the whole spherical view. To do this, glimpses are sampled quite densely at latitudes $\phi \in \Phi = \{0, \pm 10, \pm 20, \pm 30, \pm 45, \pm 75\}$ and longitudes $\lambda \in \Lambda = \{-180, -160, ..., 140, 160\}$. To get even more versatile results, at each of these coordinates, 3 glimpses with different focal lengths are generated. The focal lengths are simulated as angles of views $a \in A = \{46.4°, 65.5°, 104.3°\}$. Each glimpse can therefore be identified as:

$$\Omega_{t,\phi,\lambda,a} \rightarrow (t, \phi, \lambda, a) \in T \times \Phi \times \Lambda \times A, \tag{3.10}$$

where T is representing set of 5 second time splits of the original video – for example a 30 second video would have 6 splits. Figure 3.4 illustrates the range of glimpses sampled. The sampling is done using stereographic projection explained in detail in section 3.1.

Due to the fact that this process generates a lot of glimpses which introduces a high amount of computational overhead, an optimization technique introduced in the follow-up work by Su and Grauman [19] is used. This technique first generates only the glimpses with wide angle of view ($a = 104.3°$) and only at latitudes $\phi \in \Phi' = \{\pm 75, \pm 30, \pm 10\}$ and longitudes $\lambda \in \Lambda' = \{-160, -120, ..., 120, 160\}$. Longer, 10 second time splits are also used to further speed up the process. Using these coarsely sampled glimpses, the best trajectory is found (explained in detail in later subsection) and only after that are the glimpses sampled densely as explained before. However, only glimpses within a certain spherical distance from the best trajectory are sampled. The maximum distance is given by $\epsilon = 30°$ – for each densely sampled glimpse $\Omega_{t,\phi_g,\lambda_g,a}$ it must hold that:

$$|\phi_g - \phi_t| \leq \epsilon, |\lambda_g - \lambda_t| \leq \epsilon, \tag{3.11}$$

where ϕ_t and λ_t are coordinates of the trajectory at time t. This way, the densely sampled glimpses can be used to refine the trajectory to get the same results while reducing the computational time significantly. The ϵ is used in the trajectory selection described in a later subsection as well.

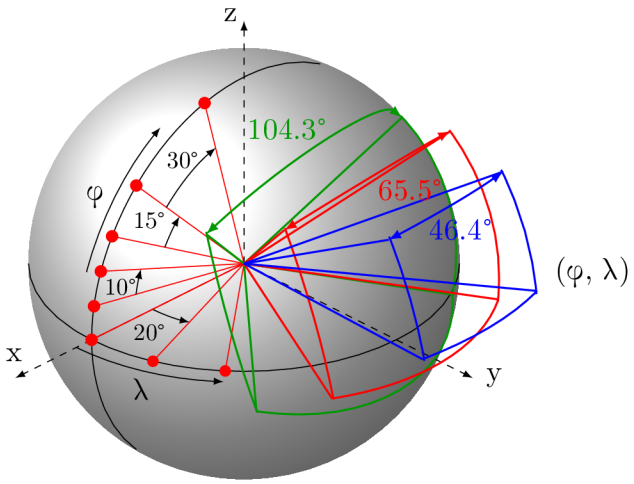


Figure 3.4: Sampling of glimpses by Su and Grauman [19] (edited). The sphere's surface is representing the 360° video. It shows the different angles at which the glimpses are sampled as well as the different angles of view at each position. Each view is projected into stereographic projection, which translates to the curved shape of the views on the sphere.

Glimpse evaluation using C3D features and capture-worthiness

As mentioned before, the glimpses represent potential views of the final normal field of view video. To find the ones that should be chosen to appear in it, it is necessary to find a way to evaluate them. Su *et al.* [20] chose to use a data-driven approach – they collected a dataset of normal field of view videos captured in the same type of environment or with the same topic as the input spherical video to learn a classifier that would score each glimpse based on how similar it is to the normal field of view videos and therefore how likely it is that such a view would be chosen and captured by a human. The underlying assumption is that the normal field of view videos were shot by people who chose to capture interesting parts of their surroundings.

To learn such a classifier, it is useful to create a more compact representation of the glimpses. To do this, C3D features by Tran *et al.* [23] are utilized. C3D features are generic spatio-temporal features obtained using 3-dimensional convolution. They capture the motion and appearance of the glimpses in a simple vector representation which makes them much easier to work with. Thanks to their high accuracy on multiple benchmarks and relatively fast extraction times they are known to be useful in tasks like this which is why they were chosen for this task as well.

After extracting the C3D features of each of the glimpses, a *capture-worthiness* classifier is trained separately for each 360° video. Normal field of view videos from a similar environment serve as positive examples of how the final video should look like while glimpses from other 360° videos serve as negative ones. A logistic regression classifier is trained on a set that contains twice the number of negatives as it does positives. The input spherical video is left out from the training process. Finally, the positive class probability scores of the glimpses sampled from the input 360° video are evaluated and treated as the capture-worthiness of each of them respectively.

The learning process requires quite a few examples of normal field of view videos as well as 360° ones, which need to be collected. Su *et al.* [20] use YouTube as the source of

both the spherical as well as the corresponding 2D videos. They use four keywords (Soccer, Mountain Climbing, Parade and Hiking) to search YouTube and download to top 100 360° and top 2000 normal field of view videos for each of them. Then they hand-pick the ones that have good quality, lighting and generally correspond to the category searched. This yields a relatively large dataset to test this approach on, which they kindly provided me with.

Evaluation using standard saliency mapping

Although C3D features are known for their good performance in the field of recognition and data driven methods nowadays usually outperform algorithmic ones, it is still useful to have a standard method to compare them to. That is why Su *et al.* [20] also used saliency mapping to evaluate the capture-worthiness of the spatio-temporal glimpses. In their approach, a method by Harel *et al.* [7] is used to evaluate the saliency of each glimpse.

The saliency map shows salient regions on the image but it doesn't directly tell how salient, or how interesting the image is overall. There are many approaches that could be used to calculate the overall saliency from the map. The first and most obvious one would be averaging the pixel values across the image, and then across all of the frames of each glimpse. Although this approach might seem logical at first, a very high or very small saliency value of just a small area of pixels can shift the average of the entire image and therefore will not provide a representative value of how salient the image is. Much more suitable for this purpose is the median function, since it filters out any noise and therefore usually provides a much better representative of the saliency values present in the image. Another thing that helps the accuracy of this method is disabling normalization – saliency mapping methods usually normalize the values present in the image to clearly highlight the areas that are more salient than others. This, however, is not desirable in the case of evaluating overall saliency since normalization evens out any big value differences between images. Using the raw values provided by the saliency methods, it is possible to get much bigger differences between the glimpse values which highlights the ones that are the most salient.

This work uses three methods of saliency mentioned in section 3.1 to evaluate the saliency mapping of the glimpses and to evaluate their capture-worthiness from them. This provides a good range of saliency methods to compare to the data driven approach using C3D features. They are also different from the one used by Su *et al.* [20], some older, some newer, and that way this work provides quite a comprehensive look at this capture-worthiness evaluation strategy and makes it a worthy opponent to the data-driven approach.

Trajectory selection

In order to find the trajectory of angles that best fits the score space of the evaluated glimpses, it is necessary to define an algorithm that will search for it. Besides finding the path with best score, the algorithm must ensure that the steps taken will not drastically change the direction the camera is oriented towards, nor the size of the angle of view. This way the final video will be smooth and resemble one that was shot by a human.

In its essence, the algorithm is simply solving a shortest path problem, with a restriction on the range of different steps that can be taken every iteration. The limit of change in both latitude and longitude every step as chosen by Su *et al.* [20] is $\epsilon = 30°$:

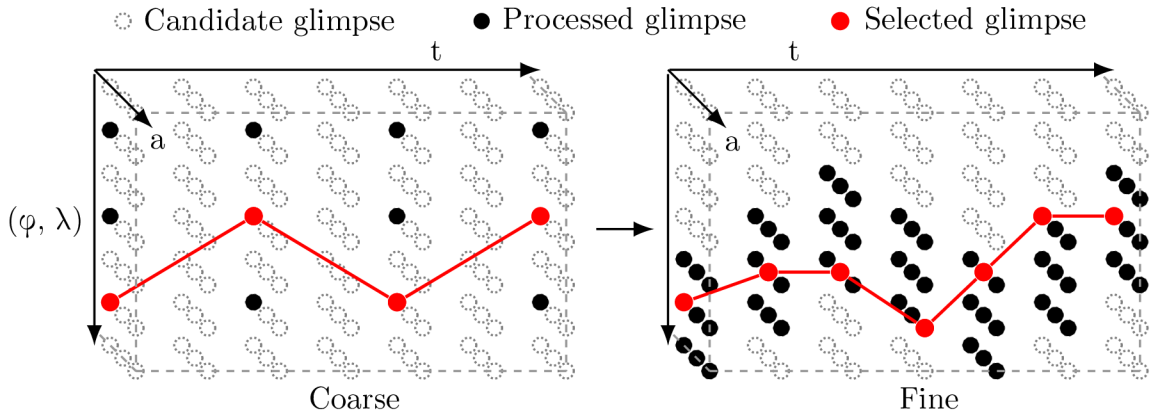


Figure 3.5: Trajectory selection by Su and Grauman [19] (edited). In the first pass, glimpses are sampled only densely at wide angle of view, so that the algorithm can find a rough best path. After that, glimpses are sampled more densely around the selected rough path and with multiple angles of view so that the algorithm can refine the path.

$$|\phi_t - \phi_{t-1}| \leq \epsilon, |\lambda_t - \lambda_{t-1}| \leq \epsilon. \tag{3.12}$$

The limit of change in the angle of view as chosen by Su and Grauman [19] in their follow-up work is constrained to changing from $a_{t-1} = 46.4°$ to $a_t = 65.5°$ and from $a_{t-1} = 65.5°$ to $a_t = 104.3°$ or vice versa. In this work they also introduce the optimization method mentioned before. Since only sparsely sampled glimpses with only a single angle of view are available after the first pass, the algorithm allows a bigger change of 2ϵ in both latitude and longitude at each step. After finding the best path in this sparsely sampled score space, the best path is searched for again using the newly obtained densely sampled one, which is restricted to the glimpses around the previous best path, as explained in a previous subsection. The whole process is illustrated on Figure 3.5.

After finding the refined path, it is a sequence of discrete coordinates $\Omega_{\phi,\lambda,a}$ which denote the optimal direction and angle of view to show from the spherical video at 5 second intervals. These are then linearly interpolated at every frame to get the final continuous trajectory that is used to render the output normal field of view video.

3.3 Automatic Importance Detection

The Automatic Importance Detection algorithm was designed as a part of the work by Pavel *et al.* [14], in which they focused mainly on the experience of watching 360° videos and introduced two novel techniques of playback for spherical videos. Both of these techniques offer reorientation. If the user starts exploring other parts of the video, either via a virtual reality headset, or via clicking and dragging a mouse, it is easy for them to become disoriented, particularly after a new shot appears, which causes them to miss the most important parts. Reorientation tries to solve this problem by reorienting the spherical video so that the important content lies in the viewers field of view based on certain triggers – these differentiate the two techniques. The first technique, *viewpoint-oriented cuts*, reorients the video automatically every time a new shot appears. The second technique, *active*

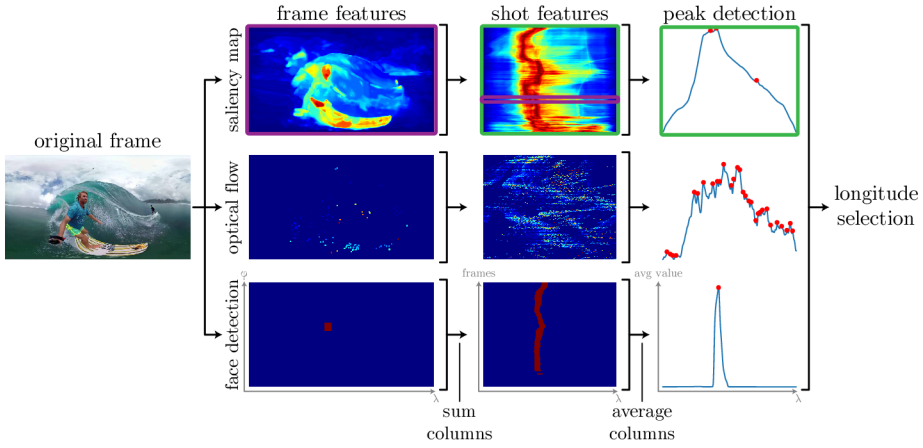


Figure 3.6: Automatic Importance Detection by Pavel *et al.* [14] (edited). The method works by extracting feature maps (b) from each frame (a) of the input video. Each feature map is then summed along its columns to get a feature vector. The acquired vectors are stacked vertically to form a shot feature maps (c). These maps are then averaged along columns to get shot feature vectors (d), on which value peaks are detected – the position of the clearest peak then determines the chosen longitude.

reorientation, allows the user to press a button at any time to automatically reorient the shot. In the paper, Pavel *et al.* go on to test and compare these techniques and find out interesting results. However, they also introduce a method capable of automatically finding important parts of the spherical video, so that they do not need to be defined manually. This method, as it turns out, proves to be useful for the similar task of automatic 360° cinematography as well.

The method processes the frames of the video sequentially, generating three different feature maps for each frame. Each feature map is then summed along the columns to create a feature vector which is then appended to a corresponding *shot feature map*. Each shot of the video therefore ends up with three feature maps, each of them composed of corresponding feature vectors of the frames from the shot stacked vertically. Each shot feature map therefore has a height equal to the number of frames in the shot. Every shot feature map then gets averaged along the columns to get a shot feature vector. Finally, the method finds value peaks in each feature vector and selects the important points accordingly. This process is illustrated in Figure 3.6, and each stage is explained in detail in the following subsections.

Automatic Importance Detection is limited by the fact that it only selects the longitude of the important point in the video – the horizontal coordinate. The latitude (vertical coordinate) is left at zero – the presumption being that 360° videos generally contain the important content around the equator, while areas around poles usually contain sky and ground or ceiling and floor. In the context of the original paper, the viewer can adjust the latitude quite easily and quickly as well. For automatic 360° cinematography, this is not possible, however, the first presumption still holds. Another parameter that Automatic Importance Detection does not set is the field of view – in the context of the original paper this variable is given by the viewing device. In this paper, the field of view is set to be constant at $\alpha = 104.3°$, the same as the wide view of AutoCam.

Feature map extraction

As previously mentioned, Automatic Importance Detection generates feature maps from each frame of the input video. These feature maps are chosen to account for low-level as well as high-level features in the frame. The first feature map is generated using Minimum Barrier Salient Object Detection method by Zhang *et al.* [29], which is a highly efficient and powerful method of saliency mapping based on an approximation of Minimum Barrier Distance Transform by Strand *et al.* [18]. When compared to many state of the art methods, Minimum Barrier Salient Object Detection is much more efficient, yet its performance is the same or better. This makes it a good candidate for low-level feature extraction.

Since the method is dealing with videos, the temporal aspect should be leveraged as well. Moving objects in videos are usually of high interest to the viewer, and that is why the second feature map is generated using Lucas-Kanade optical flow by Baker and Matthews [2]. To track objects across the frames, we first need to choose which points should be tracked. Good points to track are ones that are visually significant so that they can be reliably identified on the subsequent frames of the video. To find such points, the Harris Corner Detector [8] is utilized. The resulting set of points to track is recalculated every 0.3 seconds, as optical flow will lose tracked points as they go past the boundaries of the frame (note that in the context of spherical videos, the point simply appears on the opposite side of the frame, however, this is not something that optical flow methods account for). The feature map is then generated by setting the value of pixels at positions given by the tracked points to the distance travelled by that point since the last frame. The surrounding area of that pixel (within 5 pixels) is set to that value as well, to account for minor inaccuracy of the tracking as well as making the point more prominent.

To account for higher-level, more semantic features as well, face detection is used to generate the third feature map. First, faces in the frame are detected using the OpenFace toolkit by Baltrušaitis *et al.* [3]. A feature map is generated by setting values in the area given by the bounding box of each detected face to the confidence score assigned to that face. Finally, each one of the mentioned feature maps is normalized after being summed along columns to create feature vectors which are then appended to the shot feature maps as explained in the previous subsection.

Shot detection

While optical flow is useful to detect moving objects in the video, they can also be used to detect shot boundaries. Movement between subsequent frames within a shot is usually small, however, optical flow calculation on boundary frames of two different shots results in improbably large movement of points ($> 50\%$ of frame diagonal) or many lost points (more than half). This is a strong evidence, that a new shot has occurred and if the movement of points exceeds this threshold, the new frame is considered as the start of a new shot. The shot feature maps accumulated so far are processed as explained in the following subsection before the algorithm continues in processing subsequent frames.

Longitude selection

To select the resulting longitude, shot feature maps are first averaged along the columns to get shot feature vectors which are then normalized and smoothed using a Gaussian kernel. The feature vector is then searched for local maximums (peaks) and the horizontal location of the clearest peak determines the selected longitude (the range of longitudes is mapped

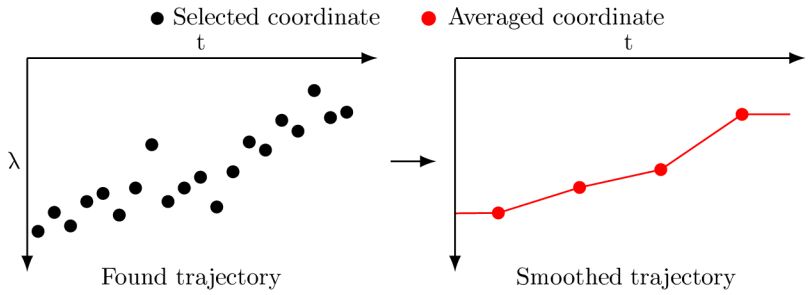


Figure 3.7: In order to prevent shaky output video, the trajectory found by the continuous approach must be smoothed by averaging groups of coordinates spanning 5 seconds and linearly interpolating between these averaged coordinates.

on the width of the frame which is in standard equirectangular projection). The clarity of a peak is computed as $\frac{S_P}{S_C}$ where S_P is the area under the peak (boundaries are determined by the surrounding local minimums) and S_C is the area under the entire curve – in the context of the feature vector this means sum of all its values. If the clarity of the clearest peaks from the feature vectors is similar, the peak from a feature with higher semantic value is selected, e.g. peaks from face detection have higher priority than peaks from saliency mapping. If none of the peaks have at least a clarity of 0.2, the origin (middle of the frame) is selected.

Continuous approach

While the original method works very well for edited spherical videos with multiple shots, it suffers when it comes to long unedited recording from a 360° camera since there is only a single shot and the method only selects a single view for the entire video. For this use case, this work introduces a slight modification of the original approach that creates a continuous trajectory, adjusting the longitude for each frame. The approach works by eliminating the phase of creating shot features from the process – instead of summing the frame feature maps to create feature vectors that would be appended to shot feature maps, they are averaged along the columns and directly searched for peaks as if they were the complete shot feature maps. In other words, each frame is considered its own shot, with the exception of not resetting the optical flow – points to track are still recalculated every 0.3 seconds, however, since every frame is essentially a new shot, shot detection as explained in a previous subsection is not performed.

Evaluating the longitude for each frame, however, may result in shaky output video, as the selected longitude can change from frame to frame. This can be solved by smoothing the resulting trajectory before using it to render the output video. First, the trajectory is divided into sections spanning 5 seconds of the output video and coordinates from each section are averaged together to get a single coordinate for that portion of the video. Then, similarly to the technique used for creating a smooth trajectory from discrete points in the AutoCam method by Su *et al.* [20], linear interpolation is used to get smooth transition from one coordinate to another. Figure 3.7 illustrates the process of smoothing the trajectory.

Chapter 4

Implementation Details

This chapter discusses the details of implementation of the methods explained in the previous chapter. First, a high-level overview of the design of the implemented application is provided. Next, details about a notable optimization technique used to speed up the rendering of glimpses is discussed. Finally, this chapter concludes with an overview of the utilized technologies and libraries.

4.1 Design of Application for Spherical Video Cropping

The application is structured into multiple classes, each responsible for a specific part of one of the methods, with several classes handling shared functionality. Figure 4.1 shows a diagram illustrating the data flow between them. When the application launches, it creates an instance of the `Logger` class which handles logging and printing to standard output. A reference for this instance is passed to almost every other class as an argument. This way, all text output can be done by calling the appropriate method of the `Logger` instance, while simultaneously being logged to a backup log file, which can be later used to debug any errors that might arise. Next, the application creates an instance of the `ArgParse` class which parses arguments and makes them easily available to any other class. It also searches through the input folder and finds all of the supported spherical videos.

Video processing and output tasks are handled by the `Renderer` class, instance of which is also shared throughout the run time of the application (note that if OpenMP parallel processing is available and the user does not override it using the `-t` parameter, an instance of `Renderer` is created for each thread separately). For easy accessibility, information about videos are held inside `VideoInfo` structures, which serve as a kind of internal representation of any video and practically every video the application deals with has a corresponding instance. The common task of all of the methods is to find a trajectory of view coordinates according to which the spherical video should be cropped. The last instance that needs to be created before launching any of the three methods is one of the `Trajectory` class which holds a sequence of coordinates specifying longitude, latitude and field of view at each frame of the output video. After the trajectory is evaluated using any of the methods, the output video can be created using the `Renderer`.

The Saliency Baseline (using automatic cropping by Suh *et al.* [21], Stentiford [17] and Fang *et al.* [6]) method and Automatic Importance Detection by method by Pavel *et al.* [14] described in Sections 3.1 and 3.3 are each handled in a single class – `AutoCrop` and `AID`, respectively. They both expose a `findTrajectory()` method which, as the name suggests,

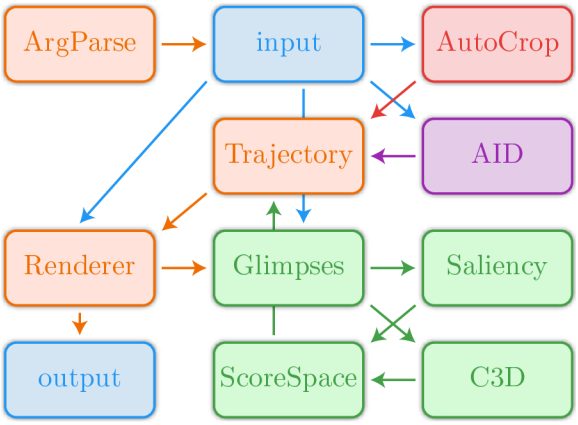


Figure 4.1: Diagram of data flow between the main classes of the application. The `Logger` class is omitted since it is used in all parts of the application. The input and output as well as any video is represented in the application by a `VideoInfo` structure (colored blue in the diagram). Note that only one of the methods is used to evaluate the video – this choice is handled by the main function using information provided by an `ArgParse` instance. Classes corresponding to different methods are distinguished by color.

finds the cropping trajectory using the corresponding method. The trajectory can then be optionally smoothed for both of these methods, as well as for the AutoCam method by Su and Grauman [19], for which the evaluation is slightly more complex. Firstly, the spatio-temporal glimpses are handled and organized using an instance of the `Glimpses` class, which provides functionality to render coarsely or densely sampled glimpses. Note that the rendering itself is handled in the `Renderer` class, `Glimpses` class only organizes the glimpses and provides a layer of abstraction above the rendering. The glimpses are then evaluated using either the `C3D` class, which evaluates them using the original approach of extracting C3D features and then using logistic regression to score the glimpses, or the `Saliency` class, which uses the saliency mapping evaluation, all of which has been detailed in Section 3.2. The scores are stored in an instance of the `ScoreSpace` class, which then provides functionality to find the trajectory corresponding to the highest score path in the space.

4.2 Optimization of Rendering Using Displacement Maps

A crucial part of all three of the implemented methods is stereographic projection as explained in Section 3.1. The mapping from equirectangular to stereographic projection is quite computationally expensive and this is has a significant impact especially on the performance of the AutoCam method by Su and Grauman [19] since it needs to generate a substantial amount of stereographically projected spatio-temporal glimpses. Each glimpse, however, has a given spherical coordinate and field of view, which stays constant throughout its duration – this means that the mapping from one projection to another stays the same as well and therefore it only needs to be computed once and then reused for all remaining frames of the glimpse.

To do this, the mapping, instead of being directly applied to the frame, is instead computed into two *displacement maps* – arrays of size equal to the size of input frames, which contain the x and y positions in the input frame that correspond to the output at the

given pixel. In other words, the displacement maps define the output image using positions in the input image. From the nature of the mapping it is clear that the computed positions will not be integers most of the time, and therefore the output values need to be computed using bilinear interpolation between the closest pixels in the input frame. Computational inaccuracies can also result in positions out of bounds of the input frame, which is solved by wrapping the positions around using a modulo operator. The displacement maps can then be used to compute all of the frames of the corresponding glimpse.

Moreover, the AutoCam method in the coarse search phase generates the same set of glimpses for each split, which means that the displacement maps can be reused for all glimpses at the corresponding coordinates throughout all of the splits of the input video. This optimization technique significantly computation complexity of the AutoCam method, making it perform as fast as the variant of the saliency baseline method using automatic cropping method by Suh *et al.* [21], which is the fastest variant.

4.3 Utilized Technologies

The application was written in the C++ language using the C++17 standard. This language was chosen because of its high speed compared to other popular languages like Java or Python, while still offering object-oriented programming and therefore a good level of abstraction. Most of the application utilizes the OpenCV library[1], which provides interfaces for reading and writing video, accessing individual frames and processing them either using high-level functions like image remapping or via direct access to individual pixel values for custom filtering. The application requires OpenCV of at least version 4.0 – most of the code is compatible even with older versions since 3.0, however version 4.0 is required by another dependency this application has, which is the OpenFace toolkit.

The OpenFace toolkit version 2.0[2] by Baltrušaitis *et al.* [3] was briefly mentioned in Section 3.3 and in this application, only its landmark detection library is utilized to detect faces in the frame. Although this toolkit offers state-of-the-art face detection, it also has quite a few dependencies – besides the previously mentioned OpenCV library of at least version 4.0, it also needs OpenBLAS library[3] and dlib library[4]. All of these dependencies are, however, usually quite easy to install thanks to package managers. At the time of the writing this thesis, however, the OpenFace toolkit relied on a few recently added functions of the dlib library which were not yet available in the pre-built binaries available via the package managers. For this reason, a recent version of the dlib library source code is included in the application. The implementation of Minimum Barrier Saliency[5] by Zhang *et al.* [29] directly from the authors of the corresponding paper is included in the source code as well, and the Lucas-Kanade optical flow method by Baker and Matthews [2] as well as Harris Corner Detector [8] are both implemented in the OpenCV library. This completes the dependencies of the Automatic Importance Detection method by Pavel *et al.* [14].

The AutoCam method by Su *et al.* [20] has the most tedious dependencies to configure since it relies C3D features by Tran *et al.* [23] which are built on top of an outdated version

[1] https://opencv.org/
[2] https://github.com/TadasBaltrusaitis/OpenFace
[3] https://www.openblas.net/
[4] http://dlib.net/
[5] https://github.com/jimmie33/MBS

of the Caffe framework[6]. The implementation of C3D features is available on GitHub[7] and a successful building of version 1.0 is a requirement for the AutoCam method to work. This is hard if not impossible to make automatic for a majority of platforms, since it relies heavily on the available GPU and libraries on the system. Therefore, the C3D source code is not included in the application, instead, it is necessary to build it separately and then place the binary in a specific location. This is explained in more detail in the `README.md` file included in the source code. Everything else in this method, like for example glimpse generation as well as logistic regression is handled by the application itself, while utilizing the OpenCV library.

The Saliency Baseline method explained in Section 3.1 utilizes the implementation of various saliency mapping and automatic cropping methods implemented by Ambrož [1]. This implementation is included in the source code. Besides OpenCV library, it relies on the VLFeat library[8], which is also included in the source code.

To build the entire application, CMake[9] of at least version 3.0 is necessary. This makes the build process much more organized and adaptable to multiple platforms and library install locations. As mentioned previously, the application uses features of the C++17 standard and therefore a compiler with support for C++17 is required as well.

[6] http://caffe.berkeleyvision.org/
[7] https://github.com/facebookarchive/C3D
[8] https://www.vlfeat.org/
[9] https://cmake.org/

Chapter 5

Experimentation

This chapter describes the process of comparing the implemented methods and discusses the obtained results. The method chosen for experimentation is based on the Perceptual Evaluation of Color-to-Grayscale Image Conversions by Čadík [31], which utilizes the two-alternative forced choice (2AFC) method by David [4]. This method measures the subjective experience of a person through the patter of their choices. The subject is presented with two options at the same time, of which it selects the better one according to their subjective opinion. With enough different test subjects the results can be considered objective. To evaluate this experiment, the data is converted into z-score scales according to the Law of Comparative Judgements, Case V by Thurstone [22]. This does not assign absolute ratings to the methods, the assigned values are relative to each other. This is, however, all that is needed to evaluate how the methods stack up to each other. The results are compared on multiple categories and the results are discussed in detail.

5.1 Experimental Setup

To prepare for the experiment, results of the methods to be compared first needed to be generated. The Pano2Vid dataset by Su *et al.* [20] (mentioned in Section 3.2) was chosen as the input for the methods, because it contains a wide range of different 360° videos and it provides the categories needed for their AutoCam method. Some of the videos in this dataset were too long for a subject to watch without losing concentration. These videos were split into multiple parts no longer than 6 minutes. As most of the longer videos were also highly repetitive throughout their runtime (i.e. soccer match), some of the parts they were split into were removed from the dataset. After these alterations the dataset yielded a total of 96 videos in 4 categories.

As there were 9 variations of the three main methods in total described in Chapter 3, comparing all of them using the 2AFC method by David [4] (detailed below) on this relatively big dataset would require $\binom{9}{2} \cdot 96 = 3456$ comparisons to have at least one result for each match. Since this was beyond the available possibilities, 5 variations were chosen based on a subjective view of their performance on a few testing videos. The saliency mapping evaluation variations of the AutoCam method by Su *et al.* [20] did not perform well and they were therefore cut from the comparison. Out of the baseline method variations, the one based on automatic cropping by Stentiford [17] performed the worst and it was not included in the comparison either. This yielded 5 variations of the three main methods that were chosen for the comparison – the baseline method using the variations based on automatic

cropping by Suh *et al.* [21] and Fang *et al.* [6], the improved data-driven AutoCam method by Su and Grauman [19] and the original shot-based Automatic Importance Detection method by Pavel *et al.* [14] as well as its continuous variation.

To compare the 5 chosen methods for spherical video cropping, the 2AFC comparison method by David [4] was chosen. This method is widely used for measuring subjective experiences – it is therefore ideal for the use-case of selecting the „better“ crop of the spherical video since this quality is purely subjective. In the context of this experiment, the test subject is presented with two videos – outputs of two different cropping methods from the same input video. The subject views the videos, either one after another or simultaneously, and selects the one they consider to be better by clicking a button below it. When the amount of different test subjects is high enough, their subjective opinions blend together to form an objective view. An important thing to note is that the subjects are not presented with the input video, since the way it is viewed (2D screen or a VR headset, equirectangular or stereographic projection) might affect their decision.

In practice, the experiment was performed online via a website that displayed a pair of output videos (as explained in the previous paragraph) and two buttons to choose the better one. A list of all possible pairs ($\binom{5}{2} \cdot 96 = 960$ items long) was first generated and randomly shuffled. The website then always showed a pair at the current index in the list, which was incremented every time a test subject compared a pair of videos. This ensured that each pair was rated at most once more or once less than any other pair while keeping the succession of displayed videos pseudo-random. The test subjects compared the videos at their time of convenience and they were free to compare more or less videos.

The resulting data was then converted into 5×5 frequency matrices in which the value in row i and column j is equal to the number of comparisons where the output of method i was chosen over the output of method j. A frequency matrix accounting for all of the comparisons as well as matrices comprised only of comparison results of videos from specific category were generated. This is due to the fact that the content of the videos varies widely between the categories and the preferable cropping method might be completely different. These frequency matrices were then converted into z-score scales according to the Law of Comparative Judgements, Case V by Thurstone [22]. This was done using a framework by Perez-Ortiz and Mantiuk [15], which also accounts for the fact that some pairs can be evaluated more often than others and weighs the scores accordingly. The resulting scales put the performance of the methods into perspective. While it does not rate the methods absolutely, it does assign them a relative score. With this, the methods can be ordered from best to worst and compare how much better or worse each method is in terms of relative improvement when compared to other methods.

5.2 Results

After two weeks of testing, 66 different subjects performed a total of 1179 comparisons. This means that each possible pair has been evaluated at least once. Using the techniques mentioned before, the data was converted into z-score scales for each category of videos. The results are shown on Figure 5.1. The first thing that becomes obvious from the graph is that the AutoCam method by Su and Grauman [19] performed quite poorly when compared to all of the other methods. This is most probably caused by the fact that this method tends to crop the input spherical videos quite heavily. If the method fails to identify the most important subject or area in the video, even if the error in coordinates is small, the cropped video might completely miss the interesting content. This shortcoming is most

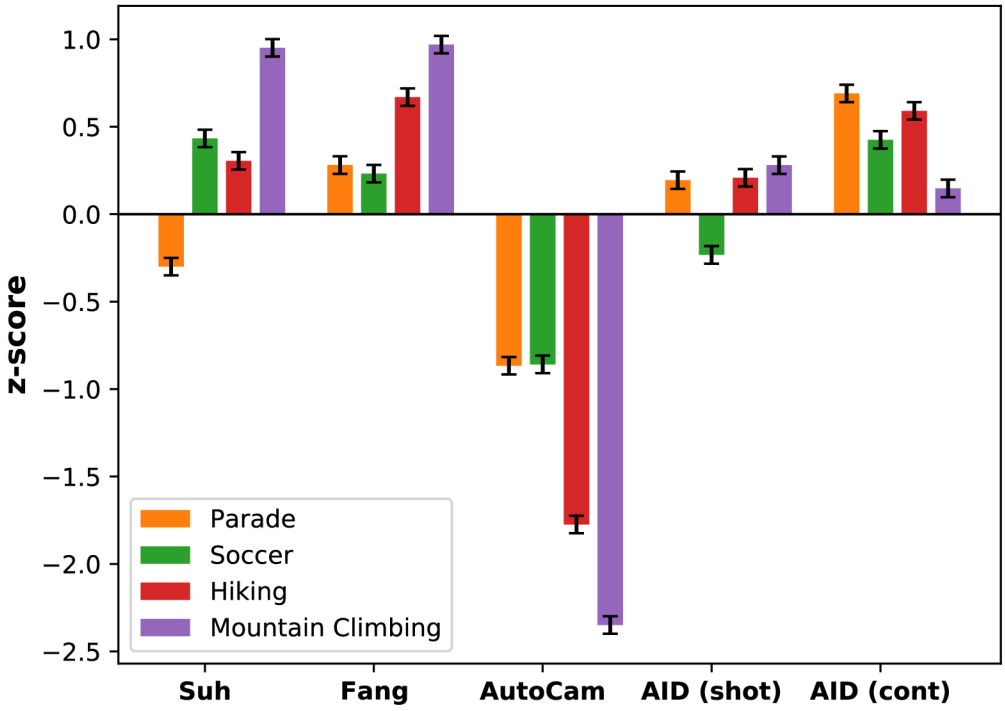


Figure 5.1: Obtained z-scores for individual categories of the videos for each of the methods. Error bars show intervals of 95% confidence.

prominent in the hiking and mountain climbing categories. These videos are usually filmed hand-held, with some of the objects of interest quite close to the camera and therefore covering a large portion of the frame. Even if the method correctly identifies the point of interest, the viewer can often get disoriented since only a small part of it is visible while the surroundings that often give invaluable context in these scenarios get cropped out.

The hiking and especially mountain climbing categories coincidentally perform best with the Saliency Baseline methods utilizing automatic cropping by Suh *et al.* [21] and Fang *et al.* [6]. This is due to the tendency of these methods to choose an extremely wide crop (more than 180°). Since these methods were originally designed for general 2D photos, which usually have a clear subject or a small number of them, this is understandable. A frame of a 360° video usually contains many salient objects and these methods therefore tend to try to keep most of them in the crop. This results in extremely warped output videos, as they get transformed to the stereographic projection. While keeping most of the content from the input spherical video inside the crop might be considered a good property, the warp it introduces can be quite unnatural to watch for some viewers. Moreover, the wide crop can become a disadvantage when the interesting objects are far from the camera and cover only a small portion of the frame. This is most prominent in the parade and soccer categories. While a wide crop can ensure that almost the entire soccer field or street on which the parade is held is visible, it would be preferable to have a tighter crop on the area of the soccer field where the players are or the currently passing car on the parade. This would allow the viewer to see the details that are currently important much more clearly.

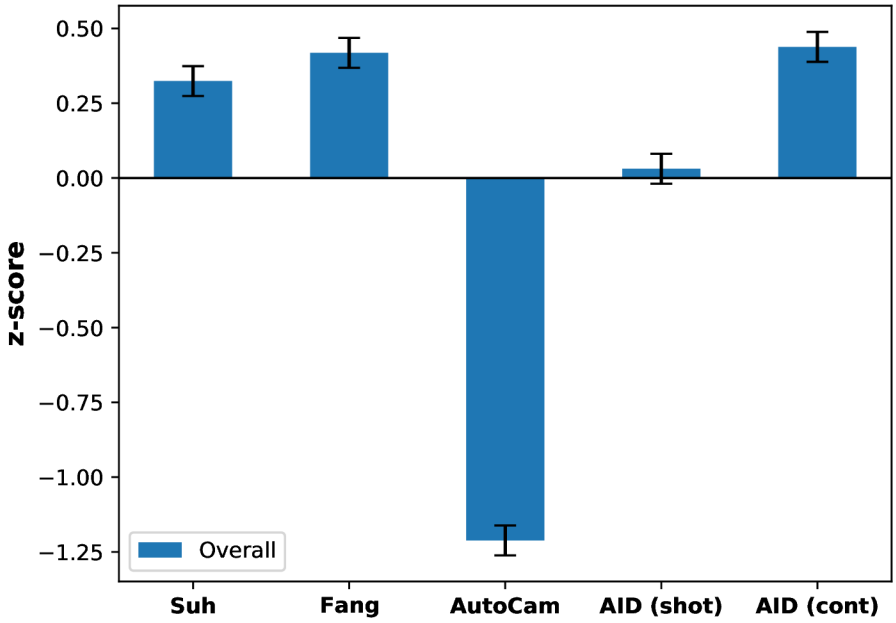


Figure 5.2: Overall z-scores obtained by each method. Error bars show intervals of 95% confidence.

This is where the Automatic Importance Detection method by Pavel *et al.* [14] and the continuous variation introduced in Section 3.3 strike a balance between the wide and narrow crop, since the crop is fixed at 104.3°. The utilization of high level features like face detection seems to also be a step in the right direction for most videos, however, it also brings with it a problematic case when the person filming the video is either holding the camera or standing very close to the tripod it stands on. In these cases the method identifies the face of that person as the most important object in the video and the crop is focused on them. This might be a welcome feature in some cases, like for example if the person is talking to the camera, however, in other scenarios the person might just be standing next to the camera and the crop should be facing a completely different direction.

When comparing the methods on the entire set of comparisons, the continuous variation of the Automatic Importance Detection method by Pavel *et al.* [14] scored the highest, with a minor, statistically insignificant advantage over the Saliency Baseline method using the automatic cropping method by Fang *et al.* [6]. This method had a slight advantage over the one by Suh *et al.* [21], most probably thanks to its more modern and versatile saliency mapping and cropping algorithms. The original Automatic Importance Detection method by Pavel *et al.* [14] performed notably worse than both of the tested Saliency Baseline method variants. This is most probably due to the nature of the tested videos. Most of them are not edited and therefore they contain only a single shot spanning the entire video. This is something the original method was not designed for – it only selects a single direction the crop is facing the entire video which means many important objects in the video might move out of the frame for a significant time span of the video. The technique of detecting the important areas in the frame, however, seems to be highly effective, since the continuous variation of this method performed significantly better. The AutoCam method by Su and Grauman [19] performed much worse than all of the other methods in all of the categories due to the reasons discussed earlier in this section and the overall score is

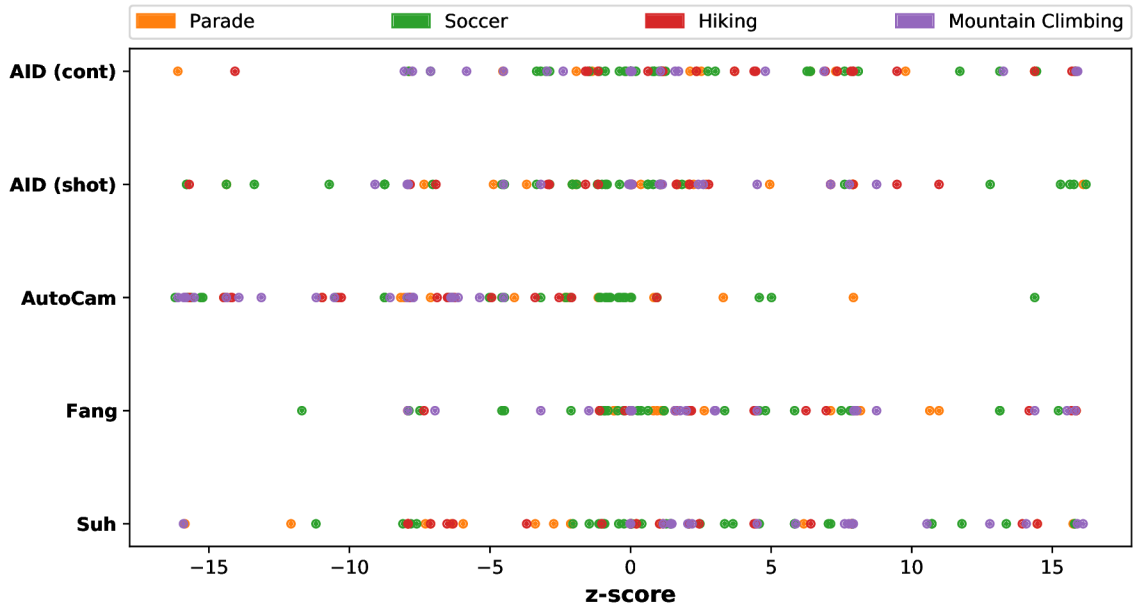


Figure 5.3: Obtained z-scores of each of the methods for individual videos. Each point represents a video, its color denotes the category it belongs to and its position to the respective method and obtained z-score.

Method	Suh	Fang	AutoCam	AID (shot)	AID (cont)
FPS	0.38633	0.32962	0.80518	**3.15657**	3.02745

Table 5.1: Average speed of evaluation for each of the tested methods.

therefore the lowest as well. Figure 5.2 shows a comparison of the obtained overall scores of each of the methods.

As discussed earlier in this section, the content of the video matters greatly when it comes to the performance of each of the methods – the scores of the methods vary widely between the categories. Figure 5.3 shows a visualization of the obtained scores for each of the individual test videos separately. The graph confirms that the differences between the top 2 methods is indeed minimal, while the other methods are separated more visibly with the AutoCam method by Su and Grauman [19] trailing behind the other methods by a significant amount. It also shows that none of the methods was able to steadily outperform all of the other methods in any of the categories. This supports the notion that the performance of each of these methods is highly dependent on the specific video they are applied to.

Another thing to consider when comparing these methods is speed of evaluation. Table 5.1 shows a comparison of average performance of these methods in terms of speed. The Automatic Importance Detection method by Pavel *et al.* [14] and the continuous variant clearly lead in this regard with both of them achieving comparable results of around 3 frames per second. The AutoCam method by Su and Grauman [19] comes in at third place with around 0.8 frames per second, mainly thanks to the optimization of the glimpse rendering process. The Saliency Baseline method variants achieved the slowest performance of around a 3 seconds per frame due to their relatively slow saliency mapping techniques. The variant based on automatic cropping by Suh *et al.* [21] achieved slightly faster performance

than the one based on the work by Fang *et al.* [6], however, that is balanced out by a slightly worse quality of its results (as shown in Figure 5.2).

All in all, the continuous variant of the Automatic Importance Detection method by Pavel *et al.* [14] introduced in Section 3.3 offers the best balance of quality, speed and crop width of all of the tested methods. While there were a number of cases where the subjects preferred the results from the Saliency Baseline method variants, their performance in terms of speed was almost 10 times slower than the Automatic Importance Detection variants. The AutoCam method by Su and Grauman [19] offered a balanced performance in terms of speed, however, preference of its outputs among the viewers was significantly lower than any of the other methods. The obtained results signify that the approach taken by Pavel *et al.* [14] is a step in the right direction, however, more research needs to be done in this area in order to widen the gap between the traditional automatic cropping methods and methods specially designed for the task of cropping spherical videos. For several image results from the researched algorithms, see Appendix A.

Chapter 6

Conclusion

This work has provided a research of the recent advancements in the area of automatic spherical video cropping and detailed 3 methods for solving this task. The first one is a Saliency Baseline method that utilizes algorithms from the well-researched area of automatic image cropping by Suh *et al.* [21], Stentiford [17] and Fang *et al.* [6]. The second one is AutoCam by Su *et al.* [20, 19], which is a novel method that uses a data-driven approach to this problem. The third method is based on Automatic Importance Detection by Pavel *et al.* [14]. While the original algorithm was designed for a slightly different task, it proved to be useful for automatic 360°cinematography as well. A variation of the AutoCam method using saliency mapping and a variation of the Automatic Importance Detection method with continuous movement of the camera were also introduced.

The methods were then implemented in the C++ language using multiple libraries, most notably the OpenCV library (see Section 4.3). A selection of the method variants then underwent a thorough experimentation using a pairwise comparison test that was held online. In the experiment, the respondent was presented with a pair of output 2D videos from two randomly chosen methods, which were generated from the same spherical input video. The task of the respondents was to watch the two videos and choose the better one based on their subjective experience.

The experimentation brought several interesting and unexpected results. While the continuous variant of the Automatic Importance Detection method by Pavel *et al.* [14] scored the highest overall, the difference when compared to the Saliency Baseline method utilizing automatic cropping by Fang *et al.* [6] was statistically insignificant. Even when using the older automatic cropping algorithm by Suh *et al.* [21], the Saliency Baseline method did really good in the testing. The AutoCam method by Su *et al.* [20, 19] performed significantly worse than any of the other methods. A full analysis of the results on different categories and the possible reasons for these results are discussed in detail in Section 5.2.

The results show that the current methods designed specifically for spherical video cropping are not able to steadily outperform the traditional automatic image cropping methods yet and more research in this field is still needed. A possible improvement of the Automatic Importance Detection could be augmenting it to choose a vertical spherical coordinate and field of view besides the horizontal coordinate. This could improve the variability of the output and it could also adapt to a wider range of videos.

There are many variables to consider when designing an algorithm for automatic spherical video cropping and one of the toughest to address is the broad range of possible content of the videos. This makes it exceptionally challenging to design a method that would work in all circumstances. This work researched two major approaches to this problem – a

data-driven approach and a traditional algorithmic approach. From what the results of the experiments have shown, the algorithmic approach seems to be the more promising one.

While working on this thesis, I have broadened my knowledge in the field of spherical video cropping as well as many underlying areas that were required to be studied before researching this specific topic. While implementing the chosen methods, I have faced many challenges, overcoming of which has further improved my skills in the C++ language. The process of experimentation performed on the implemented methods has taught me many good practices to use for user testing as well as the methods that can be used for evaluating algorithms of this kind.

Bibliography

[1] Ambrož, V.: *Algoritmy pro Automatický Ořez Fotografií.* Bachelor's thesis. Vysoké učení technické v Brně, Fakulta informačních technologií. 2018.

[2] Baker, S.; Matthews, I.: *Lucas-Kanade 20 Years On: A Unifying Framework. International Journal of Computer Vision.* vol. 56, no. 3. 2004: pp. 221–255.

[3] Baltrusaitis, T.; Zadeh, A.; Lim, Y. C.; et al.: *OpenFace 2.0: Facial Behavior Analysis Toolkit.* In *13th IEEE International Conference on Automatic Face & Gesture Recognition.* IEEE. 2018. pp. 59–66.

[4] David, H. A.: *The method of paired comparisons.* vol. 12. Oxford University Press. second edition. 1988.

[5] Elson, D.; Riedl, M.: *A Lightweight Intelligent Virtual Cinematography System for Machinima Production.* AAAI. 2007. pp. 8–13.

[6] Fang, C.; Lin, Z.; Měch, R.; et al.: *Automatic Image Cropping using Visual Composition, Boundary Simplicity and Content Preservation Models.* In *Proceedings of the 22nd ACM International Conference on Multimedia.* ACM. 2014. pp. 1105–1108.

[7] Harel, J.; Koch, C.; Perona, P.: *Graph-Based Visual Saliency.* In *Advances in neural information processing systems.* 2007. pp. 545–552.

[8] Harris, C. G.; Stephens, M.: *A Combined Corner and Edge Detector.* In *Alvey Vision Conference*, vol. 4. 1988. pp. 147–151.

[9] Hu, H.-N.; Lin, Y.-C.; Liu, M.-Y.; et al.: *Deep 360 Pilot: Learning a Deep Agent for Piloting through 360° Sports Videos.* In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* IEEE. 2017. pp. 3451–3460.

[10] Itti, L.; Koch, C.; Niebur, E.: *A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. IEEE Transactions on Pattern Analysis & Machine Intelligence.* vol. 20, no. 11. 12 1998: pp. 1254–1259. doi:10.1109/34.730558.

[11] Kim, W.; Kim, C.: *Spatiotemporal Saliency Detection Using Textural Contrast and Its Applications. IEEE Transactions On Circuits And Systems For Video Technology.* vol. 24, no. 4. 2014: pp. 646–659. ISSN 1051-8215.

[12] Lee, S.; Sung, J.; Yu, Y.; et al.: *A Memory Network Approach for Story-based Temporal Summarization of 360° Videos.* In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* IEEE. 2018. pp. 1410–1419.

[13] Margolin, R.; Tal, A.; Zelnik-Manor, L.: *What Makes a Patch Distinct?* In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2013. pp. 1139–1146.

[14] Pavel, A.; Hartmann, B.; Agrawala, M.: *Shot orientation controls for interactive cinematography with 360 video*. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM. 2017. pp. 289–297.

[15] Perez-Ortiz, M.; Mantiuk, R. K.: *A practical guide and software for analysing pairwise comparison experiments. arXiv Stat.AP*. 2017.

[16] Snyder, J. P.: *Map Projections – A Working Manual*. US Government Printing Office. 1987.

[17] Stentiford, F.: *Attention Based Auto Image Cropping*. In *Proceedings of the International Conference on Computer Vision Systems*. ICVS. 2007.

[18] Strand, R.; Ciesielski, K. C.; Malmberg, F.; et al.: *The Minimum Barrier Distance. Computer Vision and Image Understanding*. vol. 117, no. 4. 2013: pp. 429–437.

[19] Su, Y.-C.; Grauman, K.: *Making 360° Video Watchable in 2D: Learning Videography for Click Free Viewing*. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017. pp. 1368–1376.

[20] Su, Y.-C.; Jayaraman, D.; Grauman, K.: *Pano2Vid: Automatic Cinematography for Watching 360° Videos*. In *Asian Conference on Computer Vision*. Springer. 2016. pp. 154–171.

[21] Suh, B.; Ling, H.; Bederson, B. B.; et al.: *Automatic Thumbnail Cropping and its Effectiveness*. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*. ACM. 2003. pp. 95–104.

[22] Thurstone, L. L.: *A Law of Comparative Judgment. Psychological review*. vol. 34, no. 4. 1927: page 273.

[23] Tran, D.; Bourdev, L.; Fergus, R.; et al.: *Learning Spatiotemporal Features with 3D Convolutional Networks*. In *Proceedings of the IEEE international conference on computer vision*. IEEE. 2015. pp. 4489–4497.

[24] Weisstein, E. W.: *Equirectangular Projection*. From *MathWorld* – A Wolfram Web Resource. [Online; accessed 30.12.2019].
Retrieved from:
http://mathworld.wolfram.com/EquirectangularProjection.html

[25] Weisstein, E. W.: *Stereographic Projection*. From *MathWorld* – A Wolfram Web Resource. [Online; accessed 30.12.2019].
Retrieved from: http://mathworld.wolfram.com/StereographicProjection.html

[26] Wolf, L.; Guttmann, M.; Cohen-Or, D.: *Non-homogeneous Content-driven Video-retargeting*. In *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007. pp. 1–6.

[27] Yan, B.; Sun, K.; Liu, L.: *Matching-Area-Based Seam Carving for Video Retargeting. IEEE Transactions on Circuits and Systems for Video Technology.* vol. 23, no. 2. 2012: pp. 302–310.

[28] Yan, J.; Lin, S.; Bing Kang, S.; et al.: *Learning the Change for Automatic Image Cropping.* In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* IEEE. 2013. pp. 971–978.

[29] Zhang, J.; Sclaroff, S.; Lin, Z.; et al.: *Minimum Barrier Salient Object Detection at 80 FPS.* In *Proceedings of the IEEE International Conference on Computer Vision.* IEEE. 2015. pp. 1404–1412.

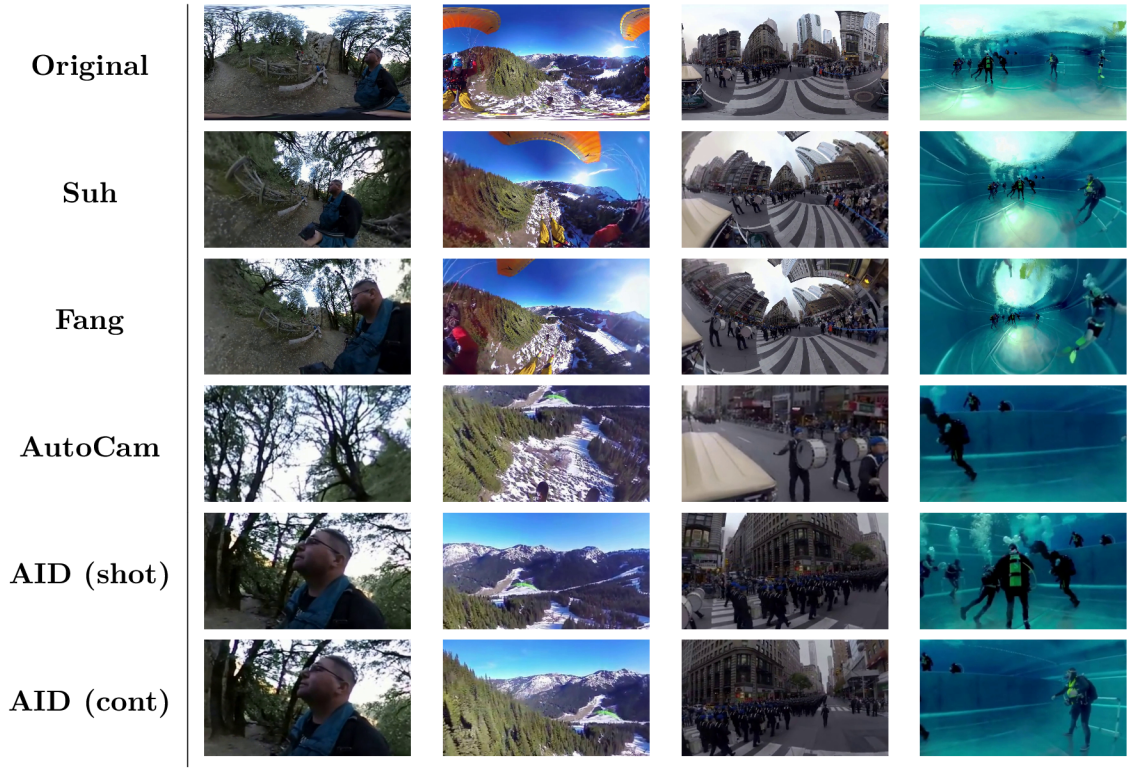
[30] Zhang, Z.; Xu, Y.; Yu, J.; et al.: *Saliency Detection in 360° Videos.* In *Proceedings of the European Conference on Computer Vision (ECCV).* ECCV. 2018. pp. 488–503.

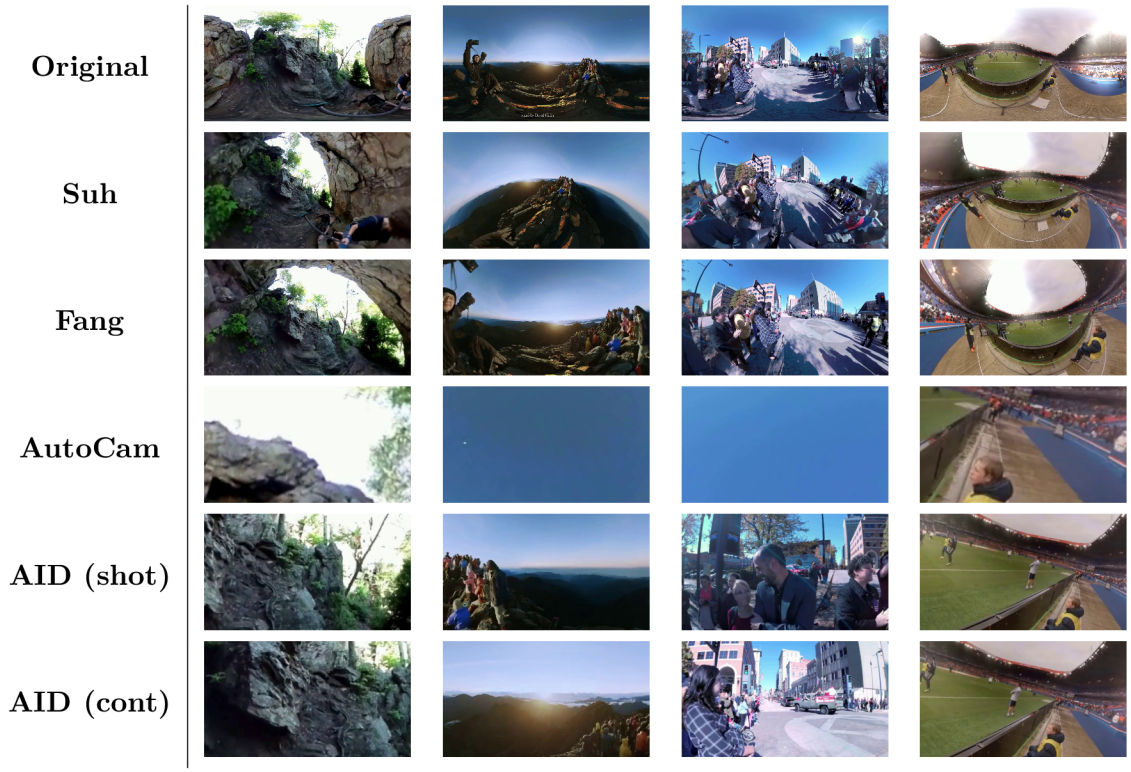
[31] Čadík, M.: *Perceptual Evaluation of Color-to-Grayscale Image Conversions.* Oxford, UK: Blackwell Publishing Ltd. 2008. ISSN 0167-7055. pp. 1745–1754.

Appendix A

Image results

This appendix includes several image results – frames from various original spherical videos and the corresponding output frames from tested methods in Chapter 5.




Original
Suh
Fang
AutoCam
AID (shot)
AID (cont)