Big Data Analysis for Media Production

By Josep Blat, Alun Evans, Hansung Kim, Evren Imre, Lukàš Polok, Viorela Ila, Nikos Nikolaidis, *Senior Member IEEE*, Pavel Zemčík, Anastasios Tefas, Pavel Smrž, Adrian Hilton, *Member IEEE*, and Ioannis Pitas, *Fellow IEEE*

ABSTRACT | A typical high-end film production generates several terabytes of data per day, either as footage from multiple cameras or as background information regarding the set (laser scans, spherical captures, etc). This paper presents solutions to improve the integration of the multiple data sources, and understand their quality and content, which are useful both to support creative decisions on-set (or near it) and enhance the postproduction process. The main cinema specific contributions, tested on a multisource production dataset made publicly available for research purposes, are the monitoring and quality assurance of multicamera set-ups, multisource registration and acceleration of 3-D reconstruction, anthropocentric visual analysis techniques for semantic content annotation, and integrated 2-D–3-D web visualization tools. We discuss as well improvements carried out in basic techniques for acceleration, clustering and visualization, which were necessary to deal with the very large multisource data, and can be applied to other big data problems in diverse application fields.

KEYWORDS | Anthropocentric semantic video analysis; big media data analysis and integration; graph processing acceleration; multimodal data processing; outdoor 3-D reconstruction; web 3-D visualization

Manuscript received August 14, 2015; revised September 23, 2015; accepted October 16, 2015. This work was supported by the European Commission, FP7 IMPART Project (Grant 316564).

J. Blat and **A. Evans** are with the Department of Information and Communication Technologies at Universitat Pompeu Fabra, 08018 Barcelona, Spain (e-mail: josep.blat@upf.edu; alunthomasevans@gmail.com).
H. Kim, **E. Imre**, and **A. Hilton** are with the Centre for Vision, Speech, and Signal Processing, University of Surrey, Surrey GU2 7XH, U.K. (e-mail: h.kim@surrey.ac.uk; h.imre@surrey.ac.uk; a.hilton@surrey.ac.uk).
L. Polok, **V. Ila**, **P. Zemčík**, and **P. Smrž** are with the Department of Computer Graphics and Multimedia, University of Brno, 61266 Brno, Czech Republic (e-mail: ipolok@fit.vutbr.cz; zemcik@fit.vutbr.cz; smrz@fit.vutbr.cz).
N. Nikolaidis, **A. Tefas**, and **I. Pitas** are with the Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece (e-mail: nikolaid@aiia.csd.auth.gr; tefas@aiia.csd.auth.gr; pitas@aiia.csd.auth.gr).

Digital Object Identifier: 10.1109/JPROC.2015.2496111

I. INTRODUCTION

The amount of data captured onset for film production is vastly increasing; currently, several terabytes are generated per day for a typical high-end film. Data come from a larger variety of capture devices, such as light detection and ranging (LIDAR) scanners, spherical cameras, still cameras, HD video cameras, 2.7 K/4 K cameras and RGBD cameras, as illustrated in Fig. 1. Other types of sensors might play a role as well. Generation and storage of digital data is considerably cheaper than in the (analog) recent past. These data need to be sorted, indexed and processed, currently requiring an immense amount of manual effort. In fact, the high volume of data generated during a shot prevents the immediate assessment of whether footage is fit for purpose. The current strategy (of reshooting in case of doubt) costs extra time and money on set, and leads to potentially redundant data, which also need to be processed. Even more data are generated during postproduction, as the raw input is usually processed multiple times, in order to obtain the output desired by the director. Extra, or poorly understood raw data, leads to higher postproduction costs and times.

This process, which requires a lot of manual input needs to be streamlined. By understanding better the data, it is possible to revert the trend of producing and storing even more data towards keeping only the suitable data instead, and to provide more intelligible content to the following stages of the digital cinema production chain.

This paper presents novel approaches for big media data analysis based on the integration of multiple big media data sources, which lead to solutions improving their management, and monitoring and understanding the quality of the data produced. The solutions support creative on-set or near-set decisions, and also facilitate and enhance postproduction, taking advantage as well of the semantic

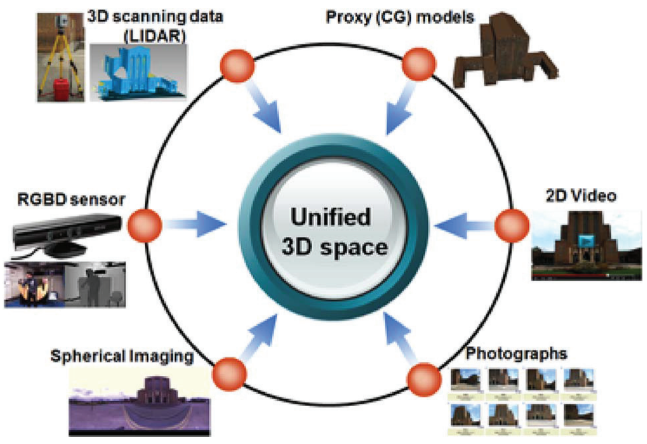


Fig. 1. ***Sources for multimodal data registration and visualisation.***

content analysis which are also presented. The structure and key contributions can be summarized as follows.

Section II starts by discussing the multiple data sources, which are typical in the production of data for high-end digital movies. It indicates the variety of devices used and data they generate and different types of representative environments (outdoor, indoor, single or multiple actors, etc.). A representative multisource IMPART dataset, approximately 10 TB in uncompressed format, has been generated to test the developed solutions with realistic and challenging material, and has been used in most of the experiments presented in this paper. It has been made public for the wider research community, together with a detailed documentation on the capture and some initial preprocessing, to facilitate its use.

Section III presents two key contributions with regards to quality assessment and registration:

1) Tools to monitor setups and to enhance quality assurance. The tools enable on-set detection of capture problems, namely: *poor coverage*, such as out-of-focus configuration, or insufficient detail provided; *calibration invalidation*, which is detected and the true camera parameters can be recovered on set or the calibration repaired in post production; and *synchronization loss* or *frame-drops*.
2) Multisource data (both 3-D, such as laser scans, and 2-D, from multiple and varied cameras) can be automatically registered into a common coordinate system (a "unified 3-D space," visually represented in Fig. 1). The current paradigm for efficient management of these data requires extensive manual input; our work introduces an automatic process to replace this manual effort, which in turn leads to a more efficient postproduction phase.

Section IV presents reformulations of some aspects of 3-D reconstruction from multiple sources introduced in Section III, and the underlying techniques are significantly optimized and accelerated, allowing for much faster processing to achieve near real-time. Complementary strategies of quality assessment and desktop visualizations of 3-D reconstructions are also presented. The reformulations can be used in other big data contexts, and include:

1) Sparse 3-D reconstruction from stills is performed through a novel fast bundle adjustment (BA) solver based on Block Matrices which features fast covariance recovery, where the BA task is formulated as a nonlinear maximum likelihood estimation on a graph of feature point observations by the respective cameras. Orders of magnitude efficiency gains are achieved.
2) Accelerated dense 3-D reconstruction from individual spherical stereo scans through reformulation of the image processing primitives in terms of partial functions called recursively for each pixel and a tiled cache as another partial function to save computation for local filters. Additionally, 3-D reconstruction from multiple spherical stereo pairs was accelerated by a novel alignment method, based on SLAM techniques.

While Sections III and IV deal with approaches related to 3-D data, Section V presents algorithms for high-level human-centered semantic metadata extraction and description through visual single- and multiview information analysis, to support fast big visual data ingestion, search and retrieval for postproduction, and archival. The presented methods are related to (human) activity-based temporal video segmentation and clustering, approximate methods for big media classification and a fast distributed clustering approach. Despite being evaluated on visual big data analysis tasks, such as facial images clustering or face recognition, most of the underlying techniques, such as distributed trimmed kernel K-means clustering or approximate methods for classification are designed so as to be fast enough to deal with large scale data, and can be very well applicable in other big data areas too.

Section VI presents interactive web visualization with integration of 2-D and 3-D sources and processed large data and metadata, for increased user driven quality assessment, creative on-set decisions and postproduction planning. Traditional big data visualization favors abstract representations, but practitioners of cinema and other fields rather prefer concrete ones (video and 3-D superimposed, in our case, for instance), while web-based visualization allows easy integration of modalities with advantage respect to desktop solutions. Improvements in progressive visualization of the very large (3-D) data are discussed as well.

Data sources and formats in Section III are largely media/cinema-specific, and we show that the applications presented in Sections IV–VI are useful for the cinema context. Additionally, the techniques in the latter sections required acceleration and improvements to deal with the challenges posed by cinema applications, and are both big data problems specific, and applicable in other areas. For instance, the PDE solution strategies, or the optimized graph-SLAM processes of Section IV are optimizations

Table 1 Examples of Typical Data Generated in Film Making

Data	Device	Format	Dimension	Volume	Used in
Principal camera	4K or HD Camcorrealder	DPX/RAW	2D+Time	Many Terabytes	All
Witness cameras	HD Camcorders	H.264/MP4	2D+Time	Many Terabytes	Animation
Motion capture	Xsens MOVEN2	Joint Angle	3D	Several Gigabytes	Animation/Rigging
Texture reference	DSLR camera	RAW/JPG	2D	Several Terabytes	Modelling/Texturing
Spherical HDR	Spheron	EXR	2D (Spherical)	Few Gigabytes	Lighting/Modelling
LIDAR Scans	Leica/FARO	Point cloud	3D	Few Terabytes	Modelling/FX

required by the very large size of data we deal with, which demand specific solutions to manage the memory and speed requirements of computers, and especially laptops which are the preferred on-set platform. The classification and clustering techniques of Section V can be applied to any type of big data where samples are represented in vectorial form. We discuss later in detail these and other examples of widely applicable improved big data techniques, especially in the final section, where we outline the actual validation of the solutions by the cinema industry.

II. BIG MEDIA DATA ACQUISITION

A. Multiple-Source On-Set Big Data Acquisition

In recent digital media production, a variety of 2-D and 3-D sensors capture a large number of assets and their geometrical properties (see Fig. 1). The amount of data generated on-set to support digital media production is increasing, as more devices become available and their data resolutions increase. In 2014, for example, the data for a film visual effects produced by Double Negative Visual Effects, one of the biggest European visual effects companies, consisted of several hundred terabytes, in different file formats from various devices as shown in Table 1.

Video cameras are the primary source of data in media production: as indicated in Table 1, their share in a typical production scenario far surpasses that of the auxiliary modalities. Multiple cameras have always been considered useful for editing. However, nowadays it is very common to see set-ups with a principal camera to shoot the main action, and a battery of static witness cameras to collect the necessary data for postproduction, in line with the current significance of "post" in determining the final success of a film. If not analog, a principal camera was conventionally a HD device, but 4 to 6 K resolutions are becoming increasingly common, and witness cameras usually HD.

Recently, low-cost RGBD cameras such as Kinect[1] and Xtion[2] simultaneously capturing color and depth information have become available to support motion analysis in the main action as well as to extract background geometry information. However, they still have limitations in large areas or outdoor scenarios, because of their limited depth range and interference between devices and materials.

In order to get more accurate static background scene information, various passive/active sensors are used. Digital stills are the most common source of texture information, due to their wide availability and ease of use.

High dynamic range omnidirectional spherical imaging is commonly used to get an aligned environment texture map or lighting source detection. An easy way to capture the full 3-D space in one shot is to use a catadioptric omnidirectional camera using a mirror combined with a CCD [1]. However, this is difficult to calibrate and has limited resolution. Point Grey developed an omnidirectional multicamera system, the Ladybug,[3] which consists of six XGA color CCDs to overcome the resolution problem. Spheron[4] developed a commercial line-scan camera with a fish-eye lens in order to capture the full static environment as a high resolution and high dynamic range spherical image.

Active depth sensors using ultrasonic, infrared or laser are sometimes used to reconstruct accurate geometry of objects or surfaces. LIDAR is one of the most popular active ranging techniques measuring the distance by the time delay between emission and reflection of a light pulse.

On-set dynamic scene acquisition takes place mainly outdoors, and differs from indoor studio shooting with controlled uniform lighting condition and background geometry (see [2] on the design of indoor studio capture systems). On-set capture for media production is more challenging, due to moving background, uncontrolled illumination and limited system support [3]. It requires aligned background scene information as well as dynamic actions in the main capture volume, on-set system monitoring and assessment tools against unsecured capture environments, and accurate composition of footage from various capture devices.

In order to support research into multimodal big data processing, we have released a big multimodal database acquired in various indoor and outdoor environments, available at http://cvssp.org/impart/. The dataset includes captured data typical of important issues facing movie production, with detailed information, and the corresponding 3-D reconstructions of static scenes and multiple synchronised video captures for dynamic actions, as illustrated in Fig. 2.

[1]Kinect: https://www.microsoft.com/en-us/kinectforwindows/develop/.
[2]Xtion: https://www.asus.com/us/Multimedia/Xtion PRO LIVE/.
[3]Pointgrey: http://www.ptgrey.com/.
[4]Spheron: http://spheron.com.

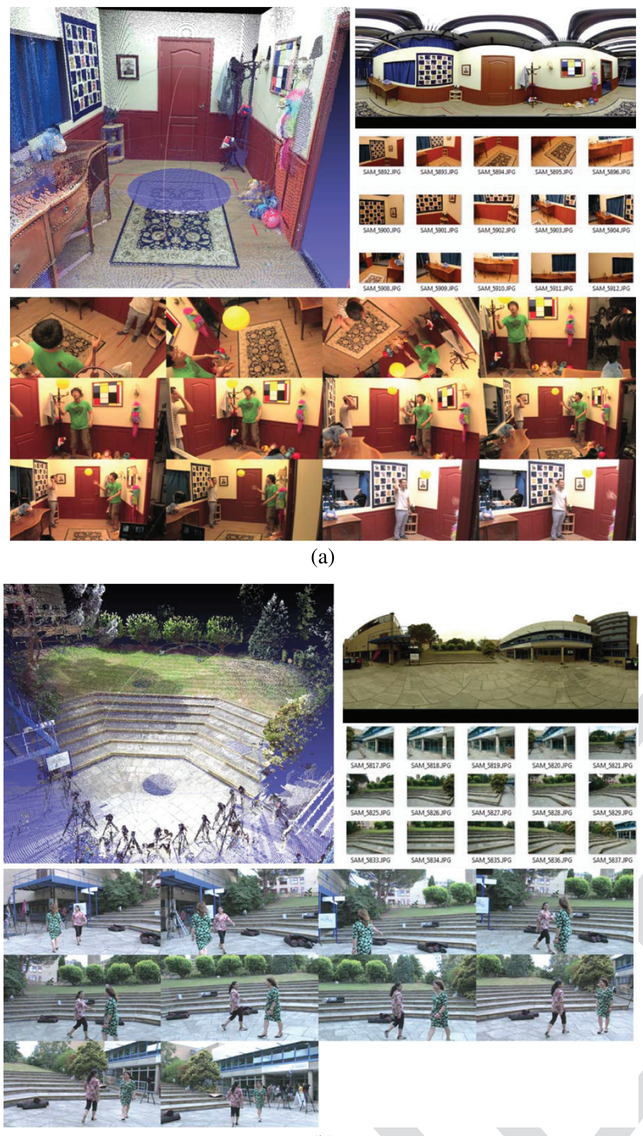


Fig. 2. ***Examples of the IMPART public multimodal dataset. (a) Indoor scene footage; (b) outdoor scene footage.***

B. Big Data Analysis Issues in Media Production

A key step in the analysis of the data has been the integration of the multiple, large, data sources, as it plays a significant role in understanding the data produced and its quality. The approach taken has been to adopt a "unified 3-D view" (see Fig. 1) for the sources, spherical images, stills, LIDAR scans, etc. It is a "3-D" approach, in the sense, for instance, that 3-D model/point-cloud is reconstructed from multiview images, and this allows to register better these images with respect to a ground-truth 3-D (in our case, assumed to be the LIDAR data); and the position of the sensors with respect to the reference system provided. Thus, multisource data are not just unified in a folder with place and time tags, but become integrated in the same space. On the other hand, this 3-D reconstruction of the background scenarios is in itself a very useful output for later postproduction effects.

This unified 3-D view supports solutions (toolsets) for monitoring the quality of the (multi)camera set-ups presented in Section III, in the different aspects mentioned above, such as coverage, synchronization, calibration, etc. The complex approaches to analyze these aspects with enough quality and flexibility to be used in the highly dynamic environment of on-set shooting are discussed in that section, as well as its evaluation with examples taken from the IMPART dataset.

A key requirement is that the solutions work in real-time or near real-time, so that fixing the issues, repairing the calibration, or setting up the cameras to improve the coverage of the scenario, etc., can take place on-set, where shooting takes place. Currently, the issues described above are detected during revision of dailies, or even during postproduction, away from the set; in both cases, the costs of shooting again the following day, or fixing during postproduction, are very high. To achieve near real-time solutions of the very high quality expected in film blockbusters shown on cinema screens represents an important challenge.

On one hand, the multiple large data sources pose very strong memory and processing demands that have to be dealt with; and preferably, not with supercomputers but with the more appropriate laptops for the on-set environment. Section IV discusses mostly the acceleration and optimization of the techniques to make the approaches presented in Section III for reconstruction and quality monitoring real- or near real-time. This is achieved by reformulation of the algorithms, with suitable use of CPU and GPU and distribution; resulting in orders of magnitude speed improvements in some basic processes used in the algorithms—largely applicable to other areas.

An integrated visualization is both a useful outcome and a tool for users to assess quality further. Section IV discusses laptop-oriented streaming techniques for the very large 3-D outcomes. Section VI is web oriented, showing its advantage to integrate a new type of semantic—2-D–3-D interactive visualizations, which moreover could be easily shared and annotated in the on-set environment. The additional challenges posed by bandwidth constraints, and specific 3-D graphics issues are addressed as well.

Section V presents action analysis from an anthropocentric perspective, generating metadata, which is useful to search and retrieve in a more intelligent way the content produced and for integrating semantics in visualizations as indicated above. The data to be dealt with is much larger: instead of being the result of a (part of a) day's shooting, the solutions should handle the data produced and postproduced in a whole production, or several of them. The improved optimization and acceleration techniques presented with respect to clustering, and using approximate methods, were needed to face this challenge providing a practically applicable solution.

III. DATA QUALITY AND REGISTRATION

First, we discuss a toolset for set-ups, and then, for registration.

A. Set-Up Monitoring and Quality Assurance

In media production, the conventional quality assurance mechanism is the review of the "dailies" (the material shot and produced during the day) at the end of each day's session. However, this process can easily be overwhelmed by the huge volumes of data that the current media production practices generate, with their use of multiple cameras and an assortment of auxiliary sensors (see Table 1), and the current solution, "if in doubt, reshoot," offers robustness through redundancy, but, as discussed, this leads to additional associated costs in terms of time and money.

The IMPART toolset offers on-set decision support and quality assurance capabilities for multicamera set-ups. The specific problems the toolset addresses include the assessment of the coverage of the capture volume, validation of the camera calibration parameters, and through-the-lens synchronisation of the set-up. The individual tools are discussed in the following sections.

1) Coverage Evaluation: A camera is said to successfully cover a volume element in the scene if it meets some application-specific criteria, e.g., the size and the location of this volume projection on the image plane. The coverage evaluation tool characterizes the scene coverage offered by a camera configuration with known pose and intrinsic parameters. This is a variant of the sensor placement problem, with its roots in the "Art Gallery Problem" [4], and its applications in surveillance [5] and industrial machine vision [6]. However, the optimal solutions provided by the sensor placement literature imply rigid preplanning and precise camera placement. In media production, on-the-spot decisions and flexibility are important, which makes feedback on an existing configuration more valuable than setting up an optimal, but inflexible alternative [7].

In order to characterise the coverage, a range of 2-D and 3-D world models with different geometric primitives is discussed in [6]. In [7], we consider a 3-D cloud of transparent spheres covering the scene, viewed by pinhole cameras. This permits more realistic camera models [8], and is sufficient to characterise how a configuration covers a specified capture volume (the space where the action takes place). The sphere cloud is then projected through the cameras, to a set of ellipses. A volume element is covered if it satisfies an application-dependent subset of unary and binary criteria listed below.

Unary criteria (involving a single camera):

- field-of-view: the sphere lies within the viewing frustum of the camera, and its projection lies within the image frame;
- framing. concerns the positioning of the subject matter in the image frame. It is satisfied if the projection lies within a specified region of the image plane;
- resolution: this constraint ensures that the volume is imaged at sufficient detail, which can be judged from the area of the ellipse associated with the volume element;
- depth-of-field. : a scene point is in focus if the diameter of the blur circle on the image plane is less than the pixel size. This defines a subvolume in the viewing frustum, where any scene points within satisfies this constraint [9].

Binary Criteria (involving camera pairs):

- viewpoint difference: defined as the maximum difference between the viewing angles of the cameras observing the scene point;
- relative resolution: concerns the maximum scale difference between the two projections of the scene point, measured as the ratio of the areas of the projection ellipses;
- joint coverage: a volume element is jointly covered if it satisfies the unary coverage criteria for each of the cameras separately.

The usefulness of the tool is demonstrated in a camera placement scenario in a set whose 3-D model is depicted in Fig. 3 [7]. Fig. 4 illustrates the layout of the cameras, placed **AQ1** on a 180-degree arc surrounding the capture volume, in addition to a top-view of a sparse 3-D model built from the images acquired by the cameras [10], and the capture volume, represented with a synthetic lattice embedded into the scene. It also presents the coverage evaluated for each scene point. Since the cameras are trained on the capture volume, the coverage reaches to 15 cameras within it, but drops off to 2 cameras for the background points. Fig. 5 indicates the change of the coverage during a dolly shot with a principal camera. As expected, the coverage is proportional to the overlap between the capture volume

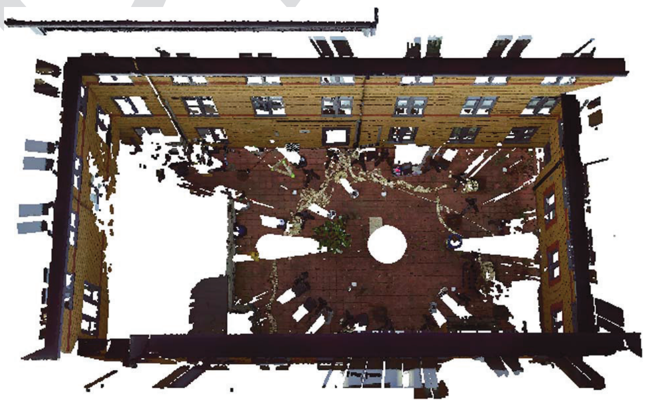


Fig. 3. ***3-D model of the capture environment, obtained via a color LIDAR.***

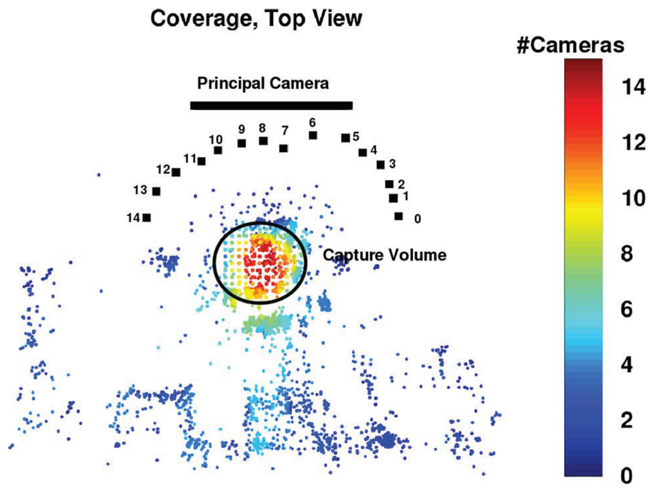


Fig. 4. *Coverage map for the witness cameras. Hotter colors indicate better coverage. The witness cameras are marked by a numbered square. The principal camera moves to right, on a path denoted by the horizontal line.*

and the field-of-view. Finally, Fig. 6 is a vision graph, which, unlike the first two examples, makes use of the binary constraints, and helps to identify any redundant cameras, or any segments, which may benefit from an additional camera.

2) Calibration Validation: For a multicamera set-up deployed for postproduction purposes, a very accurate calibration is essential. Such a calibration can be obtained via a dedicated, preliminary shot involving a known object [11]. However, maintaining calibration throughout multiple shots poses a real challenge, considering how busy a production set is and the cost of any delay. The calibration validation tool identifies any cameras which are unintentionally perturbed, invalidating their calibration. The correct calibration parameters can be recovered by subsequently registering them to a 3-D model of the scene estimated from the rest of the set-up [12]. This strategy is preferable to recalibration via, for example, VisualSfM [13], [14]: since the perturbations are often minor and limited to a small number of cameras, and the computational expense of a full structure-from-motion procedure is not justified.

Pairwise calibration validation: The calibration parameters for a camera pair imply a geometric relationship between the corresponding image features, which is encapsulated by a 3 × 3 matrix (i.e., a fundamental or an essential matrix). This relationship can also be estimated directly from the image correspondences. The calibration validation tool leverages on the observation that, the estimated and the implied relationships will be consistent only if the calibration parameters are correct. Otherwise, at least one of the cameras has invalid calibration parameters. However, it is not possible to identify which one, or whether both are perturbed at the pairwise level.

Global calibration validation: The pairwise validation block issues a verdict of *intact* or *perturbed* for each camera pair. This leads to a graph, where each node is a camera and each edge corresponds to the verdict for the associated camera pair. The number of inliers (successfully explained image correspondences) are assigned as edge weights. Any vertices for which there are too few observations (e.g., less than two edges) are labeled as *undetermined*. For the remaining cameras, the algorithm seeks the best combination of vertex labels, by exhaustively instantiating all possible combinations of the binary tags *valid* and *invalid*. An *intact* edge is consistent with a labelling, if the associated vertices are *valid*. A *perturbed* edge requires at least one *invalid* vertex. A labeling hypothesis is scored by summing up the weights of the consistent edges.

Fig. 7 depicts the calibration validation pipeline. As an example, the calibration parameters obtained via [11] for the set-up in Fig. 4, when employed in a multiview triangulation task [10], yield 4711 3-D points. The calibration verification tool recommends the correction of the cameras 4, 8, and 14. With the updated parameters, the triangulation algorithm returns a 3-D structure with 5512 points.

3) Multicamera Synchronization: Any postproduction task that makes use of a multicamera set-up invariably requires synchronisation. This is typically achieved by a hardware signal. A through-the-lens synchronisation and frame drop detection capability is of value in the event that:

- the hardware signal is interrupted, for example, due to a broken or disconnected cable;
- a device does not have a hardware synchronisation facility (e.g., a Kinect), or cabling is impractical (e.g., a principal camera on a moving platform);
- the recording media drops frames due to, for example, a buffer overload.

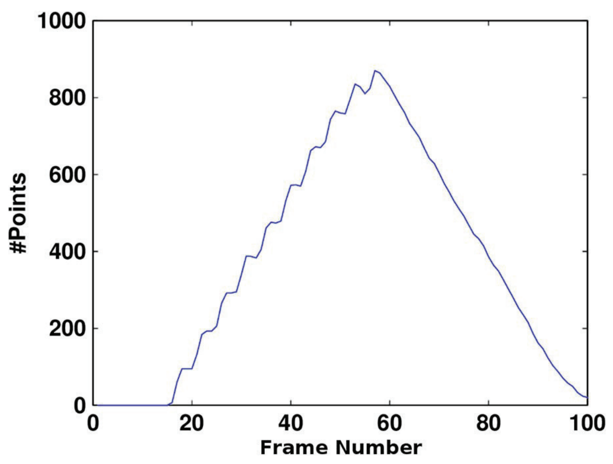


Fig. 5. *Number of capture volume points covered by the principal camera, as it moves to right.*

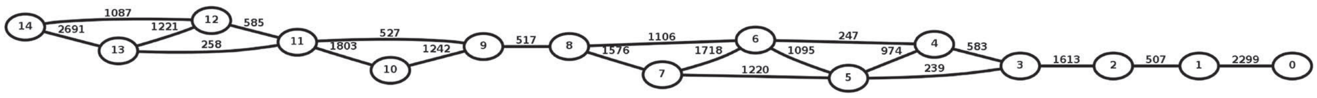


Fig. 6. ***Vision graph for the witness cameras. See Fig. 4 for the camera numbers.***

The alternative, audio synchronization is sensitive to the background noise, and limited to frame-level synchronisation, due to the slow sound propagation speed in the air [15].

The multicamera synchronization tool estimates a frame rate and an offset for each camera in the set-up. It also detects the frame drop events, reporting a time window and the number of dropped frames for each event [16]. The tool makes use of the observation that, if two frames are acquired at the same time instant, the corresponding image points on the dynamic scene elements (e.g., actors) satisfy a certain geometric constraint (epipolar constraint) [17]. This requires a calibrated camera set-up, but enables the use of point features, sidestepping a major challenge: feature trajectories, as used in [18] and [19], are difficult to establish and maintain on deformable objects such as actors. As alternatives, spatio-temporal features [20] and global image similarity metrics [21] do not have this issue, but they are very sensitive to viewpoint and appearance changes [22].

The synchronization pipeline, illustrated in Fig. 8, has two distinct stages: relative and absolute synchronization.

Relative camera synchronization: In the presence of frame drops, the indices of the corresponding frames for a pair of image sequences lie on a broken line with a fixed slope. The slope corresponds to the relative frame rate, whereas the offset of the first segment is the relative temporal offset. Any frame drop events are manifested as the break points, where the number of lost frames can be estimated from the shift. The relative synchronisation module establishes the index correspondences via the Viterbi algorithm [23], where the similarity of a frame pair is measured by the conformance of the image feature pairs to the epipolar constraint. When there are gaps in the index correspondences, the time of the frame drop events can be reported as windows.

Absolute synchronisation: The pair-wise synchronisation measurements can be represented on a graph, where each vertex is a camera, and each edge corresponds to a relative synchronisation measurement, weighted by the number of supporting frame indices. In this graph, each minimum spanning tree offers an absolute synchronisation hypothesis, namely a frame rate and a temporal offset for each camera. The absolute synchronisation procedure samples the solution space by randomly generating minimum spanning trees and returns the best hypothesis. A hypothesis is scored by summing the weights of the edges consistent with it.

Frame drop fusion: Each pair-wise measurement effectively proposes a temporal offset for each frame in the associated camera pair. The fusion algorithm scores these proposals with the edge weights in the absolute synchronisation graph. A frame drop is reported for the segments where the strongest offset hypothesis is different from the absolute offset estimated for that camera.

As an example, the tool is tested on a 30-s action sequence acquired by the set-up in Fig. 4. The tool successfully identifies the −1, 20, and 1-frame offsets in the cameras 8, 11, and 15, respectively, along with the correct frame rate. A more extensive evaluation in [16] reports that the frame drop events can be localised down to a 1–3 s temporal window, and the number of lost frames is correctly identified.

B. Multimodal Data Registration

The multimodal data capture process ends up with huge amount of unstructured footage which is hard to efficiently search, arrange and manage. Datasets acquired

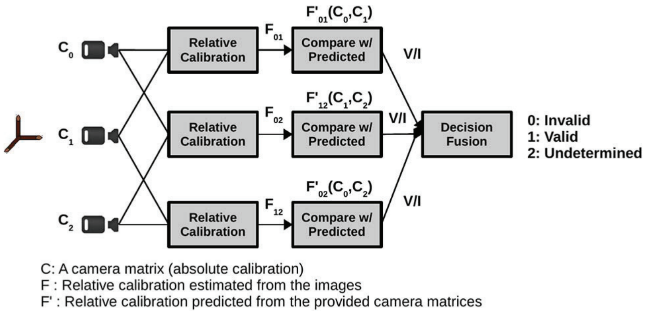


Fig. 7. ***Calibration validation pipeline.***

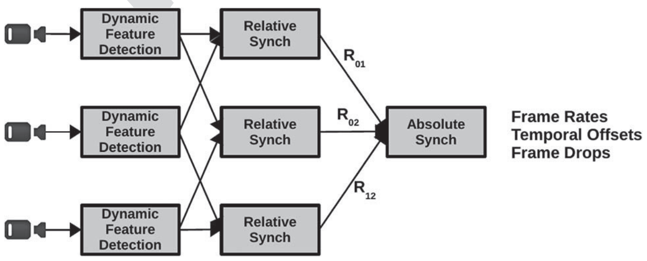


Fig. 8. ***Synchronization pipeline.***

by various devices introduced in Section II-A exist in different coordinate systems with different dimensions, formats, densities, characteristics and noise as shown in Table 1. The processing and management of this amount of heterogeneous data consumes considerable resources, most of them intensive manual labor. A key issue to allow efficient data management and visualization is automatic registration of the multisource data into a common coordinate system.

There have been a few researches for 2-D–3-D data matching and registration [24]–[26], but they have focused only on registration for a single data modality. 2-D–3-D registration between pairs of modalities such as photos to LIDAR [27], [28], spherical images to LIDAR [29], and images to range sensor [30], [31] have also been investigated. In our preliminary researches, 3-D feature descriptors were tested and their performance on multimodal registration in various domains was analyzed [32], [33].

Here, we introduce a complete pipeline for multimodal 2-D–3-D data registration for media production, based on a unified 3-D space where 2-D and 3-D data are registered as shown in Fig. 1. The LIDAR point cloud and its coordinate system are defined as the reference (target) model space for registration because they provides an accurate 3-D geometry of the scene at real-world scale. If LIDAR scans are not available, any data modality can be used as a target reference.

1) 2-D Data Registration via 3-D Reconstruction: 2-D data are registered to the target coordinate via 3-D reconstruction because direct registration of 2-D images to 3-D structure is difficult. We assume that multiple 2-D data are available for the same scene so that 3-D geometric information can be extracted. Relative camera pose information to the reconstructed model is computed during the 3-D reconstruction process. As a result, all original camera poses are automatically registered to the target reference coordinate if the reconstructed model is successfully registered to the target reference model.

RGBD cameras provide appearance (color) and respective depth information. 3-D scene geometry and camera poses can be estimated from consecutive RGBD frames using the KinectFusion algorithm [34]. Registration of normal photographs without depth information is more challenging.

Shape-from-Motion algorithms such as Bundler [35] followed by PMVS [36] provide dense 3-D scene reconstruction with camera poses from photos. Autodesk also provides an on-line image-based 3-D reconstruction tool with camera pose estimation, RECAP360.[5] However, these Shape-from-Motion approaches for digital stills may not be appropriate for on-set media production due to their processing speed and being proprietary. A fast reconstruction algorithm from digital stills is proposed later in Section IV-A. A spherical image is represented on the longitude-latitude coordinate instead of the common $x-y$ system. Spherical images are captured as vertical stereo pairs to allow dense reconstruction of the surrounding scene using stereo matching [37]. This stereo matching algorithm is further accelerated in Section IV-C.

In case of multiple wide-baseline witness video cameras, the sparse reconstruction from Section III-A can be used for registration. However, it is sometimes difficult to extract good geometry of the static background if cameras are too sparsely placed (i.e., they have little overlap) to find corresponding points between viewpoints. In such a case, camera calibration information estimated by wand-based extrinsic camera calibration [11] is directly registered by aligning the calibration coordinates to the origin of the LIDAR sensor.

2) 3-D Feature Detection: 3-D feature detection identifies locations of distinct points in terms of shape or appearance in an input 3-D structure. Feature detection is an important step because its distinctiveness and repeatability across models directly influences the performance of matching and registration. Many feature detection methods for 3-D point clouds have been investigated. Dutagaci *et al.* [38] and Tombari *et al.* [39] provide benchmark evaluations of existing 3-D feature detectors. However, all detectors which were highly ranked in these evaluations do not guarantee such high performance for multimodal data, due to their noise, geometric errors and distortions in 3-D reconstruction.

We evaluated various Various 3-D feature detectors, resulting in our choice to use the 3-D extension of the classic Kanade-Tomasi detector [40], which uses the ratios of eigenvalues of surface normal vectors for 3-D edge and corner detection. This detector is not too selective but still produces a relatively high number of repeatable and distinctive 3-D features between cross-modalities in spite of geometrical errors induced from incomplete 3-D reconstruction.

Fig. 9 shows feature points detected for the Cathedral scene available in the IMPART public multimodal database. The Kanade–Tomasi detector and the 3-D SIFT detector [41], one of the most popular feature detectors using difference-of-Gaussian filter and Hessian eigenvalue test, were compared. The SIFT and Kanade–Tomasi detect a similar number of feature points but the results of Kanade–Tomasi are more distinctive in representing clean 3-D edges and corners.

3) 3-D Feature Description and Registration: A feature descriptor is a vector representing different distinctive characteristics of a specific point in the scene. Recently, Guo *et al.* [42] presented a survey and evaluation of local 3-D feature descriptors but the test was carried out for a single modality. Most 3-D feature descriptors use local

[5]RECAP360: http://recap360.autodesk.com.

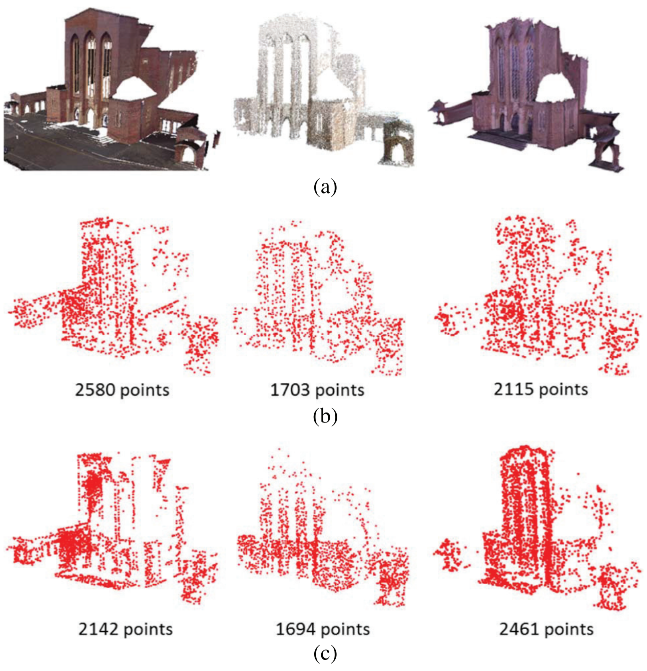


Fig. 9. ***3-D Feature detection results (Left: LIDAR; Middle: reconstruction from Pphotos; Right: reconstruction from spherical imaging). (a) Point cloud; (b) 3-D SIFT detector; (c) 3-D Kanade–Tomasi detector.***

geometric information. However, these descriptors are suitable only for models with low geometric errors. In our preliminary research for multimodal data registration [33], we verified that a combination of descriptors applied on different domains such as colour and local/global geometry can improve the point matching and registration performance.

Among various descriptors introduced in [33] and [42], we use fast point feature histograms (FPFH) [43] in our pipeline because it is fast and shows stable performance with a short description. FPFH is computed by a weighted sum of neighboring point feature histogram values which are calculated by three angles between neighboring points. One FPFH descriptor is represented as a vector with 33 bins (11 bins for each angle).

The FPFH descriptor is extended to multiple domains to use geometry and color information together. FPFH descriptors in three different domains (local, semiglobal, and color) are calculated for the same input point cloud. The Local FPFH is calculated with neighboring points in the same way as in the original FPFH descriptor. The semiglobal FPFH is calculated only with detected feature points in the larger volume radius, which represents the distribution of feature points. The color FPFH is calculated with the same neighbours of the Local FPFH, but it uses CIE color components instead of surface normal components. The result is represented as a 2-D vector with 33 $\times$ 3 bins.

Once the 3-D descriptor sets are computed, all datasets are registered to the target (LIDAR) point cloud by descriptor matching. There may be many outliers in descriptor matching because of low distinctiveness and repeatability of detected features. RANSAC is a common method to find an optimal solution when unknown outliers exist. SAC-IA [43] is a RANSAC-based initial alignment algorithm which eliminates outliers and estimate a 3-D rigid transform matrix between source and target models. We modify this SAC-IA algorithm to adaptively adjust the contribution of description domains in matching according to the distinctiveness of the descriptor. The matching cost between two points $\mathbf{p}$ and $\mathbf{q}$ in the RANSAC process is defined as a weighted sum of individual domain descriptor matchings of the form

$$D(p,q) = \lambda_L D_L(p,q) + \lambda_G D_G(p,q) + \lambda_C D_C(p,q) \qquad (1)$$

where $D(\cdot)$ denotes the distance between two descriptors, and subscripts L, G, and C represent local, semiglobal, and color domains, respectively. The weighting factor λ is computed by the ratio of the second to the first nearest neighbor distances. The initial alignment resulted from the modified SAC-IA is refined over the whole point cloud using the iterative closest point (ICP) algorithm [44].

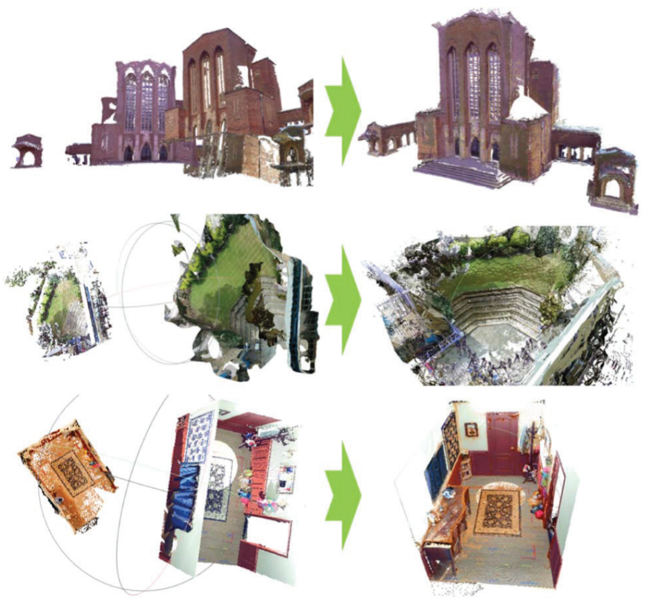
Fig. 10 illustrates the original datasets and registration results. In Fig. 10(a) the original 3-D point clouds generated from different sources exist in different coordinates which are automatically registered into a single unified coordinate system through the proposed registration pipeline. The original 2-D footage is visualized on the target LIDAR reference in Fig. 10(b). We can observe that all 2-D footage including photographs and HD videos are registered to the target coordinate with correct location and orientation.

This enables the automated registration of multimodal data sources allows web-based visual inspection for completeness and supporting creative decisions in production, as discussed in Section VI.

IV. ACCELERATION AND QUALITY ASSESSMENT OF BIG 3-D RECONSTRUCTION

To support various special effects, principal camera match-moving, and rendering animated characters, among others, 3-D reconstruction is often employed in digital cinema production. In Section II, we described a wide variety of available input sensors for 3-D reconstruction is available.

Two basic modalities of the 3-D reconstruction are commonly used, static and dynamic. The static one is typically used for the reconstruction of the movie set or for the natural environments to complement digital matte paintings. The variety of input sensors can be used, and a plethora of algorithms exist for this purpose. On the other hand, dynamic 3-D reconstruction, on the other hand, is


(a)

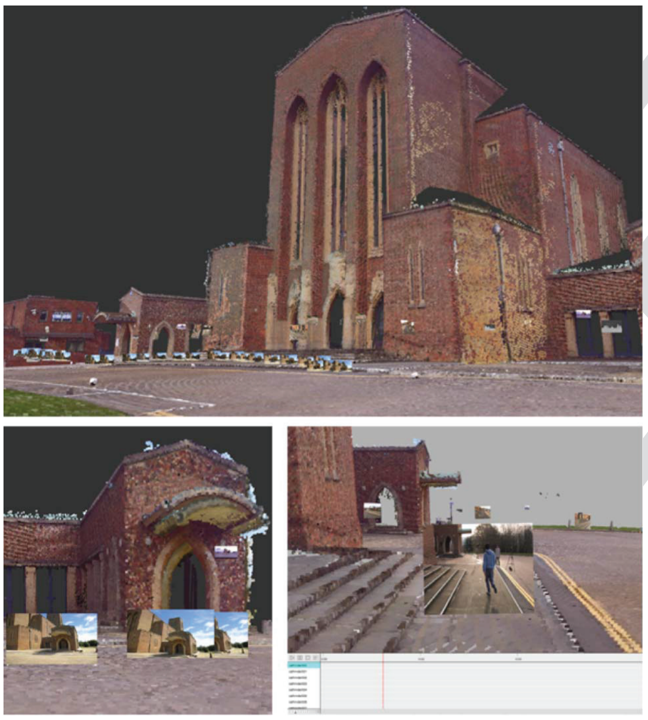

(b)

Fig. 10. ***Multimodal data registration results. (a) Point cloud registration [Top: cathedral set in Fig. 9(a); Middle: indoor set in Fig. 2(a); Bottom: outdoor set in Fig. 2(b)]. (b) Visualization of registered 2-D footage in the unified 3-D space.***

typically being used for the actors and it can be obtained by a multiview camera setup and the visual hull algorithm [45] or one of its variants [46]. In this section, we focus mainly on the static 3-D reconstruction.

Sparse 3-D reconstruction can be obtained from still images using the bundle adjustment algorithm [47]. In this context, the term sparse refers to the relative density of the 3-D point cloud obtained. A dense reconstruction can be obtained by an additional postprocessing pass, e.g., by the PMVS [36] or CMVS [48] algorithms. Unlike the sparse reconstruction, the dense reconstruction is not particularly interesting from the acceleration and quality assessment points of view, as it seldom fails and it usually gives satisfactory results if it is given a successful sparse reconstruction to start with.

A. 3-D Reconstruction From Stills

Several open-source and commercial software packages are available for 3-D scene reconstruction from unstructured set of photographs of the scene: Bundler [35], VisualSFM [13], PhotoSynth,[6] PhotoScan,[7] to name just a few. Most of the existing software packages are based on bundle adjustment (BA) [35] to obtain a refined structure of the environment from captured images.

However, usage of such algorithms in digital cinema production suffers from several drawbacks. A main one is their execution time; the reconstruction using the available software takes far too long to be performed on-set; therefore, it is typically performed later, on a render farm. This can lead to problems if it turns out that the capture was insufficient, requiring the repetition of the scene capture. Another problem is the use of cloud computing by several of the available solutions (Microsoft's Photosynth, Autodesk 123-D,[8] and RECAP360[9]). When processed in a Cloud, the original data are transferred outside the VFX facility or even outside of the respective country, which is always an issue with copyright-protected or otherwise sensitive content.

In this work, we propose a novel fast BA solver based on Block Matrices [49] which features fast covariance recovery [50] and hence enables online error visualization and correction. We formulate the Bundle Adjustment task as a nonlinear maximum likelihood estimation on a graph of feature point observations by the respective cameras, similar to [47].

As discussed in the previous section, the 3-D reconstruction starts by calculating the initial poses of the cameras and the 3-D points in the environment. They are obtained by considering pair-wise image matching. In general, every 3-D point is visible in more than two images and the contributions of all the measurements need to be considered for a better estimation of the 3-D structure. Bundle adjustment starts from the initial estimates of the camera and 3-D point poses and iteratively refines the solution. Conceptually, this is done by minimizing the reprojection errors. In our work we formulate the BA as a nonlinear optimization on graphs, where the vertices are the variables to be estimated, namely camera poses and 3-D points in the environment, and the edges are the

[6]https://photosynth.net/
[7]http://www.agisoft.com/
[8]http://www.123dapp.com/catch
[9]https://recap.autodesk.com/

measurements. In order to obtain the optimal configuration of the graph, we perform a maximum likelihood estimation (MLE) of the set of variables $\boldsymbol{\theta} = [\theta_1 \dots \theta_n]$, usually containing the 3-D points in the environment $\mathbf{p} = [p_1 \dots p_{np}]$ and camera parameters $\mathbf{c} = [c_1 \dots c_{nc}]$ (position, and in the case of uncalibrated cameras, the intrinsic parameters), given the set of observations $\mathbf{z} = [z_1 \dots z_m]$

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \; P(\boldsymbol{\theta} \mid \mathbf{z}) \\ &= \arg\max_{\boldsymbol{\theta}} \; \{-\log(P(\boldsymbol{\theta} \mid \mathbf{z})\}.\end{aligned} \tag{2}$$

Each observation $\mathbf{z_k}$ is the 3-D point projection onto the image plane, $\hat{z}_k = \Pr_k(c_i, p_j)$, where $\Pr(\cdot)$ is the projection function of a 3-D point, p_j, onto the camera, c_i. Each observation is assumed to have zero-mean Gaussian noise with the covariance Σ_k and we measure the reprojection error

$$P(z_k \mid c_i, p_j) \propto \exp\left(-\frac{1}{2}\left\|z_k - \Pr_k(c_i, p_j)\right\|^2_{\Sigma_k}\right) \tag{3}$$

where z_k is the actual value in pixels of the projected 3-D point. Finding the MLE from (2) is done by solving the following nonlinear least squares problem:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \left\{\frac{1}{2}\sum_{k=1}^{m}\left\|z_k - \Pr_k(c_i, p_j)\right\|^2_{\Sigma_k}\right\}. \tag{4}$$

Iterative methods such as Gauss–Newton (GN) or Levenberg–Marquard (LM) are used to find the solution of the NLS in (4). An iterative solver starts with an initial point $\boldsymbol{\theta}^0$ and, at each step, computes a correction $\boldsymbol{\delta}$ towards the solution. For small $\|\boldsymbol{\delta}\|$, Taylor series expansion leads to linear approximation in the neighborhood of $\boldsymbol{\theta}^0$

$$\tilde{e}(\boldsymbol{\theta}^0 + \boldsymbol{\delta}) \approx \mathbf{e}(\boldsymbol{\theta}^0) + J\boldsymbol{\delta}, \tag{5}$$

where $\mathbf{e} = [e_1, \dots, e_m]^\top$ is the set of all nonlinear reprojection errors between the observed and reprojected 3-D points, $e_k(c_i, p_j, z_k) = z_k - \Pr_k(c_i, p_j)$, with $[c_i, p_j] \subseteq \boldsymbol{\theta}$ and J is the Jacobian matrix containing the derivative of the components of $\mathbf{e}$.

Thus, at each ith iteration, a linear least squares problem needs to be solved

$$\boldsymbol{\delta}^* = \arg\max_{\boldsymbol{\delta}} \frac{1}{2}\|A\;\boldsymbol{\delta} - \mathbf{b}\|^2 \tag{6}$$

where the $A = \boldsymbol{\Sigma}^{-\top\backslash 2}J(\boldsymbol{\theta}^i)$ is the system matrix, $\mathbf{b} = \boldsymbol{\Sigma}^{-\top\backslash 2}\mathbf{e}(\boldsymbol{\theta}^i)$ the right hand side (r.h.s.) vector and $\boldsymbol{\delta} = (\boldsymbol{\theta} - \boldsymbol{\theta}^i)$ the correction to be calculated [51]. The minimum is attained where the first derivative vanishes

$$A^\top A\;\boldsymbol{\delta} = A^\top \mathbf{b} \quad \text{or} \quad \Lambda\boldsymbol{\delta} = \boldsymbol{\eta} \tag{7}$$

with $\Lambda = A^\top A$, the square symmetric system matrix and $\boldsymbol{\eta} = A^\top \mathbf{b}$, the right hand side vector.

The solution to the linear system can be obtained either by sparse matrix factorization followed by backsubstitution or by linear iterative methods. After computing $\boldsymbol{\delta}$, the new linearization point becomes $\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^i \oplus \boldsymbol{\delta}$.

In BA applications, the initial solution θ^0 can be relatively far from the optimal one; therefore, LM is preferred over the GN methods. LM is based on efficient damping strategies which allow convergence even from poor initial solutions. For this reason, LM solves a slightly modified variant of (7), which involves a damping factor λ

$$(\Lambda + \lambda\bar{D})\boldsymbol{\delta} = \boldsymbol{\eta} \quad \text{or} \quad H\boldsymbol{\delta} = \boldsymbol{\eta} \tag{8}$$

where $\bar{D}$ can be either the identity matrix, $\bar{D} = I$, or the diagonal of the matrix Λ, $\bar{D} = \mathrm{diag}(\Lambda)$.

The remaining part of the section discusses important aspects that must be understood in order to compute efficiently solutions of the BA problem from linear system properties to nonlinear solvers.

Indeed, to solve the linear system in (8) efficiently, some particulars of the BA problem can be considered. For example, by grouping the elements of the system matrix corresponding to the camera poses and the 3-D points separately, one can solve for camera poses first and refine the 3-D points in a second step. This is a common practice in solving 3-D reconstruction problems, where the camera poses are linked only through the points and the algebraic decomposition of the system matrix is called Schur Complement. For this purpose, the system matrix is split in four blocks

$$\begin{bmatrix} C & U \\ U^\top & P \end{bmatrix} \cdot \begin{bmatrix} \mathbf{c} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\eta}_c \\ \boldsymbol{\eta}_p \end{bmatrix}. \tag{9}$$

Usually, the number of cameras is much lower than that of observed points and P occupies a relatively large portion of the matrix.

It results in diagonal C and P matrices, that can be easily inverted. Following that P is invertible, the Schur complement of the block P is $C - UP^{-1}U^\top$, and it is used to solve for the camera poses first (this is sometimes referred to as the reduced camera system). Points are then obtained by solving the remaining system.

Table 2 Bundle Adjustment Solving Time Breakdown; Intel Core i5 is Mid-Range Quad Core CPU, NVIDIA GTX 680 is Low-End Consumer GPU and NVIDIA K40 is High-End Scientific GPU

Operation	Implementation	Device	Time [sec]
Cholesky factorization	Sparse (CSparse)	Core i5 CPU	215.83
	Blockwise (SLAM ++)	Core i5 CPU	32.75
	Dense (Eigen)	Core i5 CPU	0.91
	Dense (CULA)	GTX 680 GPU	0.72
	Dense (CULA)	K40 GPU	**0.22**
GEMM2	Sparse (CSparse)	Core i5 CPU	28.72
	Blockwise (SLAM ++)	Core i5 CPU	13.25
	Sparse (cuSPARSE)	GTX 680 GPU	16.84
	Sparse (cuSPARSE)	K40 GPU	13.05
	Blockwise (proposed)	K40 GPU	**6.05**

To obtain fast solutions of the BA problem, sparse block matrices are employed. The derivatives in Λ are grouped into blocks, whose size corresponds to the number of degrees of freedom (DOF) of the corresponding variables: three DOF for the 3-D points and 12 DOF for the cameras (6 for description of the pose and a further 6 for the intrinsic camera parameters). We can say that the problem has a natural sparse block structure; therefore, for the representation and solving of (8) or (9), it is preferable to use sparse block matrix representation and linear algebra packages rather than elementwise sparse representation (such as, e.g., compressed sparse row (CSR) or compressepoint cloud sparse column (CSC) [52]). A detailed description of the block matrix data structure for fast nonlinear solving can be found in [53], while [54] shows how this data structure highly benefits real-time solving.

Note that most of the existing packages do not fully take advantage of this block structure, and despite some of them use an intermediate sparse block matrix formats, they all convert to element-wise sparse matrix before solving the associated linear system. Using sparse block matrices is the main novelty of our approach.

To obtain even higher performance, it is possible to take advantage of parallel architectures, such as GPUs. The most time consuming operations in the solving process (9) on a CPU are 1) matrix multiplication in UP^{-1}, further referred to as general matrix multiplication GEMM1, 2) the second matrix multiplication in $UP^{-1}U^{\top}$, further referred to as GEMM2, and 3) decomposing the system in $C - UP^{-1}U^{\top}$, further referred to as reduced camera system solve (RCS).

As already mentioned above, the size of the RCS is much smaller than the original system since the number of cameras is much smaller than the number of observed 3-D points. On the other hand, the corresponding matrix is also less sparse, and it is usual to solve it by means of a dense solver. On a GPU, this is easily parallelized and state of the art implementations are available, e.g., in CULA[10] or cuSOLVER[11] packages. The speedup of dense matrix decomposition on a GPU, compared to the CPU version, is nearly two orders of magnitude (see the upper half of Table 2), which is deemed sufficient in production environments.

On the other hand, the speedup of the sparse GEMM kernel is not as good, as can be seen in the lower half of Table 2, specifically the rows with cuSPARSE results. The consumer GPU is somewhat slower than the CPU implementation and the high-end GPU is only marginally faster. This stems from the fact that the CPU implementation is taking advantage of the block structure [53], while the GPU implementation is working at the level of sparse matrix elements. To improve this, we implemented a custom sparse matrix multiplication kernel on GPU [55] and further modified it to work with block matrices. Although it is only a proof of concept, the implementation already outperforms the CPU by about a factor of 2.

B. Quality Assurance in 3-D Reconstruction From Stills

Once the 3-D reconstruction has been calculated, it is possible to visualize it. Thanks to the above-mentioned optimizations, it is possible to reconstruct large scale scenes on a high-end laptop in a matter of tens of minutes, and thus, it is possible to inspect the reconstruction on-set. However, some of the shortcomings might not be immediately apparent, especially if the scene geometry is complex. Fortunately, by defining the underlying optimization problem as maximum likelihood estimation (MLE), we have information theoretic error metrics at our disposal.

The usual metric in statistics and information theory is covariance (resp. marginal covariance). However, recovering the covariance matrix Σ involves inverting the system matrix Λ, and while Λ is sparse in BA, Σ would be completely dense. This is not only problematic because of the computational cost, but also because of the storage requirements. The matrices routinely encountered in BA are hundreds of thousand elements square, which corresponds to roughly 74.5 GB of memory which is not available nowadays even in high-end laptops. Fortunately,

[10]Available at http://www.culatools.com/.
[11]Part of CUDA 7 and higher, http://developer.nvidia.com/cuda-zone.

not all of the elements of the covariance matrix are needed in order to display the 3-D reconstruction precision. In [56], it was shown how specific elements from the covariance matrix can be efficiently calculated from $R = \mathrm{chol}(\Lambda)$ by applying the recursive formula

$$\Sigma_{ii} = \frac{1}{R_{ii}}\left[\frac{1}{R_{ii}} - \sum_{k=i+1,R_{ik}\neq 0}^{n} R_{ik}\Sigma_{ki}\right] \tag{10}$$

$$\Sigma_{ij} = \frac{1}{R_{ii}}\left[-\sum_{k=i+1,R_{ik}\neq 0}^{j} R_{ik}\Sigma_{kj} - \sum_{k=j+1,R_{ik}\neq 0}^{n} R_{ik}\Sigma_{jk}\right]. \tag{11}$$

In case R is sparse, the above formulas can be used to compute the elements of Σ at the positions of nonzero elements in R very efficiently [57]. To compute multiple elements of the covariance matrix, such as the whole block diagonal, these formulas become inefficient unless all the intermediate results are stored. Our implementation performs the calculation of (11) in blockwise manner, same as the linear solving, yielding considerable gains in computation speed.

In order to visualize the 3-D reconstruction quality, the block diagonal of the covariance matrix, which corresponds to the covariances of the individual cameras and 3-D points, is calculated. Cholesky decomposition of the individual diagonal blocks is performed, yielding a matrix with the coordinate basis of an ellipsoid enveloping the uncertainty of each respective variable. If the scale of the reconstruction matches the physical scale, it would be possible to calculate the precision of each 3-D point estimate, e.g., in inches, for instance. For displaying a false color point cloud such as the one in Fig. 11, the squared norm of the matrix is taken instead and the resulting value is used to look up a false color in a suitable palette. Visualization of marginal covariances was not previously attempted, as the cost of the algorithms required to compute them using conventional algorithms was prohibitive [50].

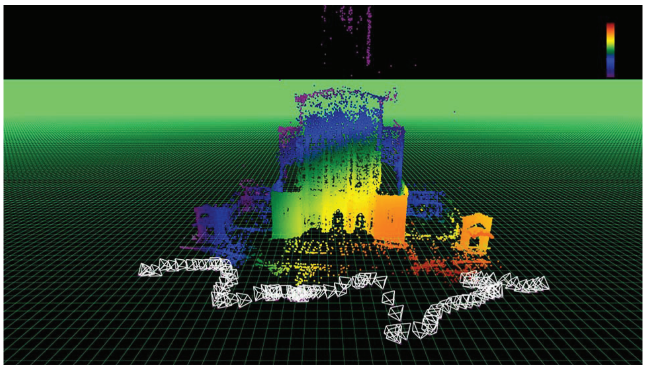


Fig. 11. ***Color-coded 3-D reconstruction Quality Assurance of the Surrey Cathedral. Orange colors correspond to low covariance, blue colors correspond to high covariance. There should be additional capture to the blue parts, in order to reduce covariance of the solution. The white pyramids are the camera poses.***

C. Dense Reconstruction From Spherical Stereo Images

Furthermore, we accelerate the dense 3-D reconstruction from spherical stereo scans, originally described in [37]. The spherical stereo consists of two spherical images unwrapped into latitude-longitude format, captured with vertical displacement and the result of the reconstruction is a spherical depth map of the same size as the images. The corresponding columns in the two images capture the same portion of the scene from a different viewpoint and they are sufficient to calculate one column of the dense depth map. The method of [37] consists of dense disparity estimation by constrained block matching followed by regularization by a PDE. The former one is computationally bound while the latter is memory bound. To accelerate block matching, parallelization is employed: each column can be treated separately.

The PDE in the original version of the algorithm is implemented as a series of image processing primitives, each reading and writing the results to a separate image. This is highly inefficient with respect to the memory traffic and usage of cache as the images are too large to fit in it; so by the time the next operation starts reading the image, the data must be fetched from RAM. To optimize this, the operations are formulated as partial functions and chained together in such a way that query to a result pixel value recursively calls the operations and performs all the computation for that single pixel at once. This significantly improves locality of references, as the input image is now read from and written to RAM only once, the rest of traffic being facilitated by cache and CPU registers.

However, the implementation of local filters required for partial derivative calculation causes repeated computation of the overlapping source pixels. In some cases, the repeated computation is faster than storing the intermediate result in memory. However in our case, increased throughput was gained by implementing a transparent tiled cache as another partial function. This way, a small patch of the intermediate image is computed once and then the local filter can read from the tile, thus saving computation.

Fig. 12 shows a plot of relative speedup of the entire processing pipeline based on the number of cores (this was measured on 16-core Xeon platform). We can see nearly linear scaling up to 8 cores and then sublinear scaling to 16 cores, where both CPUs of this NUMA system become utilized and some extra communication overhead ensues. The entire processing time was reduced from over 14 min to less than 1.5 min, or by a factor of 11.3.

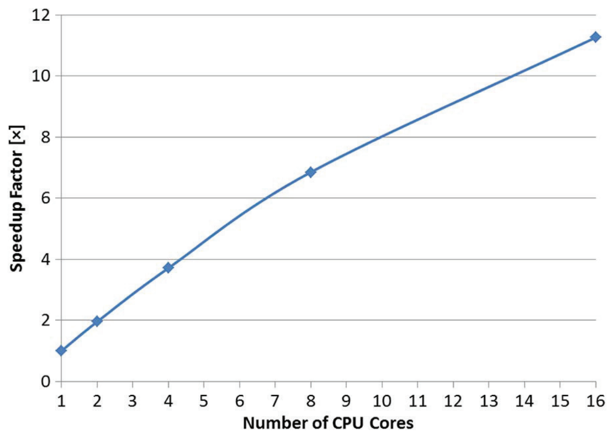


Fig. 12. ***Performance scaling of accelerated dense reconstruction from spherical stereo images.***

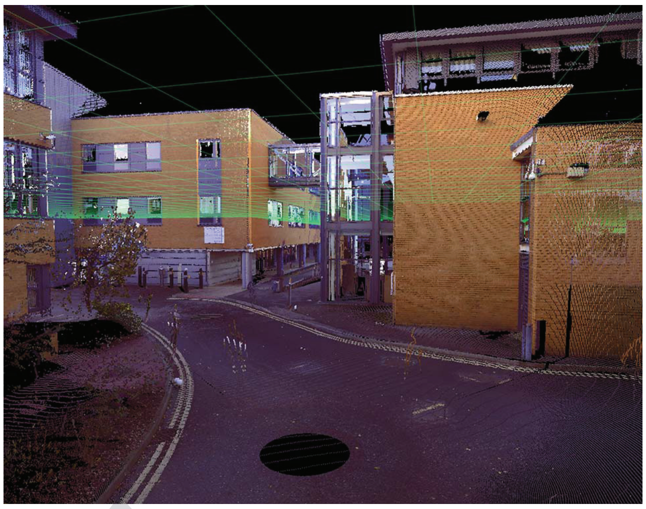


Fig. 13. ***GraphViewer displaying a LIDAR scan of the Plaza Scene in the IMPART dataset.***

To enable reconstruction of larger areas than a single spherical stereo scan can cover, an algorithm for 3-D reconstruction from *multiple* spherical images was developed [58]. It is based on combined RGB and depth interest point matching, and consists of spherical image feature extraction, matching, initial estimation and optimization based on graph-SLAM process, which can be calculated using the same accelerated algorithms as the BA in the previous subsection.

Multiple experiments have been performed in order to evaluate the performance of the proposed 3-D reconstruction from multiple spherical images. To analyze the accuracy of the proposed technique, ground truth was measured on the IMPART datasets. The precision and time of the reconstruction was compared to that of the ICP algorithm. Both ICP and SLAM have similar performance in terms of accuracy, but the main advantage of the SLAM approach is in the processing time: it is almost three orders of magnitude faster than the ICP algorithm, for all the tested datasets.

D. 3-D Reconstruction Visualisation

While our web-based on-set visualization is presented in Section VI, we have also implemented an offline visualization solution based on a subset of OpenGL 4 and OpenGL ES 2.0, for testing purposes. In the basic display mode, it shows the reconstructed point cloud, either with colors where available (see Fig. 13), or just in red otherwise. It can also display graph edges, for coverage and debugging purposes. It supports camera or 3-D point selection, by clicking on a specific camera, only the edges of that camera are displayed. Edges of multiple cameras can be displayed as well, by holding ctrl or shift while making the selection.

To allow viewing of very large scenes, a streaming mechanism was implemented. The data is split into several blocks which individually fit in the GPU memory, while each block contains a set of 3-D point positions: their corresponding colors, and a portion of the edges of the graph, referring to these points. Each block is then sent for rendering separately and then deleted from GPU memory. This allows displaying 3-D scene of any size while only sacrificing the rendering speed.

V. SEMANTIC MOVIE CONTENT ANALYSIS

The huge amounts of multimedia data which arise during shooting have to be managed, stored and integrated in appropriate ways for subsequent processing, postproduction and footage archiving. Cinema production is demanding not only on-set but afterwards too. Important requirements include fast, efficient ways of searching, browsing and analyzing footage, towards automatic generation of semantic metadata. Video summarization and fast content analysis algorithms for big media data are some of the techniques needed to meet these needs.

Film content analysis research efforts originally focused on exploiting text-based approaches. Thus, film audio description scripts and screenplays constituted the basis of most of the methods developed, up until the previous decade. On the other hand, analysis of low-level visual features was widespread for other types of content like sports and surveillance CCTV footage, while several attempts had also been made to exploit higher level semantic visual features. However, semantic film content analysis by exploiting visual features or combining them with audio ones is gradually becoming prevalent, bridging the semantic gap that existed when only text-based approaches were employed [59].

In film production, semantic content analysis of big media data is mainly used in the ingestion, postproduction and footage archiving stages of the pipeline (see Fig. 14), by adding human-related or other semantic annotations to content, thus enabling novel functionalities for fast browsing and preview of footage, as well as retrieval of the most relevant streams out of the entire available footage.

Technologies that have significant impact in this workflow include temporal video segmentation exploiting semantic information, video segment clustering towards video summarization and batch processing, face detection/recognition/clustering and visual analysis based on production-related specific activities, as discussed in the following paragraphs.

One approach for performing temporal video segmentation is to segment the stream according to the depicted human activities. This can be accomplished by applying recursively activity-based temporal video segmentation techniques in order to detect changes in the depicted activity and split content accordingly. Activity change detection can be achieved by employing activity related video descriptions and measuring their dispersion on overlapping video segments, as described in Section V-A.

Fast browsing within takes, dailies previewing and batch processing of similar takes constitute real needs in film production. These needs can be successfully catered for with clustering techniques, forming clusters of similar takes based on some type of semantic information (e.g., the displayed activity). In the technique briefly described in Section V-A, descriptors widely used in human action recognition are employed for the description and subsequent clustering of the activity segments created using the temporal video segmentation method mentioned above. In the multiview video case, the additional information is also exploited and view-independent action representation is achieved.

In general, once video segments, frames or frame regions have been represented by appropriate feature vectors, one can apply clustering or classification upon them. Many clustering or classification algorithms cannot deal with big data without proper adaptation. These algorithms often involve the construction of a similarity matrix of all the available training vectors. A novel kernel matrix trimming algorithm, which aims to both increase the performance of baseline kernel k-means clustering [60] and reduce the number of nonzero kernel matrix elements, thus accelerating the iterations of kernel k-means and requiring less memory, is presented in Section V-B and is an example of a fast and scalable technique for big media data analysis. A distributed implementation of this algorithm, that utilizes the Map-Reduce programming model and allows the fast processing crucial for the movie industry, is also presented in the same section. These approaches have been applied and evaluated in the task of facial images clustering in large scale datasets.

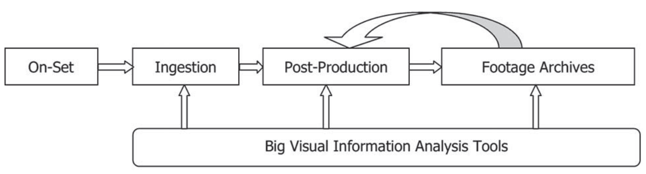


Fig. 14. ***Semantic content analysis in cinema production.***

Big visual data classification problems, such as face or human activity recognition often appear in movie production and postproduction. A simple approach to deal with the vast amount of training data is to model each class of the population separately, by employing an ensemble of one-class classifiers. However the training data for each separate class can still be huge. Approximate approaches can be used to overcome this problem. A novel approximate solution for least squares one-class support vector machines is presented in Section V-C. The performance of the approach is experimentally evaluated on face recognition in a large facial images dataset.

The approaches presented can also be applied to big media data found in other application domains such as large media archives, surveillance etc. In addition, the clustering and classification techniques can be also applied to all sorts of big data where individual samples can be represented in vector form.

A. Activity-Based Temporal Video Segmentation and Clustering

Understanding/analyzing human activities in video is vital in many applications related to media production and postproduction, such as video summarization, highlight extraction, event detection or content-based annotation [61]. This problem has been primarily approached by applying action/activity recognition techniques on video data from one [62] or multiple [63] cameras. One of the disadvantages of action analysis within a recognition setting is that the set of all possible human actions should be *a priori* defined and an adequate number of (labeled) action videos should be available for the training of the involved classifiers. In several application scenarios, e.g., in movie production/post-production and content-based video retrieval, the objective is to temporally delineate the different action patterns and perhaps cluster them into sets of similar actions, rather than to perform a strict characterization (i.e., recognition) of the observed actions. Indeed, temporal video segmentation into meaningful segments, as well as video segment clustering can be important steps in the production and postproduction processing chain, since they allow automatic semantic annotation of the video segments for fast footage ingestion, archiving and retrieval, all being instrumental due to the huge volumes of data.

Three methodologies have been widely used [64] for temporal action segmentation, namely the sliding window, boundary detection and grammar concatenation ones. The *sliding window* approach (e.g., in [65]) divides a motion sequence into multiple (usually overlapping) video segments. The success of this approach strongly depends on the discrimination ability of the employed action video representations. *Boundary detection* methods [66] generally search for discontinuities in observed human action videos. Boundaries usually imply a basic action taxonomy, without depending on explicit class definitions. The use of *grammars* in temporal action segmentation originates from speech recognition. The mainstream approach is to model state transitions between action states using hidden Markov models (HMM) along with some method for action feature generation, such as dynamic system representations [67], geometrical property encoding [68], curvature scale space, centroid distance function [69], etc. Moreover, two models of increasing popularity are change-point detection [70], [71] and switching linear dynamical systems (SLDS) [72].

In the following, we will summarize a novel semantic temporal video segmentation approach based on human activity information and the Fisher discriminant analysis. The approach goes beyond the standard shot boundary (shot cut) detection applied on video data, employs a state-of-the-art video representation and can be applied both to single-view and multiview video data. The derived temporal video segments can then be clustered into clusters. Each such cluster contains video segments depicting similar human activities and can be semantically annotated according to these activities. Moreover, face detection and face recognition can be performed within each segment extracting additional actor identity-based annotation of the footage.

For action representation within the proposed method we have employed the Dense Trajectories-based action video description [73] combined with the bag of words (BoW) model, since it has been shown to provide state-of-the-art performance in the related task of human action recognition. In Dense Trajectories video description, each video is described by using a set of five descriptors calculated along the trajectory of interest points that are tracked for a number of L consecutive video frames.

More specifically, in order to perform temporal segmentation of an action video possibly depicting several consecutive actions, we employ the Dense Trajectories description, in order to calculate descriptors $\mathbf{d}_i^v$, $i = 1, \ldots, P_d$, (P_d: number of interest points detected in the video) $v = 1, \ldots, V$ $(V = 5)$ on the trajectories of densely-sampled video frame interest points of the entire action video sequence. We then apply k-means clustering on $\mathbf{d}_i^v$, in order to calculate a set of descriptor prototypes (codebook). By using this codebook, which is exclusively derived from the video under consideration, and the video frame indices corresponding to each trajectory, we create BoW-based representations of M (overlapping) video segments for each descriptor, denoted by $\mathbf{b}_i^v$, $i = 1, \ldots, M$, $v = 1, \ldots, V$. Subsequently, we concatenate the five BoW vectors $\mathbf{b}_i^v$, $v = 1, \ldots, V$, in order to fuse the information appearing in each trajectory, i.e., $\mathbf{b}_i = [\mathbf{b}_i^{1\,T}, \ldots, \mathbf{b}_i^{5\,T}]^T$. Overlapping video segments consisting of T_v video frames (e.g., $T_v = 20$) having overlap of $T_v - 1$ video frames are then created. Let us denote by N the number of frames in a video split into M video segments and S be the set of the $\mathbf{b}_j$, $j = 1, \ldots, M$ BoW-based representations of the resulting video segments. Through the temporal relationship of the video segments, we create two sets of video segment representations $\mathcal{S}_i$, $i = 1, 2$, each consisting of M_i, $i = 1,\ 2$ vectors, where $M_1 + M_2 = M$. By employing $\mathbf{b}_j$, $j = 1, \ldots, M$ and the corresponding set labels c_j, the within-set and total variance can be respectively measured by

$$s_w = \sum_{i=1}^{2} \sum_{j, c_j = i} (\mathbf{b}_j - m_i)^T (\mathbf{b}_j - m_i) \tag{12}$$

$$s_T = \sum_{j=1}^{M} (\mathbf{b}_j - m)^T (\mathbf{b}_j - m) \tag{13}$$

where

$$m_i = \frac{1}{M_i} \sum_{j, C_j = i} \mathbf{b}_j, \quad m = \frac{1}{M} \sum_{j=1}^{M} \mathbf{b}_j. \tag{14}$$

By combining s_w, s_T, we obtain the *Fisher criterion* $J = s_w / s_T$ [74]. Since $\mathbf{b}_j$ represent the video segments in the video to be segmented, the minimization of J leads to the maximization of the compactness of the two video segment sets $\mathcal{S}_1, \mathcal{S}_2$. The optimal temporal segmentation of the video is performed by finding the minimum of $J(h)$, $h = 1, \ldots, H$, where H denotes the number of possible bisections of S, as described above. To this end, we employ a line search strategy for the determination of the best temporal segmentation position $h_m (1 \leq h_m \leq H)$. The above-described process is illustrated in Fig. 15.

In order to treat a long sequence, we extend our method of line search strategy over the two resulting video segments in a recursive way, by computing the Fisher criterion on each video segment. The procedure stops when a minimum video segment length has been reached.

The method can be also applied in the case where action instances are depicted in multiple synchronized videos, each captured from a different viewpoint. Indeed, if temporal segmentation in the different views has been adequately accurate, one can employ a majority voting over all camera segmentation timelines (namely the sequences of video frame labels, where each label denotes the video

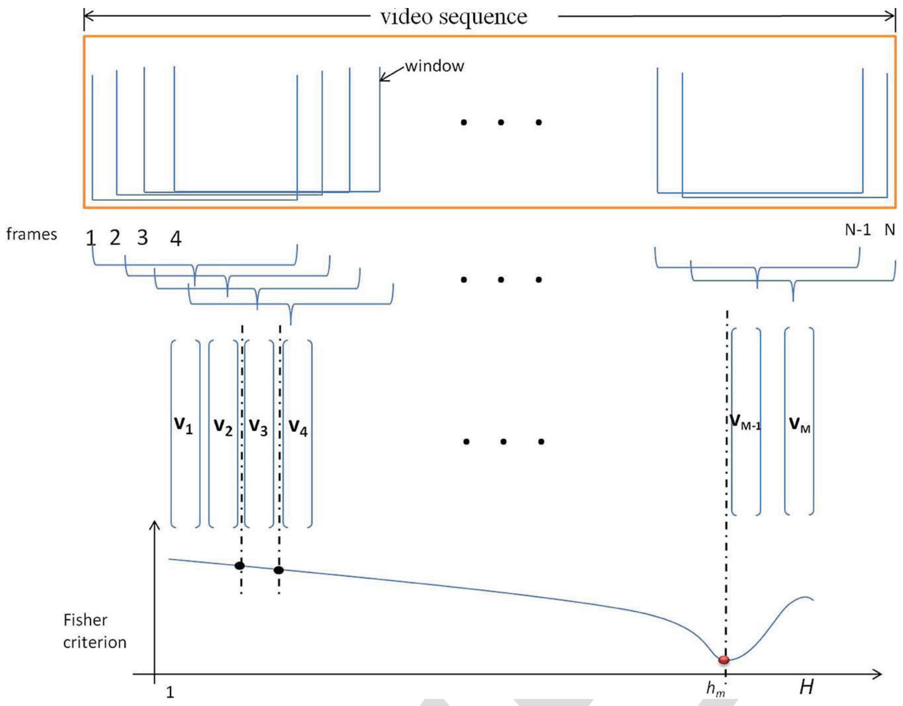


Fig. 15. *Determination of the temporal segmentation point.*

segment it belongs to) in order to create a single "global" segmentation timeline for the multiview video.

Once one or more videos have been temporally segmented into N_I segments using the procedure outlined above, video segment clustering can be performed in order to group them into K clusters containing similar actions. For this, we apply clustering on the BoW-based video representations $\mathbf{x}_i^v$, $i = 1, \ldots, N_I$ by employing kernel k-means algorithm. In order to combine the different action properties described in different BoW-based representations $\mathbf{x}_i^v$, we employ the RBF-χ^2 kernel, where different descriptor types are combined in a multichannel approach [75]. If by the end of this procedure one manually sets labels (e.g., walk, run, jump) in each cluster (in the case where the derived clusters are fairly homogeneous), each action video can be assigned the corresponding action cluster label. For clustering multiview videos, one can exploit the circular shift invariance property of the discrete Fourier transform (DFT) coefficients [76] in order to obtain a view-independent action representation.

B. Distributed Trimmed Kernel K-Means Clustering

The objective of *data clustering* is to divide a given group of unlabeled data samples in subgroups (*clusters*), so that data samples belonging to the same cluster are similar to each other and dissimilar to samples belonging to any other cluster. Clustering has many applications in different scientific fields. Despite the fact that there has been an extremely rich bibliography on this subject for years [77], it is still an active research field.

One of the earliest clustering methods is the k-Means algorithm [78] that is still popular, despite its age. Its main drawback is that the surfaces separating the clusters can only be hyperplanes. Thus, if the clusters are not linearly separable, the standard k-Means algorithm will not be able to give good results. In order to overcome this limitation, the classical algorithm has been extended into the kernel k-Means [60]. The basic idea behind kernel approaches is to project the data into a higher, or even infinite dimensional space. It is possible for a linear separator in that space to have a nonlinear projection back in the original space, thus solving the nonlinear separability issue. The *kernel trick* [79] allows us to circumvent the actual projection to the higher dimensional space. The trick involves using a *kernel function* to implicitly calculate the dot products of vectors in the kernel space using the feature space vectors.

A convenient way to have quick, repeated access to the dot products without calculating the kernel function every time, is to calculate the function once for every possible combination of data samples. The results can be stored in a $n \times n$ matrix $\mathbf{K}$ called the *kernel matrix*, where $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$.

Kernel k-Means provides a popular starting point for many state of the art clustering schemes [80]–[83]. A recent survey on kernel clustering methods can be found in [84].

Distributed computing can provide the means to handle problems on very large datasets, often encountered in media production, that would otherwise be almost impossible to solve [85]. Provided that a task can be split into many independent subtasks, then it can theoretically be performed in a reasonable amount of time, regardless of the data size, given enough processing units.

Distributed versions of clustering algorithms related to kernel k-Means, like classic k-Means [86] and k-Medians [87] have already been proposed. However, to the best of our knowledge, a distributed approach to kernel k-Means has not been proposed yet. Such an approach that also involves kernel matrix trimming is summarized below. Full details and experimental evaluation are provided in [88].

The approach follows the MapReduce programming model [89], which is a high level framework for distributed processing on a computing cluster. The implementation uses Apache Spark [90], a cluster computing framework, which is similar to and compatible with Hadoop [91]. The computing cluster can include a wide variety of hardware from high-end, multiprocessor computers with large amounts of RAM, to average recent PCs. The focus of the proposed implementation is to avoid the need to store n^2 kernel matrix entries into the distributed memory at the same time, if possible. In order to achieve this goal, we employ a novel kernel matrix trimming algorithm, which enables us to significantly reduce the number of nonzero entries in the kernel matrix, while also increasing clustering performance. The proposed distributed clustering scheme is divided into three major parts: kernel matrix computation, kernel matrix trimming algorithm and, finally, kernel k-Means itself. The experimental evaluation of the proposed approach was performed on the facial image clustering task, which is important in movie content analysis and description, as it allows us to search for actor face appearances in video content.

In more detail, we consider the kernel matrix entries to express data sample facial image similarity. These entries have large/small values within the same cluster/between different clusters, respectively. We aim to eliminate (trim out) small K_{ij} entries, while retaining as many large K_{ij} entries as possible. In the presented algorithm, it is possible to retain a different number of entries K_{ij} for different data samples.

In general, the proposed kernel matrix trimming algorithm attempts to determine the cardinality of the cluster that a data sample belongs to, through a voting system. Each data sample casts votes on the various candidate cluster cardinalities for itself. The votes for each cluster cardinality j are summed up for every sample. Each cardinality is then assigned a score by using a suitability function that measures how close the number of votes for j is to the nearest integer nonzero product of j. The winning cardinality is the one with the highest score. Every data sample that voted for the winning cardinality value is determined to belong to a cluster of that cardinality

When there are no more votes, every data sample has received an estimate of the cardinality of the cluster it belongs to. The trimming of the kernel matrix $\mathbf{K}$ entries is performed in a row-wise manner. Suppose that the estimated cluster cardinality for data sample a_i is w_i. We zero (trim out) every entry K_{ij} in the ith row of $\mathbf{K}$ whose value is less than the w_ith largest value of the row. The resulting matrix $\hat{\mathbf{K}}$ may no longer be symmetric, thus the final trimmed similarity matrix is obtained as $\mathbf{K}^* = \max(\hat{\mathbf{K}}, \hat{\mathbf{K}}^T)$.

For the distributed implementation of the above mentioned clustering algorithm the *MapReduce* distributed computing programming model [92], whose implementations include Hadoop and Spark, has been utilized. MapReduce simplifies the coding of distributed programs and was specifically developed to allow easy processing of very big datasets on computing clusters. A master node in the MapReduce framework automatically splits the dataset up into smaller data sample collections and distributes them to the workers, each processing the assigned data independently.

As the name implies, there are two major components to this programming model. With the *Map* command, every worker applies a user defined function to each data sample. Each worker can then return the results to the master node. Additionally, with the *Reduce* command, a worker applies a commutative and associative operation to collect the data elements, or the results of a previously mapped function, into a single result. As the operation is commutative and associative, the results for each worker are independent from those of other workers and they can also be combined in the same way on the master node.

All three parts of the proposed trimmed kernel K-means algorithm, namely the kernel matrix computation, its trimming and the actual kernel K-means have been cast in a MapReduce framework. For example, the computation of the kernel matrix proceeds as follows. Assuming there are n data samples, each of which has d features, we read the data samples into n d-dimensional data vectors, which are distributed to the cluster worker nodes. Then we iterate through every data vector and map the kernel function of the current vector with every other vector. This provides us with a single row of the kernel matrix, which we can then write to the disk. After n iterations, the computation is complete. This step requires $O(nd)$ distributed memory and $O(n^2d)$ operations [88].

The performance of the proposed trimmed kernel k-means approach and its distributed implementation have been judged on a number of experiments. The first experiment involved the MNIST handwritten digit dataset. In this experiment we used the *Normalized Mutual Information* (NMI) metric [93] to measure the similarity between the clustering results and the ground truth while the reduction in the size of the kernel matrix has been measured with the ratio nz/n^2 of the nonzero elements of the kernel matrix after trimming to the number of elements of the full matrix. In this experiment, the proposed trimmed kernel k-means approach utilizing an RBF kernel provided the best clustering performance in terms of NMI (0.5687), compared to 0.4936 obtained by kernel k-means while retaining only about 4% of the full kernel matrix elements. By comparing the proposed approach to approximate kernel *k*-Means [94] on the same dataset it was found that the latter needs about 7% of the full kernel matrix, in order to match the full kernel matrix performance (0.4941), while our approach achieves better performance (0.5687) with about the same kernel matrix size.

Tests were also conducted in order to check the effect of the number of processing/computing nodes in a distributed processing environment to the processing time of the proposed distributed implementation of the clustering algorithm. The plot of computation time with respect to number of computing nodes is expected to ideally have the form of the rectangular hyperbola $f(x) = 1/x$. Experiments to verify this were conducted on the Youtube Faces dataset [95] that certainly qualifies as big data since it consists of local binary patterns (LBP) descriptors [96] for 621126 faces of various celebrities extracted from Youtube videos. Since it was not practical to run the kernel matrix computation in its entirety for various numbers of cores, as it would take about 150 days for a single core to finish the task, we measured the time required by the computing cluster to calculate 50 rows of the kernel matrix. The computing cluster consisted each time of a different number of Virtual Machines (VMs) as workers, each VM having two cores and four Gigabytes of memory. The resulting acceleration curve can be seen in Fig. 16 and indeed reasonably follows the predicted rectangular hyperbola. In total, the algorithm (kernel matrix computation and trimming, kernel *k*-means itself) for the entire dataset required about 14.21 days on 12 cores compared to the 150 days on a single core mentioned above.

C. Approximate Methods for Big Media Data Classification

Large scale visual data classification problems, including face recognition, activity recognition and video shot-type characterization, commonly appear in a movie production and postproduction stage. For such problems, the state-of-the art approach is to use classification methods that produce nonlinear decision functions by employing a nonlinear piece-wise mapping function to map the data from the input space to a feature space of higher dimensionality. In order to express data similarity in the feature space, one can employ the kernel trick [79], where similarity is expressed with the kernel matrix $\boldsymbol{K} \in \mathbb{R}^{N\times N}$, N being the number of the available training data. The derived solutions for state-of-the-art methods, such as Kernel Principal Component Analysis [97], Kernel Ridge Regression [98], regularized neural networks [99], least-squares support vector machines [100], or randomized neural networks (also referred to as extreme learning machines [101]), involve the eigen-decomposition or the inversion of $\boldsymbol{K}$. Thus, in classification problems involving big visual data, where N is very large, the application of such approaches can be prohibitive, since the theoretical computational complexity and memory requirements are of order $O(N^3)$ and $O(N^2)$, respectively. As the number of the available training data increases with the number of the classes to be modeled, one simple approach to the issue above could be to try to model each class separately, by employing an ensemble of one-class classifiers, each modeling only one class. It has been shown that such an approach can achieve very good performance in large-scale multiclass biomedical data classification problems [102]. Furthermore, one-class classification methods show good performance when only the class of interest needs to be modeled and discriminated from the rest of the world. Moreover one-class approaches have been used in visual data classification problems, such as video surveillance and video summarization [103]. State-of-the-art one-class classification methods, such as the one-class support vector machines (OC-SVM) [104], the support vector data description [105] and least squares one-class support vector machine (LS-OC-SVM) [106], achieve significantly

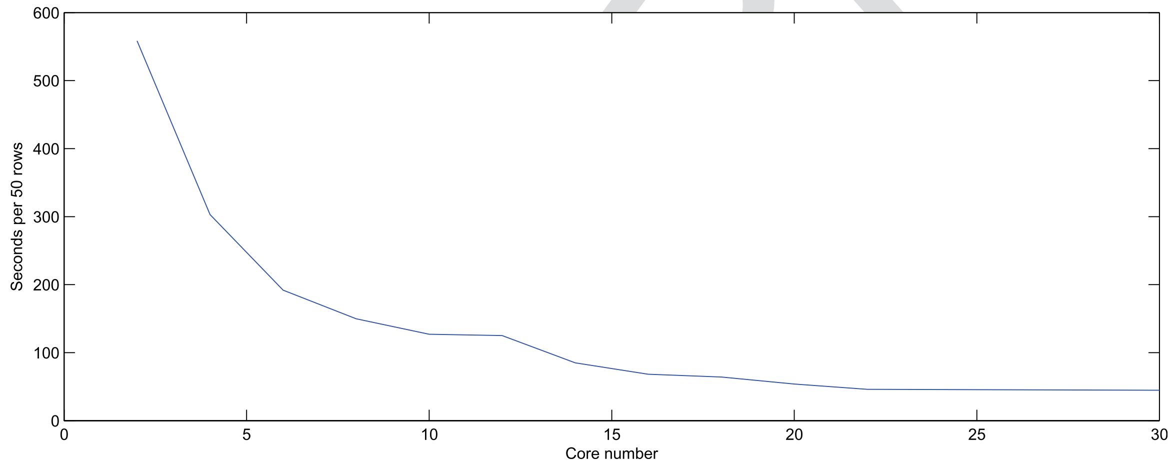


Fig. 16. ***Seconds to compute 50 rows of the kernel matrix with respect to number of cores.***

better performance in their kernel version over their linear alternatives. Two up-to-date relevant reviews can be found in [107], [108].

However, in the big data case, even the number of training data for each separate class can be enormous. In order to overcome such restrictions of kernel methods, kernel matrix approximation approaches have been proposed [109]–[111], based on the Nystrom method. In this line of work, a rank n approximation of the kernel matrix $\boldsymbol{K} \in \Re^{N\times N}$, can be obtained by random (column) sampling as $\boldsymbol{K} \approx \tilde{\boldsymbol{K}}\mathbf{B}^{\dagger}\tilde{\boldsymbol{K}}^T$, where $\tilde{\boldsymbol{K}} \in \mathbb{R}^{N\times n}$ contains the n sampled columns and $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ is the kernel matrix of the training data corresponding to the n sampled columns. Thus, only a reduced number of data similarities need to be calculated and stored, leading to lower computational and memory demands. Although matrix approximation methods can be used in every kernel based learning method with decent results, this may not always be the best possible approximation option.

In support vector machines, the decision hyperplane can be expressed as a linear combination of the support vectors [104], which are expected to be fewer than the training data. Therefore, a method that approximates the extreme points (that are more likely to be the support vectors) has been proposed in [112]. This can lead to decreased memory requirements and reduced computational complexity, without sacrificing classification performance. Other approximation methods include the fast determination of a k-nearest neighbor (kNN) graph, using k-dimensional trees or local sensitive hashing [113], [114].

Towards this end, a novel approximate solution for least squares one-class support vector machines has been introduced. In order to be effectively approximated, the solution is restricted to be a linear combination of a subset of the training data in the feature space, by employing a randomization approach. Moreover, in order to model the geometric class data information in the optimization process, specific regularizers based on the data similarity graphs can be included and can be implemented without increasing the computational complexity and memory requirements dramatically. The proposed one-class classifier is designed to be employed in large scale visual data classification problems, where each class can be modeled independently to be distinguished from the rest of the world.

Let us denote by $\boldsymbol{\Phi} \in \Re^{|\mathcal{F}|\times N}$ a matrix that contains all training data representations in the feature space, such that each row of $\boldsymbol{\Phi}$ contains $\phi(\boldsymbol{x}_i) \in \mathcal{F}$. Also let $\boldsymbol{K} = \boldsymbol{\Phi}^T\boldsymbol{\Phi}, \boldsymbol{K} \in \Re^{N\times N}$ be the corresponding kernel matrix that contains the training data similarities in the feature space $\mathcal{F}$. In the case where many training data exist, N is huge, so that employing the kernel version of OC-SVM may be computationally impossible. In order to obtain a OC-SVM specific approximate solution, we consider to obtain an approximate solution for the hyperplane $\boldsymbol{w}$. Based on the Representer Theory [115], the separation hyperplane $\boldsymbol{w}$ can be expressed as a linear combination of the training data, by employing a reconstruction vector $\boldsymbol{a} \in \Re^N$, such that $\boldsymbol{w} = \boldsymbol{\Phi a}$. In order to obtain an approximate version of kernel OC-SVM, we restrict $\boldsymbol{w}$ to be a linear combination of fewer (n) training data, such that $n \ll N$, where n is the number of sampled elements, obtained, e.g., by random sampling. Thus, the approximate version of $\boldsymbol{w}$ can be expressed as follows:

$$\tilde{w} = \tilde{\boldsymbol{\Phi}}\boldsymbol{a} \tag{15}$$

where $\tilde{\boldsymbol{\Phi}} \in \Re^{|\mathcal{F}|\times n}$ contains the sampled data representations in $\mathcal{F}$. By using (15), the approximate kernel OC-SVM (AOC-SVM) optimization problem is formulated as follows:

$$\text{Minimize} \quad \frac{1}{2}\boldsymbol{a}^T\tilde{\boldsymbol{\Phi}}^T\tilde{\boldsymbol{\Phi}}\boldsymbol{a} - \rho + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i \tag{16}$$

$$\text{Subject to} \quad \boldsymbol{a}^T\tilde{\boldsymbol{\Phi}}^T\phi(\boldsymbol{x}_i) \le \rho - \xi_i \tag{17}$$

$$\xi_i \ge 0. \tag{18}$$

The above optimization problem can be solved by finding the saddle points of the corresponding Lagrangian which leads to a quadratic programming optimization problem. By substituting the Hinge loss in (16)–(18) with the squared loss, we formulate the approximate one-class least-squares SVM (AOC-LS-SVM) objective which leads to a linear system of equations instead of a quadratic programming problem and can be easily solved. The extension to two-class or multiclass approximate optimization objectives is straightforward.

Experiments have been conducted in order to evaluate the performance of the proposed variants of LS-OC-SVM classifier and compare them to other state-of-the art classification methods, as well as approximation methods. More specifically, we compared the approximate method with the OC-SVM classifier [104], its approximate version using the Nystrom method (AOC-SVM) [104], [110], the standard LS-OC-SVM [106], its Nystrom approximate variant (NY-LS-OC-SVM) [106], [116] and another approximate LS-OC-SVM version (ALS-OC-SVM). Three variants of the LS-OC-SVM classifier using geometric information were included in the conducted experiments, namely the ones that employ the total class scatter (GE-LS-OC-SVM-T), the within class scatter with respect to subclass information (GE-LS-OC-SVM-W), and the kNN type graph (GE-LS-OC-SVM-KNN). We evaluated the performance of the classifiers in terms of classification performance and training time. For classification performance, we have employed the g-mean metric [117], which is geometric mean of the accuracy for data samples belonging to the modeled class (positives) and outliers (negatives). In one such experiment we used the YouTube Faces dataset [118] already mentioned in Section V-B.

In this experiment we have employed a test protocol similar to the "restricted" protocol proposed in [118]. The classification performance and training times in seconds for this experiment are shown in Table 3. As it can be seen, the approximate GE-LS-OC-SVM has superior performance over the standard OC-SVM and LS-OC-SVM in all its variants. By setting the percentage of the used training samples $p \geq 0.05$, it can be seen that all tested approximate algorithms can match or improve the performance of OC-SVM and LS-OC-SVM. Moreover, even for values of $p = 0.05, 0.10$, we can obtain state-of-the-art classification performance with a significant time gain for all GE-LS-OC-SVM variants.

Experiments were also conducted on activity recognition and shot type characterization datasets, since these problems are relevant to movie production, and again verified the superiority of the approximate methods over the competition in large scale media annotation experiments.

VI. INTEGRATED WEB VISUALIZATION

As we have argued in Section III, the "unified 3-D space paradigm" represented in Fig. 1 enables a more efficient management of multiple source data, and is the basis of better quality monitoring and assessment of data captured on-set. Furthermore, the optimization and acceleration discussed in Section IV allows for real- or near real-time support for creative decisions, likely leading to important cost savings both in production and in postproduction.

In this section we add a further innovative tool, the integrated hybrid web visualization of different data sources and modalities. Indeed, the ability to register multiple data sources to a reference (LIDAR) system and to position the sensors generating the data discussed in Section III allows visual representation of the results of the "unified 3-D space." Our technique represents, within an interactive 3-D graphics environment, the different datasets and modalities—and even makes use of the metadata generated, such as, for instance, those discussed in Sections IV and V. This visual integrated tool provides support for human assessment of the quality issues, as users can understand better the whole picture, instead of the more traditional visualizations of each modality separately, which come from using different processing algorithms, and which tie the visualizations to the results of different software for each modality.

The visualization paradigm proposed is novel as well: The actual shooting environment is simulated, combining the LIDAR reference point cloud, with 3-D reconstruction from other sources, the actual simulation of those data sources themselves at the actual positions of the sensors; and using the metadata to filter as well some of the content—some human actions, for instance. Traditional data visualizations are abstract, leading mostly to dynamic graphics (and symbolic pictograms in the less abstract cases). Our paradigm supports a very concrete visualization, that of the actual sources and content. This is aligned with the current cinema professional practices, which needs to understand the actual content to assess some aspects of quality. This hybrid visualization is inspired by [119], which showed efficiency gains, but our paradigm proposes a much higher stage of integration.

Integrating the visualizations in a web environment can easily lead to on-set collaboration and annotation, which would be another advantage. But, on the other hand, we discuss and show below that the web environment is better to support multiple modalities integration than desktop based environments. On the other hand, the interactive very large 3-D graphics visualization poses a number of technical challenges, whose solution is discussed below as well.

Table 3 Classification Performance and Training Times in the YouTube Faces Dataset

method/percentage	0.01	0.02	0.05	0.10	0.20	0.50	1.00
OC-SVM [104]	-	-	-	-	-	-	96.32
	-	-	-	-	-	-	3.70s
OC-LS-SVM [106]	-	-	-	-	-	-	96.11
	-	-	-	-	-	-	7.01s
AOC-SVM [104], [110]	93.60	94.86	96.19	96.71	96.86	96.74	96.32
	2.07s	1.83s	1.61s	1.91s	2.02s	2.73s	4.78s
NY-LS-OC-SVM [106], [110]	92.73	95.23	96.02	96.16	96.02	95.90	96.11
	0.07s	0.07s	0.09s	0.16s	0.38s	2.28s	15.74s
AOC-LS-SVM	92.92	95.30	96.25	96.34	96.26	96.11	96.11
	0.02s	0.02s	0.04s	0.06s	0.16s	0.95s	7.30s
GE-LS-SVM-T	93.93	95.65	**96.40**	96.45	96.53	96.56	96.68
	0.02s	0.02s	0.04s	0.06s	0.14s	0.75s	5.86s
GE-LS-SVM-W	**94.24**	**95.92**	95.57	**96.68**	**96.69**	**96.69**	**96.79**
	0.09s	0.09s	0.13s	0.19s	0.49s	3.01s	22.18s
GE-LS-SVM-KNN	93.88	95.78	**96.40**	96.50	96.48	96.50	96.59
	1.61s	1.63s	1.58s	1.63s	1.73s	2.64s	9.15s

A. Web-Based Visualization

The advent of cloud technology has changed modern digital workflows considerably. Remote and collaborative workflows and web-based tools are now becoming common in the workplace, with users increasingly accessing applications from anywhere, and on any platform, to share and collaborate. This potentially includes both those professionals present on-set (a large crew with different roles, generating different modalities of data and metadata) and those working remotely, who have roles in the production and postproduction processes, visual effects, etc. There is a strong drive for any on-set visualization app to be web-based, as this requires no external software to be installed and, by its very nature, is suited for remote data access, for supporting collaboration. Nevertheless, supporting these requirements in the digital production world presents a series of difficulties:

a) the sheer volume of data (whether raw image data, processed 3-D data, or metadata) which is created is not very compatible with the concept of distributed visualizationn via the web (and particularly the mobile web), due to bandwidth constraints;
b) the wider access permitted by cloud and web-based tools means that there is a wider range of hardware devices capable of accessing it (from high powered desktop machines to mobile phones), a fact that any visualization must take into account;
c) the same wider access also poses open questions for data security, a critical issue in the world of cinema production.

Points (a) and (b) above affect directly a key consideration for all web-based applications: the speed of interaction, which is a key ingredient. The expectation of the user when opening a web page is very different to the traditional application experience (whether desktop or mobile). There is no explicit application startup or data processing information relayed to the user (the typical “startup splash screen” ubiquitous to many applications); once the user visits a url, their expectations are for an instantaneous response, yet one with an inconsistent or partial presentation. The classic case is that of a web-page loading and displaying text first, while any associated images appear later as and when they are loaded. Another example is that of streaming video, users no longer expect to download the entire file before beginning to watch it.

In parallel, combinations of 3-D and other modalities on the web have recently appeared, as exemplified by Jankowski *et al.* [119], [120], who presented an interface combining hypertext and simple 3-D graphics and showed that the performance with this so called “dual-mode” interface was better than for single modalities (even taking into account switches). Inspired by this paradigm, the visualisation work presented below goes beyond a simple dual-mode interface, in the sense that it is a truly hybrid interactive visualization, tightly integrating video, static image, hypertext and real-time 3-D graphics. In spite of the difficulties for creating such a web-based visualization (as pointed out above), the modern web browser is in fact a propitious context for creating a hybrid application, which combines several modalities. Apart from the ability to rapidly create attractive and adaptive user interface layouts (using CSS), it has a well established and stable system for downloading and streaming 2-D images and video data and, with the release of WebGL in 2011, a standard manner of creating hardware accelerated real-time 3-D graphics applications. Furthermore, the browser context encourages the parallel use of 2-D and 3-D contexts (in combination with audio and video streams), in order to create hybrid web applications which present data from dozens of potential sources.

The main goal of our visualization research is to make best use of these advantages, while overcoming the difficulties described above.

B. Progressive Visualization of 3-D Data

The release of the WebGL standard in 2011 has meant a qualitative change in web-based 3-D graphics. Now supported by all major desktop and mobile browsers, the WebGL API allows the browser to access hardware accelerated graphics without requiring 3rd party plugins, such as Unity3-D or Adobe Flash. WebGL can be programmed imperatively directly via the browser using Javascript, although several more declarative methods of programming Web-based 3-D have gained popularity in research [121], [122]. Web-based 3-D applications share many of the advantages common to all web-based technology, namely platform independence, no reliance on 3rd party software, and ease of distribution and maintenance. Using WebGL also allows a seamless integration of 2-D and 3-D, which facilitates the creation of powerful and innovative interfaces [119], [123], which would be more difficult to create with nonbrowser-based software. However, the difficulties inherent to many client-server based technologies, such as those relating to bandwidth and synchronisation, are particularly present in all web 3-D applications, due to the typically large file size of the assets used. For more information on the current state of the art in web-based 3-D graphics, we refer the reader to a recent comprehensive survey [124].

As explained in previous sections, the 3-D data generated by movie production is represented in both point cloud or reconconstructed 3-D mesh format. Visualizationn of such big 3-D data over the web presents several difficulties. The challenges relating to bandwidth, as described above, are quite clear; yet merely applying one of several well-understood mesh compression algorithms [125] is not necessarily the best course of action. The lack of computing power available to the browser context, due to the (semi)-interpreted nature of the javascript client-side API; and the restrictions of the

WebGL API itself, which has neither the power nor the flexibility of its OpenGL desktop brother, mean that the decompression time for several mesh compression algorithms can actually create a greater time bottleneck than the one due to available communication bandwidth [126]. This situation is best explained by Chun [127], whose "WebGL loader" has become a popular method to load large 3-D meshes over the web; it relies does on a custom file format which encodes a mesh using the UTF-8 string format to represent binary data that are compressed using a standard gzip algorithm. This approach takes advantage of decompression methods inherently available to all browsers (thanks to HTML standards) yet is limited, from a big data perspective, due to restrictions in the UTF-8 data format, which requires the mesh to be broken down into various smaller chunks. Behr *et al.* [128] adopted a different approach to the transmission of binary data, storing them in the pixels of lossless portable network graphics (PNG) format images. This also has the advantage of fast decompression and a further one of pushing much of the mesh reconstruction work to the GPU. Lavoue *et al.* [129] present a method for progressive mesh rendering, decimating the mesh using vertex collapse mechanisms and reconstructing on the fly. This method allows progressive reconstruction of the mesh in the browser client, which is a desirable user experience, as it provides an "instant" low-resolution view, before being refined stepwise.

A common problem facing all of these mesh transfer and visualization techniques stems from the WebGL standard, which permits only a 16-bit buffer. While extensions do exist to enable a 32-bit index buffer, they are not as widely supported as WebGL itself, particularly in mobile hardware, and cannot be relied upon to be present in each client's browser. Due to the fact that an index buffer is an essential requirement for any efficient transfer of mesh data, from a big data perspective, the restriction to a mere 65536 vertices is problematic, as it necessitates splitting the mesh into several smaller meshes.

As a result, for the visualization of the unified 3-D space presented in this paper, we chose to use coloured pointclouds almost exclusively. This avoids some issues regarding meshes, as indicated above, and also is somewhat logical, given hat the raw data is provided in (sometimes rather sparse) pointcloud format.

In order to visualize the point cloud correctly, we use a progressive transfer mechanism based on our previous work [130], [131]. To begin with, the point data is linearly normalized to its axis aligned bounding box, to 16-bit precision. Using a 30m set size for example, this provides a precision of less than half a centimeter, which we consider to be adequate for visualization. In practice, an even lower resolution could be used, but the 16-bit normalization provides a useful structure with which to manipulate data quickly using Javascript Typed Arrays in the client application. Color data can also be normalized, if required, though in the work presented in this paper we use standard 24-bit RGB color. In an offline process, we add the entire set of points to a memory efficient octree. The color of each node of the tree is calculated as an average of the colour of all the nodes below it. The tree is then saved breadth first as a series of binary files, which are stored on a web-server. The point cloud is now ready for transmission to the browser client.

When the user arrives at the relevant URL, the browser starts downloading, sequentially, the list of files which contain the breadth-first description of each point cloud. After each file is downloaded, it draws the octree to the scene, with each node in the tree represented by a GL POINT, whose size is that of the width of the octree node. Point widths are kept constant with respect to the distance to the camera by multiplying the desired size by the height of the near projection plane, in homogenous coordinates. As more data is downloaded, the point cloud is updated. However, this is not simply a case of drawing higher-resolution data over the (previously drawn) lower resolution data, as the lower resolution data will occlude any higher resolution points. To deal with this issue, the "level" of the octree is tracked, and lower-resolution data are periodically removed from the draw-buffer.

Display times for a typical 2.5 million point dataset, over a clamped 8 megabyte/second internet connection, are 1 second for the initial low-resolution view, 10 seconds for 50% completion, and under 20 seconds for the full dataset. Such results are difficult to compare directly with other approaches as there has been little or no other research published on progressive point cloud visualization via the web. Yet the combined download and processing times for our approach improve on the results of the similar, yet different, challenge of progressive 3-D mesh visualization via the web [129].

Fig. 17 shows a visual representation of the progressive rendering effect, with four different pointclouds being downloaded simultaneously. While LIDAR data usually provides the highest resolution representation of the set, multiple 2-D data sources such as spheron data, static image data, and RGBD data are also used to create 3-D point clouds, as discussed in Section III, and are visualized also.

C. Hybrid 3-D, Video, and Image Rendering

As mentioned above, one of the advantages of using a browser context for visualizing multimodal data is that the HTML5 standard allows relatively straightforward combination of 2-D and 3-D data sources, permitting us to view image or streaming video data on surfaces within the 3-D scene.

The results of the work presented in Section III include data obtained via feature matching and registration algorithms, which are used in order to back-calculate the real position in the scene of the sensor used to record the data. These positions are then registered to the LIDAR data

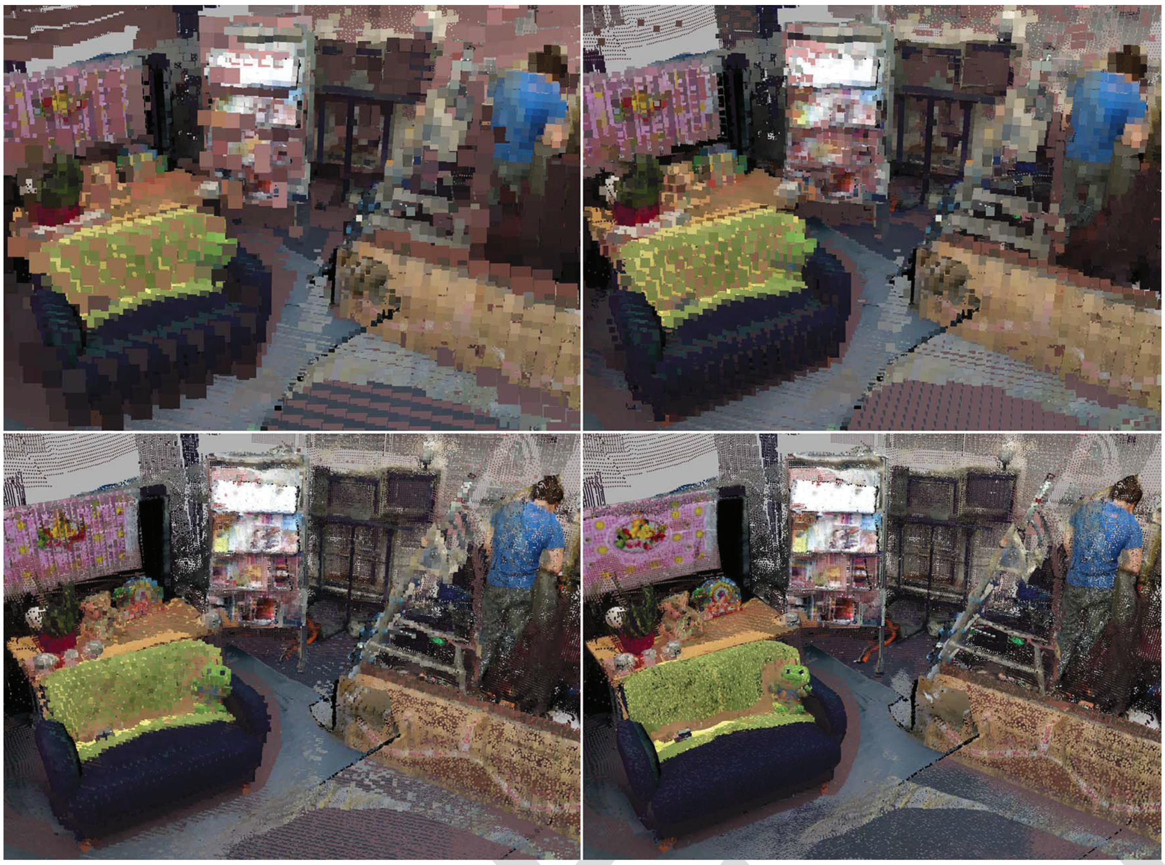


Fig. 17. ***Four screenshots of progressive rendering of the same scene (increasing time/resolution: left-to-right, top-to-bottom), constructed from LIDAR, spheron, static images and RGBD camera data.***

which is taken as the ground truth reference for the scene. The result is a set of matrices representing the position (and, for video camera footage, the orientation) of each sensor in the scene. On a static image level, this permit us to place a single textured plane (showing the source image for the sensor) located in the 3-D scene precisely where the sensor was when it took its image (see Fig. 10).

For video data, a hidden HTML5 video element is created for each video in the scene, and added to the browser document object model (DOM), the standard system for managing the HTML elements of the page. The fact that the element is hidden ensures that it does not affect the 3-D visualization. However, the WebGL API can read the HTMLVideoElement in the DOM, and extract the pixels of the current frame into texture data, which can be used within the 3-D scene. Similar to the static image case presented above, a simple plane mesh is created, using the position and orientation of each witness video camera in the scene as the model matrix. Then, every draw frame, the DOM video elements pipe their texture information as WebGL textures, which are displayed on the plane meshes. The result is the video data being rendered in real-time on meshes within the 3-D scene with the actual position and orientation of the camera, providing a tight 2-D/3-D integration. To control the playback of the video and to ensure that at most one video is playing at a time (avoiding needless bandwidth and rendering power use), the application features a timeline interface created as a 2-D Canvas and added to the DOM separately. This element allows the user to select a camera, which then moves the 3-D camera to a position immediately behind the plane mesh displaying the selected camera. Play/pause/stop controls exist for video playback, and scrubbing allows the user to skip forward and backwards (by setting the time of the HTML5 video element via javascript). Fig. 10 shows a screenshot of the timeline interface and the mesh planes featuring the video frames, illustrating the enhanced naturalness of the visualisation.

D. Metadata Visualization: Actions, Saliency, Coverage

As described in Section V, actors and human actions in multiview footage can be recognized through the use of visual data analysis techniques especially accelerated to deal with big media data on set. Furthermore, the camera which is most salient with respect to an action can be recognised. Saliency could be used, for instance, to decide which take should be included when editing in postproduction; metadata are useful to solve more intelligent queries. The metadata resulting from the processing is provided as an XML file and associated description with:

1) identification of one or more actors in a scene;
2) identification of actions carried out by the actors;
3) separation of the footage into timed segments;
4) saliency of the camera;
5) any additional metadata which has been added manually.

This information can then be overlayed on the timeline discussed above. Other metadata related to quality is the estimation of camera coverage as described in Section III. The result of this analysis is a collection of points in 3-D space with associated values representing the number of cameras that can accurately see that point. Fig. 18 shows how this sparse point cloud can be overlaid onto the set context, allowing users rapid feedback with regards to the coverage of a given configuration of cameras.

VII. DISCUSSION AND PERSPECTIVES

The generation of ever more digital data, which come from different sources and in different formats, is increasing. The case dealt with in this paper is the high-end movie industry, where currently 6 TB per on-set shooting day are average, and the variety of sensors used, data formats that result, and different reference systems, have been indicated. Poor understanding of the data in this industry (and others) leads to generate even more data ("in doubt, reshoot") with the associated costs of production and processing.

A first issue addressed was to make the multisource large data more intelligible through its integration into a unified 3-D space; from spherical, multiple stills, etc., different 3-D datasets are reconstructed, registered, and the sensors positions provided with respect to a ground truth reference (LIDAR in our case). Several aspects are improved with respect the current state-of-the-art in 3-D reconstruction.

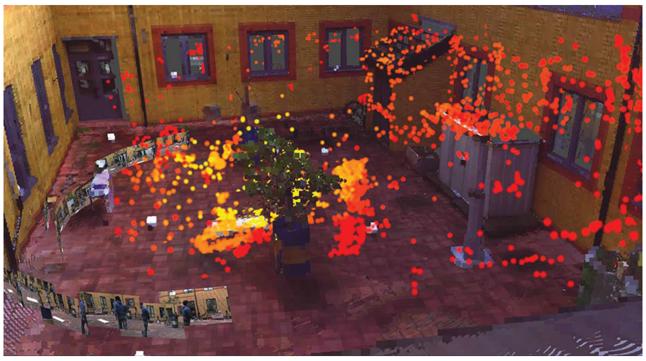


Fig. 18. ***Camera coverage point cloud superimposed on the LIDAR set data. Bright yellow points are seen by more cameras, dark red points are seen by fewer.***

Secondly, toolsets based on the previous approach to monitor quality of the data and of the set-ups have been provided and their success evaluated, testing them with part of the IMPART dataset. These toolsets are intended to support on-set or near-set decisions, to recalibrate, or change set-up to improve coverage, or reshoot in case of out-of-focus or frame drops. Some remedial action could also be taken in postproduction, and the methods introduced support better postproduction strategies as well. The real-time or near real-time requirement has been mainly met through reformulation of the approaches in terms of techniques of widespread use in big data problems, where important gains in some optimization and acceleration techniques have been achieved.

To allow more efficient management of the data, and better use further along the digital cinema production chain, for example, in postproduction, semantic analysis of multiview video led to the generation of metadata from an anthropocentric perspective. Novel concrete integrated visualization of different datasets, which takes advantage of the integration of multiple data sources and is based upon enhanced streaming techniques, can support further user driven quality assessment.

These contributions are rather digital cinema application specific. Most of the research advances presented in this paper have been already tested on film related material, some of it from actual productions, some of it from the IMPART research dataset mentioned previously. As a measure of their actual applicability, we should mention that most of the tools have been integrated in the software used by a major film industry player (Double Negative Visual Effects) to organize and process the vast amount of data captured. The automatic strategies presented in this paper do not always work for all on-set cases, but in the large number of cases where they do work, there is a substantial amount of savings in human noncreative labour by creative professionals.

On the other hand, the presented novel approaches which advance the state of the art (e.g., of integrating multisource data into a unified space—in this case, a 3-D one with common reference coordinates; of semantic temporal segmentation of videos, and of tight integrated visualization), could be applied in other big data areas, besides film production and postproduction. Moreover, the acceleration strategies (including novel numerical implementations for very basic processes), the distributed trimmed clustering technique, the proposed approximate Least Squares One-Class SVM classification approach, and the 3-D visualization (web or desktop), are novel approaches which are applicable to quite different types of big data problems, e.g., any types of data with a vectorial representation in the case of clustering and classification approaches. It is worth mentioning that the visualization of quality/errors through marginal covariances was not previously attempted because of the cost of the existing algorithms was prohibitive. The extensive testing of acceleration strategies in CPU and GPU for the very large scale problems (with which we were dealing) should be useful as well to orient further research and development.

Research perspectives do exist in the different areas considered, such as more challenging 3-D reconstruction and quality monitoring problems, further acceleration of processes, derivation of approximate, distributed or incremental versions of other clustering and classification algorithms, and augmented reality as a further tighter integrated visualization strategy.

In general, the area of big media (or multimedia) data processing and analysis is a very demanding but also promising new field, that already attracts the interest and research efforts of scientists and engineers in the area [132], as verified by the establishment of at least one conference series dealing with this topic (IEEE International Conference on Multimedia Big Data). Challenges and topics in the multimedia big data field include, among others [133], ultra-high efficiency compression, coding and transmission, content analysis, mining, interaction, visualization and semantic retrieval, deep learning and cloud computing for multimedia big data, high-efficiency storage, multimedia big data systems etc. ■

REFERENCES

[1] S. Nayar, “Catadioptric Omnidirectional Camera,” in *Proc. CVPR*, 1997, pp. 482–488.

[2] J. Starck, A. Maki, S. Nobuhara, A. Hilton, and T. Matsuyama, “The Multiple-Camera 3-D Production Studio,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 6, pp. 856–869, Jun. 2009.

[3] H. Kim, J.-Y. Guillemaut, T. Takai, M. Sarim, and A. Hilton, “Outdoor Dynamic 3-D Scene Reconstruction,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 11, pp. 1611–1622, Nov. 2012.

[4] V. Chvátal, “A combinatorial theorem in plane geometry,” *J. Combinatorial Theory, Series B*, vol. 18, no. 1, pp. 39–41, Feb. 1975.

[5] D. G. Costa and L. A. Guedes, “The Coverage Problem in Video-Based Wireless Sensor Networks: A Survey,” *Sensors*, vol. 10, no. 9, pp. 8215–8247, Sep. 2010.

[6] A. Mavrinac and X. Chen, “Modeling Coverage in Camera Networks: A Survey,” *Int. J. Comput. Vis.*, vol. 101, no. 1, pp. 205–226, Jan. 2013.

[7] E. Imre and A. Hilton, “Coverage evaluation of camera networks for facilitating big-data management in film production,” in *Proc. ICIP*, 2015.

AQ2 [8] J. Zhao, S. C. Cheung, and T. Nguyen, “Optimal visual sensor network configuration,” in *Multi-camera networks: Principles and Applications*. 2009, pp. 139–162.

[9] K. G. Cowan and P. D. Kovesi, “Automatic sensor placement for vision task requirements,” *IEEE Trans. Pattern Recognit. Mach. Intell.*, vol. 10, no. 3, pp. 407–416, May 1988.

[10] E. Imre, J.-Y. Guillemaut, and A. Hilton, “Moving camera registration for multiple camera setups in dynamic scenes,” in *Proc. BMVC*, 2010, pp. 38.1–38.12.

[11] J. Mitchelson and A. Hilton, “Wand-based Multiple Camera Studio Calibration,” Univ. Surrey, CVSSP, Surrey, U.K., Tech. Rep. VSSP-TR-2/2003, 2003.

[12] E. Imre, J.-Y. Guillemaut, and A. Hilton, “Moving camera registration for multiple camera setups in dynamic scenes,” in *Proc. 3DIMPVT*, 2011, pp. 260–267.

[13] C. Wu, VisualSFM: A Visual Structure from Motion System, last visited on 30 July 2015. [Online]. Available: http://ccwu.me/vsfm

[14] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, “Multicore bundle adjustment,” in *Proc. CVPR*, 2011, pp. 3057–3064.

[15] N. Hasler, B. Rosenhahn, T. Thormahlen, M. Wand, J. Gall, and H.-P. Seidel, “Markerless motion capture with unsynchronized moving cameras,” in *Proc. CVPR*, 2009, pp. 224–231.

[16] E. Imre, J.-Y. Guillemaut, and A. Hilton, “Through-the-Lens multi-camera synchronisation and frame drop detection for 3D reconstruction,” in *Proc. 3DIMPVT 2012*, 2012, pp. 395–402.

[17] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2004, 0521540518.

[18] L. Wolf and A. Zomet, “Wide baseline matching between unsynchronized video sequences,” *Int. J. Comput. Vis.*, vol. 68, no. 1, pp. 43–52, Mar. 2006.

[19] P. A. Tresadern and I. D. Reid, “Video synchronisation from human motion using rank constraints,” *Comput. Vis. Image Understanding*, , pp. 891–906, Aug. 2009.

[20] J. Yan and M. Pollefeys, “Video synchronisation via space-time interest point distribution,” in *Proc. ACVIS 2004*, 2004.

[21] F. Diego, D. Ponsa, and J. Serrat, “Video alignment for change detection,” *IEEE Trans. Image Process.*, vol. 20, no. 7, pp. 1858–1869, Jul. 2011.

[22] F. L. C. Padua, R. L. Carceroni, A. M. R. G. Santos, and K. N. Kutulakos, “Linear sequence-to-sequence alignment,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 2, pp. 304–320, Feb. 2010.

[23] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Trans. Inf. Theory*, vol. IT-13, no. 2, pp. 260–269, Feb. 1967.

[24] I. Stamos, L. Liu, C. Chen, G. Wolberg, G. Yu, and S. Zokai, “Integrating automated range registration with multiview geometry for the photorealistic modeling of large-scale scenes,” *Int. J. Comput. Vision*, vol. 78, no. 2–3, pp. 237–260, 2008.

[25] T. Sattler, B. Leibe, and L. Kobbelt, “Improving image-based localization by active correspondence search,” presented at the ECCV, 2012.

[26] M. Restrepo and J. Mundy, “An evaluation of local shape descriptors in probabilistic volumetric scenes,” in *Proc. BMVC*, 2012, pp. 46.1–46.11.

[27] A. Mastin, J. Kepner, and J. Fisher, “Automatic registration of LIDAR and optical images of urban scenes,” in *Proc. CVPR*, 2009, pp. 2639–2646.

[28] S. Budge, N. Badamikar, and X. Xie, “Automatic registration of fused lidar/digital imagery (texel images) for three-dimensional image creation,” *Opt. Eng.*, vol. 54, no. 3, p. 031105, 2015.

[29] R. Wang, F. Ferrie, and J. Macfarlane, “Automatic registration of mobile LiDAR and spherical panoramas,” in *Proc. CVPR*, 2012, pp. 33–40.

[30] U. Hahne and M. Alexa, “Combining time of flight depth and stereo images without accurate extrinsic calibration,” *Int. J. Intell. Syst. Technol. Appl.*, vol. 5, no. 3/4, pp. 325–333, 2008.

[31] I. Stamos and P. K. Allen, “Integration of range and image sensing for photorealistic 3-D modeling,” in *Proc. ICRA*, 2000, pp. 1435–1440.

[32] H. Kim and A. Hilton, “Evaluation of 3D feature descriptors for multi-modal data registration,” in *Proc. 3DV*, 2013, pp. 119–126.

[33] H. Kim and A. Hilton, “Influence of colour and feature geometry on multi-modal 3D point clouds data registration,” in *Proc. 3DV*, 2014, pp. 4321–4328.

[34] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, “KinectFusion: Real-time dense surface mapping and tracking,” presented at the IEEE ISMAR, 2011.

[35] N. Snavely, S. Seitz, and R. Szeliski, “Photo tourism: Exploring photo collections in 3D,” in *Proc. ACM SIGGRAPH*, 2006, pp. 835–846.

[36] Y. Furukawa and J. Ponce, “Accurate, dense, and robust multiview stereopsis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 8, pp. 1362–1376, Aug. 2010.

[37] H. Kim and A. Hilton, “3D scene reconstruction from multiple spherical stereo pairs,” *Int. J. Comput. Vis.*, vol. 104, no. 1, pp. 94–116, 2013.

[38] H. Dutagaci, C. P. Cheung, and A. Godil, “Evaluation of 3D interest point detection techniques via human-generated ground truth,” *Vis. Comput.*, vol. 28, no. 9, pp. 901–917, 2012.

[39] F. Tombari, S. Salti, and L. Di Stefano, “Performance evaluation of 3D keypoint detectors,” *Inte. J. Comput. Vis.*, vol. 102, pp. 198–220, 2013.

[40] C. Tomasi and T. Kanade, “Detection and tracking of point features,” *Pattern Recognit.*, vol. 37, pp. 165–168, 2004.

[41] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[42] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, “A comprehensive performance evaluation of 3D local feature descriptors,” *Inte. J. Comput. Vis.*, vol. On-line First version, 2015.

[43] R. B. Rusu, N. Blodow, and M. Beetz, “Fast Point Feature Histograms (FPFH) for 3D registration,” in *Proc. ICRA*, 2009, pp. 3212–3217.

[44] P. Besl and N. McKay, “A method for Registration of 3-D Shapes,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[45] A. Laurentini, “The visual hull concept for silhouette-based image understanding,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 2, pp. 150–162, Feb. 1994.

[46] G. K. Cheung, S. Baker, and T. Kanade, “Visual hull alignment and refinement across time: A 3d reconstruction algorithm combining shape-from-silhouette with stereo,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2003, vol. 2, p. II-375.

[47] M. I. Lourakis and A. A. Argyros, “Sba: A software package for generic sparse bundle adjustment,” *ACM Trans. Math. Software (TOMS)*, vol. 36, no. 1, p. 2, 2009.

[48] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, “Towards internet-scale multi-view stereo,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2010, pp. 1434–1441.

[49] L. Polok, V. Ila, and P. Smrz, “Cache efficient implementation for block matrix operations,” in *Proc. 21st High Perform. Comput. Symp.*, 2013, pp. 698–706.

[50] S. V. Ila, L. Polok, M. Šolony, P. Zemčík, and P. Smrž, “Fast covariance recovery in incremental nonlinear least square solvers,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 1–8. [Online]. Available: http://www.fit.vutbr.cz/research/view_pub.php?id=10823IEEE Computer Society

[51] F. Dellaert and M. Kaess, “Square root SAM: Simultaneous localization and mapping via square root information smoothing,” *Intl. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1203, Dec. 2006.

[52] T. A. Davis, *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*. Philadelphia, PA, USA: SIAM, 2006.

[53] L. Polok, M. Solony, V. Ila, P. Zemcik, and P. Smrz, “Efficient implementation for block matrix operations for nonlinear least squares problems in robotic applications,” presented at the IEEE Int. Conf. Robot. Autom., 2013.

[54] L. Polok, V. Ila, M. Solony, P. Smrz, and P. Zemcik, “Incremental block cholesky factorization for nonlinear least squares in robotics,” presented at the Robot.: Sci. Syst., 2013.

[55] L. Polok, V. Ila, and P. Smrz, “Fast sparse matrix multiplication on gpu,” in *Proc. 23st High Perform. Comput. Symp.*, 2015, pp. 1–8Association for Computing Machinery.

AQ3 [56] M. Kaess and F. Dellaert, “Covariance recovery from a square root information matrix for data association,” *Robot. Autonom. Syst.*, 2009.

[57] A. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia, PA, USA: SIAM, 1996.

[58] M. Šolony, E. Imre, V. Ila, L. Polok, H. Kim, and P. Zemčík, “Fast and accurate refinement method for 3d reconstruction from stereo spherical images,” in *Proc. 10th Int. Conf. Comput. Vis. Theory Appl.*, 2015, pp. 1–8. [Online]. Available: http://www.fit.vutbr.cz/research/view_pub.php?id=10866

[59] T. Chen, A.-D. Lu, and S.-M. Hu, “Visual storylines: Semantic visualization on movie sequence,” *Comput. Graphics*, vol. 36, no. 4, pp. 241–249, 2012.

[60] B. Schölkopf, A. Smola, and K.-R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998.

[61] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce, “Automatic annotation of human actions in video,” presented at the Int. Conf. Comput. Vis. (ICCV), , 2009.

[62] I. Laptev, M. Marszałek, C. Schmid, B. Rozenfeld, I. Rennes, I. I. Grenoble, and L. Ljk, “Learning realistic human actions from movies,” presented at the Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), 2008.

[63] A. Iosifidis, A. Tefas, and I. Pitas, “Minimum class variance extreme learning machine for human action recognition,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 11, pp. 1968–1979, Nov. 2013.

[64] D. Weinland, R. Ronfard, and E. Boyer, “A survey of vision-based methods for action representation, segmentation and recognition,” *Comput. Vis. Image Understand.*, vol. 115, no. 2, pp. 224–241, 2011.

[65] L. Santos, K. Khoshhal, and J. Dias, “Trajectory-based human action segmentation,” *Pattern Recognit.*, pp. 568–579, 2014.

[66] Q. Shi, L. Wang, L. Cheng, and A. Smola, “Discriminative human action segmentation and recognition using semi-markov model,” in *Proc. Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2008, pp. 1–8.

[67] K. Kahol, P. Tripathi, S. Panchanathan, and T. Rikakis, “Gesture segmentation in complex motion sequences,” in *Proc. Int. Conf. Image Process.*, 2003, vol. 2, pp. 105–108.

[68] K. Oka, Y. Sato, and H. Koike, “Real-time fingertip tracking and gesture recognition,” *IEEE Comput. Graphics Appl.*, vol. 22, pp. 64–71, 2002.

[69] F. Bashir, A. Khokhar, and D. Schonfeld, “View-invariant motion trajectory-based activity classification and recognition,” *Multimedia Syst.*, vol. 12, pp. 45–54, 2006.

[70] X. Xuan and K. Murphy, “Modeling changing dependency structure in multivariate time series,” in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 1055–1062.

[71] Z. Harchaoui, F. Bach, and E. Moulines, “Kernel change-point analysis,” *Proc. Neural Inf. Proc. Syst.*, 2008.

[72] V. Pavlovic, J. M. Rehg, and J. Maccormick, “Learning switching linear models of human motion,” *Adv. Neural Inf. Process. Syst.*, pp. 981–987, 2000.

[73] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, “Dense trajectories and motion boundary descriptors for action recognition,” *Int. J. Comput. Vis.*, vol. 103, no. 1, pp. 60–79, 2013.

[74] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[75] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid, “Local features and kernels for classification of texture and object categories: A comprehensive study,” *Int. J. Comput. Vis.*, vol. 73, no. 2, pp. 213–238, 2007.

[76] A. Iosifidis, A. Tefas, N. Nikolaidis, and I. Pitas, “Multi-view human movement recognition based on fuzzy distances and linear discriminant analysis,” *Comput. Vis. Image Understand.*, vol. 116, no. 3, pp. 347–360, 2012.

[77] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A review,” *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sept. 1999.

[78] J. B. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.

[79] V. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.

[80] S. Yu, L.-C. Tranchevent, X. Liu, W. Glanzel, J. A. Suykens, B. D. Moor, and Y. Moreau, “Optimized data fusion for kernel k-means clustering,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 5, pp. 1031–1039, May 2012.

[81] F. Zhou, F. De la Torre Frade, and J. K. Hodgins, “Hierarchical aligned cluster analysis for temporal clustering of human motion,” *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)*, vol. 35, no. 3, pp. 582–596, Mar. 2013.

[82] H. Jia, Y. Ming Cheung, and J. Liu, “Cooperative and penalized competitive learning with application to kernel-based clustering,” *Pattern Recognit.*, 2014.

[83] M. R. Ferreira and F. de A. T. de Carvalho, “Kernel-based hard clustering methods in the feature space with automatic variable weighting,” *Pattern Recognit.*, 2014.

[84] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, “A survey of kernel and spectral methods for clustering,” *Pattern Recognit.*, vol. 41, no. 1, pp. 176–190, 2008.

[85] D. Agrawal, S. Das, and A. El Abbadi, “Big data and cloud computing: Current state and future opportunities,” in *Proc. 14th Int. Conf. Extending Database Technol.*, ser. *EDBT/ICDT ’11*, 2011, pp. 530–533.

[86] L. M. Rodrigues, L. E. Zárate, C. N. Nobre, and H. C. Freitas, “Parallel and distributed kmeans to identify the translation initiation site of proteins,” in *IEEE SMC*, 2012, pp. 1639–1645.

[87] A. Ene, S. Im, and B. Moseley, “Fast clustering using mapreduce,” in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, ser. *KDD ’11*, 2011, pp. 681–689.

[88] N. Tsapanos, A. Tefas, N. Nikolaidis, and I. Pitas, “A distributed framework for trimmed kernel k-means clustering,” *Pattern Recognit.*, vol. 48, pp. 2685–2698, 2015.

[89] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” *ACM Commun.*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[90] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: Cluster computing with working sets,” in *Proc. 2Nd USENIX Conf. Hot Topics Cloud Comput.*, ser. *HotCloud’10*, 2010, pp. 10–10, Berkeley, CA, USA: USENIX Association.

[91] T. White, *Hadoop: The Definitive Guide* 1st. Sebastopol, CA, USA: O’Reilly Media, Inc., 2009.

[92] H. Karloff, S. Suri, and S. Vassilvitskii, “A model of computation for mapreduce,” in *Proc. 21st Annu. ACM-SIAM Symp. Discrete Algorithms*, ser. *SODA’10*, 2010, pp. 938–948.

[93] T. O. Kvålseth, “Entropy and correlation: Some comments,” *IEEE Trans. Syst., Man, Cybern.*, vol. 17, no. 3, pp. 517–519, May 1987.

[94] R. Chitta, R. Jin, T. C. Havens, and A. K. Jain, “Approximate kernel k-means: Solution to large scale kernel clustering,” in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 895–903.

[95] L. Wolf, T. Hassner, and I. Maoz, “Face recognition in unconstrained videos with matched background similarity,” presented at the Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2011.

[96] T. Ojala, M. Pietikainen, and D. Harwood, “Performance evaluation of texture measures

with classification based on kullback discrimination of distributions," in *Proc. 12th IAPR Int. Conf. Pattern Recognit. Comput. Vis. Image Process.*, 1994, no. 1, pp. 582–585.

[97] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Proc. Artif. Neural Netw.—ICANN'97*, 1997, pp. 583–588.

[98] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *Proc. 15th Int. Conf. Mach. Learn. (ICML-1998)*, 1998, pp. 515–521.

[99] T. Evgeniou, M. Pontil, and T. Poggio, "Regularization networks and support vector machines," *Adv. Comput. Math.*, vol. 13, no. 1, pp. 1–50, 2000.

[100] I. Steinwart, "Consistency of support vector machines and other regularized kernel classifiers," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 128–142, 2005.

AQ4 [101] A. Iosifidis, A. Tefas, and I. Pitas, "On the kernel Extreme Learning Machine classifier," *Pattern Recognit. Lett.*, 2015.

[102] B. Krawczyk and M. Woźniak, "Handling label noise in microarray classification with one-class classifier ensemble," in *Proc. ICT Innovations 2014*, vol. 311, *ser. Advances in Intelligent Systems and Computing*, A. M. Bogdanova and D. Gjorgjevikj, Eds., 2015, pp. 351–359.

[103] V. Mygdalis, A. Iosifidis, A. Tefas, and I. Pitas, "Video summarization based on subclass support vector data description," presented at the IEEE Symp. Series Comput. Intell., 2014.

[104] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.

[105] D. M. Tax and R. P. Duin, "Support vector data description," *Mach. Learn.*, vol. 54, no. 1, pp. 45–66, 2004.

[106] Y.-S. Choi, "Least squares one-class support vector machine," *Pattern Recognit. Lett.*, vol. 30, no. 13, pp. 1236–1240, 2009.

[107] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Process.*, vol. 99, pp. 215–249, 2014.

[108] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire, "An experimental evaluation of novelty detection methods," *Neurocomputing*, vol. 135, pp. 313–327, 2014.

[109] D. Achlioptas and F. McSherry, "Fast computation of low rank matrix approximations," in *Proc. 33rd Annu. ACM Symp. Theory Comput.*, 2001, pp. 611–618.

[110] P. Drineas and M. W. Mahoney, "On the nyström method for approximating a gram matrix for improved kernel-based learning," *J. Mach. Lear. Res.*, vol. 6, pp. 2153–2175, 2005.

[111] K. Zhang, I. W. Tsang, and J. T. Kwok, "Improved nyström low-rank approximation and error analysis," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1232–1239.

[112] M. Nandan, P. P. Khargonekar, and S. S. Talathi, "Fast svm training using approximate extreme points," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 59–98, 2014.

[113] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," *VISAPP (1)*, vol. 2, 2009.

[114] M. Muja and D. G. Lowe, "Flann, fast library for approximate nearest neighbors," presented at the Int. Conf. Comput. Vis. Theory Appl. (VISAPP'09), 2009.

[115] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Computational Learning Theory*. Berlin, Germany: Springer-Verlag, 2001, pp. 416–426.

[116] C. Cortes, M. Mohri, and A. Talwalkar, "On the impact of kernel approximation on learning accuracy," in *Proc. Int. Co. Artif. Intell. Statist.*, 2010, pp. 113–120.

[117] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Mach. Learn.*, vol. 30, no. 2–3, pp. 195–215, 1998.

[118] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2011, pp. 529–534, IEEE.

[119] J. Jankowski and S. Decker, "On the design of a dual-mode user interface for accessing 3d content on the world wide web," *Int. J. Human-Comput. Studies*, vol. 71, no. 7, pp. 838–857, 2013.

[120] J. Jankowski and S. Decker, "A dual-mode user interface for accessing 3d content on the world wide web," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 1047–1056ACM.

[121] J. Behr, P. Eschler, Y. Jung, and M. Zöllner, "X3dom: A dom-based html5/x3d integration model," in *Proc. 14th Int. Conf. 3D Web Technol.*, 2009, pp. 127–135.

[122] K. Sons, F. Klein, D. Rubinstein, S. Byelozyorov, and P. Slusallek, "Xml3d: interactive 3d graphics for the web," in *Proc. 15th Int. Conf. Web 3D Technol.*, 2010, pp. 175–184.

[123] J. Agenjo, A. Evans, and J. Blat, "Webglstudio: A pipeline for webgl scene creation," in *Proc. 18th Int. Conf. 3D Web Technol.*, *ser. Web3D '13*, 2013, pp. 79–82. [Online]. Available: http://doi.acm.org/10.1145/2466533.2466551

[124] A. Evans, M. Romeo, A. Bahrehmand, J. Agenjo, and J. Blat, "3D graphics on the web: A survey," *Comput. Graphics*, vol. 41, pp. 43–61, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0097849314000260

[125] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3d mesh compression: A survey," *J. Visu. Commun. Image Representation*, vol. 16, no. 6, pp. 688–733, 2005.

[126] M. Limper, S. Wagner, C. Stein, Y. Jung, and A. Stork, "Fast delivery of 3d web content: a case study," in *Proc. 18th Int. Conf. 3D Web Technol.*, 2013, pp. 11–17.

[127] W. Chun, "Webgl models: End-to-end," in *OpenGL Insights*, P. Cozzi and C. Riccio, Eds. Boca Raton, FL, USA: CRC Press, 2012.

[128] J. Behr, Y. Jung, T. Franke, and T. Sturm, "Using images and explicit binary container for efficient and incremental delivery of declarative 3d scenes on the web," in *Proc. 17th Int. Conf. 3D Web Technol.*, 2012, pp. 17–25.

[129] G. Lavoué, L. Chevalier, and F. Dupont, "Streaming compressed 3d data on the web using javascript and webgl," in *Proc. 18th Int. Conf. 3D Web Technol.*, 2013, pp. 19–27.

[130] A. Evans, J. Agenjo, and J. Blat, "Web-based visualisation of on-set point cloud data," in *Proce. 11th Eur. Conf. Visual Media Product.*, 2014, p. 10.

[131] A. Evans, J. Agenjo, and J. Blat, "Hybrid visualisation of digital production big data," in *Proc. 20th Int. Conf. 3D Web Technol.*, 2015, pp. 69–72.

[132] S.-C. Chen, R. Jain, Y. Tian, and H. Wang, "Guest editorial," *IEEE Trans. Multimedia, Special Issue on Multimedia: The Biggest Big Data*, vol. 17, no. 9, pp. 1401–1403, Sep. 2015.

[133] IEEE International Con. Multimedia Big Data. [Online]. Available: http://bigmm2016.asia.edu.tw/call-for-papers/

ABOUT THE AUTHORS

Josep Blat received the Ph.D. degree in mathematics from Heriot-Watt University, Edinburgh, U.K., in 1985.

He is currently a Full Professor in the Department of Information and Communication Technologies at Universitat Pompeu Fabra, Barcelona, Spain. After his initial research in Applied Nonlinear Analysis, he moved to modeling and mathematical analysis of images (collective prize Philip Morris France 1991). His current research interests include advanced 3-D graphics (human modeling and animation, games), human-computer interaction (older people, ethnography, geolocation), and technology-enhanced learning (learning standards and tools, collaborative learning, new learning environments).

Alun Evans received the Ph.D. degree in medical physics from University College, London, U.K., in 2006.

He is currently an Associate Lecturer at the Universitat Pompeu Fabra, Barcelona, Spain. After completing his postdoctoral work in computer graphics and character animation, he moved into the visual media production industry where he was a Technical Lead on a successful video game, and CTO and Head of Research for a tech-startup company. He rejoined academia in 2013, where he currently researches and teaches in the fields of 3-D graphics and web-application development.

Hansung Kim received the MS and Ph.D degrees in electronic and electrical engineering from Yonsei University, Seoul, Korea, in 2001 and 2005, respectively.

He was employed as a Researcher of Knowledge Science Lab (KSL) at Advanced Telecommunications Research Institute International (ATR), Japan, from 2005 to 2008. He is currently a Research Fellow (RA2) at the Centre for Vision, Speech, and Signal Processing (CVSSP) at the University of Surrey, Surrey, U.K. His research interests include 3-D computer vision, image-based modeling, multimodal data processing, and media production.

Evren Imre received the Ph.D. degree from Middle East Technical University, Ankara, Turkey, in 2007.

He has been with the University of Surrey, Surrey, U.K., as a Research Fellow since 2009. His research interests include multiview geometry and its applications, calibration and synchronization of multicamera set-ups, RANSAC, and covariance estimation.

Lukàš Polok was born in Brno, Czech Republic. He received the M.Sc. degree in computer science, with a specialization in computer graphics and multimedia, from Brno University of Technology, Brno, Czech Republic.

He is currently a Researcher at the Brno University of Technology working on his thesis about using GPUs for general purpose calculations.

Viorela Ila received the Engineering degree in industrial engineering and automation from the Technical University of Cluj-Napoca, Cluj-Napoca, Romania, in 2000, and the Ph.D. degree in information technologies from the Universitat de Girona, Girona, Spain, in 2005.

During her research career, she worked in several worldwide-recognized robotic and computer vision research centers [VICOROB-Girona, IRI-Barcelona, Georgia Tech-Atlanta, LAAS-Toulouse and Brno University of Technology (BUT)], participated in four EU FP7 projects, five Spanish national, one French RTRA-STAE Project, and one Czech Republic national project and published more than 35 high quality papers (two book chapter, five journals, 26 conferences, and two technical reports) having about 550 citations. Currently, she is with the ARC Centre of Excellence for Robotic Vision at the Australian National University.

AQ5 **Nikos Nikolaidis** (Senior Member, IEEE)

He is currently an Assistant Professor at the Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece. He has coauthored 1 book, 15 book chapters, 48 journal papers, and 152 conference papers, and coedited one book and two special issues in journals. The number of citations to his work by third authors exceeds 3000 (h-index 24). He has participated in 24 research projects funded by the EU and national funds. His areas of interest/expertise include anthropocentric video analysis (human detection and tracking, activity recognition), computer vision, digital image/video processing, stereoscopic/multiview video processing/analysis, computer graphics and visualization.

Dr. Nikolaidis is currently serving as an Associate Editor for the *EURASIP Journal on Image and Video Processing* and four more international journals. He served as Exhibits Chair of IEEE ICIP 2001, Technical Program Chair of IEEE IVMSP 2013 workshop and is currently serving as Publicity Cochair of EUSIPCO 2015. He will be publicity chair of IEEE ICIP 2018.

Pavel Zemčík was born in Brno, Czech Republic. He received the M.Sc and Ph.D. degrees in computer science, specializing in computer graphics, from the Faculty of Electrical Engineering and Computer Science, Brno University of Technology, Brno, Czech Republic, in 1989 and 1995 respectively.

He is currently employed as a Professor of computer graphics at Brno University of Technology. His research interests include algorithms of graphics and vision accelerated in FPGA and GPU, user interfaces, and applications of signal and video processing. He authored and coauthored over 15 journal papers, 60 conference papers, and 3 book chapters. His teaching activities include computer graphics and image processing.

Anastasios Tefas received the B.Sc. degree in informatics, in 1997, and the Ph.D. degree in informatics in 2002, both from the Aristotle University of Thessaloniki, Thessaloniki, Greece.

Since 2013, he has been an Assistant Professor at the Department of Informatics, Aristotle University of Thessaloniki. From 2008 to 2012, he was a Lecturer at the same University. From 2006 to 2008, he was an Assistant Professor at the Department of Information Management, Technological Institute of Kavala. From 2003 to 2004, he was a temporary lecturer in the Department of Informatics, University of Thessaloniki. From 1997 to 2002, he was a researcher and teaching assistant in the Department of Informatics, University of Thessaloniki. He has participated in 12 research projects financed by national and European funds. He has coauthored 59 journal papers, 145 papers in international conferences, and contributed 8 chapters to edited books in his area of expertise. Over 2700 citations have been recorded to his publications and his H-index is 27 according to Google scholar. His current research interests include computational intelligence, pattern recognition, statistical machine learning, digital signal and image processing and computer vision, biometrics, and security.

Pavel Smrž is an Associate Professor in the Department of Computer Graphics and Multimedia, Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic, where he leads the Knowledge Technology Research Group. His research interests include big data processing, hardware-accelerated machine learning, large-scale distributed and parallel processing, human-computer interaction, and information extraction. AQ6

Adrian Hilton (Member, IEEE) received the B.S.(Hons.) and D.Phil. degrees from University of Sussex, Sussex, U.K., in 1988 and 1992, respectively.

He is a currently a Professor of Computer Vision and Graphics and Director of the Centre for Vision, Speech, and Signal Processing at the University of Surrey, Surrey, U.K. His research interests include robust computer vision to model and understand real world scenes. Contributions include technologies for the first hand-held 3-D scanner, modeling of people from images and 3-D video for games, broadcast, and film production. He currently leads research investigating the use of computer vision for applications in entertainment content production, visual interaction, and clinical analysis.

Ioannis Pitas (Fellow, IEEE) received the Diploma and Ph.D. degrees in electrical engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece.

Since 1994, he has been a Professor at the Department of Informatics at Aristotle University of Thessalonik. He served as a Visiting Professor at several Universities. His current interests are in the areas of image/video processing, intelligent digital media, machine learning, human centered interfaces, affective computing, computer vision, 3-D imaging, and biomedical imaging. He has published over 800 papers, contributed in 44 books in his areas of interest and edited or (co)authored another 10 books. He has also been member of the program committee of many scientific conferences and workshops. He participated in 68 R&D projects, primarily funded by the European Union and is/was principal investigator/researcher in 40 such projects. He has 21000+ citations to his work and h-index 68+ (7/2015).

Dr. Pitas has served as Associate Editor or Coeditor of eight international journals, and as General or Technical Chair of four international conferences.