Towards GPU Accelerated HyperSpectral Depth Estimation in Medical Applications

Jaime Sancho*, Gemma Urbanos*, Luisa Ruiz*, Marta Villanueva*, Gonzalo Rosa*
Alberto Diaz*, Manuel Villa* Miguel Chavarrías*, Alfonso Lagares‡ Rubén Salvador†, Eduardo Juárez*, César Sanz*

*Universidad Politécnica de Madrid (UPM), Research Center on Software Technologies and Multimedia Systems
‡Instituto de Investigación Sanitaria Hospital 12 de Octubre (imas12), Madrid, España
†CentraleSupélec
{jaime.sancho, gemma.urbanos, luisa.rruiz, marta.villanueva.torres, miguel.chavarrias, eduardo.juarez, cesar.sanz}@upm.es
{gonzalo.rosa.olmeda, a.martinp, manuel.villa.romero}@alumnos.upm.es
ruben.salvador@centralesupelec.fr
alfonsolagares@salud.madrid.org

***Abstract*—HyperSpectral (HS) images are commonly used for classification tasks in different domains, such as medicine. In this field, a recent use is the differentiation between healthy tissues and different types of cancerous tissues. To this end, different machine learning techniques have been proposed to generate classification maps that indicate the type of tissue corresponding to each pixel in the image. These 2D representations can be used stand-alone, but they can not be properly registered with other valuable data sources like Magnetic Resonance Imaging (MRI), which can improve the accuracy of the system. For this reason, this paper builds the foundations of a multi-modal classification system that will incorporate 3D information into HS images. Specifically, we address the acceleration of one of the hotspots in depth estimation tools/algorithms.**

MPEG-I Depth Estimation Reference Software (DERS) provides high-quality depth maps relying on a global energy optimizer algorithm: Graph Cuts. However, this algorithm needs huge processing times, preventing its use during surgical operations. This work introduces GoRG (Graph cuts Reference depth estimation in GPU), a GPU accelerated DERS able to produce depth maps from RGB and HS images. In this paper, due to the lack of HS multi-view datasets at the moment, results are reported on RGB images to validate the acceleration strategy.

GoRG shows a $\times$25 average speed-up compared to baseline DERS 8.0, reducing total computation time from around one hour for 8 frames to only a few minutes. A consequence of our parallelization is an average decrease of 1.6 dB in Weighted-to-Spherically-Uniform Peak-Signal-to-Noise-Ratio (WS-PSNR), with some remarkable disparities approaching 4 dB. However, using Structural Similarity Index (SSIM) as metric results come closer to baseline DERS. Effectively, an average decrease of only 1.20% is achieved showing that the obtained speed-up gains compesate the subjective quality losses.

***Index Terms*—HSI, Depth Estimation**

This work was supported by the Spanish Government through PLATINO project (TEC2017-86722-C4-2-R) and the Regional Government of Madrid through NEMESIS-3D-CM project (Y2018/BIO-4826).

I. Introduction and motivation

HyperSpectral (HS) imaging has become a trendy research field when classification and detection is involved. Using more information from the electromagnetic spectrum allowed researchers to differentiate materials in a better way, based on the concept of spectral signature; a 2D representation of an acquired pixel where x axis refers to the wavelength and y axis refers to the level of reflectance acquired for that pixel and wavelength. This strength, joint to the benefits of an image structure, caused its expansion into numerous areas.

One relevant field is medicine, where HS images information can improve the detection and classification of tissues of different nature. In this regard, several previous works addressed the border detection of different kind of tumours, using supervised machine learning and image processing techniques [8], [11], [20]. These solutions aim at the generation of a 2D representation of the exposed tissue at a certain position and time, with a fixed camera position and at the moment the capture is acquired. However, other widely extended devices such as MRI (Magnetic Resonance Imaging), also employed in tumour identification, inherently produce 3D results. Specifically, for brain tumours [7], these two data sources are independent and even acquired at different moments: HS images are captured during the surgical operation whilst MRI before it.

This work represents the first step to create a link between these two technologies. The aim is to provide, in real-time, 3D depth information to classification maps generated from HS images during tumour resection operations Thanks to it, these classification maps could be properly fused with the information present in MRIs, improving cancer detection accuracy during surgical operations. In addition, the existence of a 3D model would help building better visualization tools for surgeons.

To do so, this work relies on MPEG Depth Estimation Reference Software (DERS), a tool developed by the MPEG-

I group and intended to provide accurate depth estimations used in immersive video applications. Using its structure as a basis, this paper presents a GPU accelerated application able to generate depth maps from HS and RGB images. Due to the lack of multi-view HS data-sets, only the RGB acceleration is addressed here, building the foundations for the HS depth estimation that will be tackled in future works.

In this paper we expose the general structure of DERS, depict its architecture in different stages, which we analyze, and introduce the acceleration strategies followed. Results show that our accelerated DERS, denominated Graph cuts Reference depth estimation in GPU (GoRG), achieves a $\times 25$ average speed-up for the sequences tested, with an average loss Weighted-to-Spherically-Uniform Peak-Signal-to-Noise-Ratio (WS-PSNR) of 1.6 dB and Structural SIMilarity Index (SSIM) loss of 1.20 %.

The paper structure is as follows: Section II contains a brief review of the state of the art and Section III features a stage by stage presentation of the algorithm and its acceleration. We present and analyze the results obtained in Section IV and draw some conclusions and future lines in Section V.

II. Related work

Recent depth estimation techniques employ monocular cameras along with Convolutional Neuronal Networks to obtain depth estimations in real-time using GPUs [6], [17], [21]. However, these works aim at autonomous driving applications, where the image aspect ratio is wide-horizontal and the level of detail required is low compared to immersive video applications, setting them apart from the case considered here.

A novel approach to compute depth maps consists in using plenoptic cameras, which enable obtaining accurate and fast depth estimations based on dense light-field acquisitions. Currently, researchers are addressing this problem from two points of view: (i) algorithm optimization [10], [23] and (ii) acceleration, mainly using GPUs [13].

Although specialized devices are employed, the use of traditional RGB cameras is common practice for depth estimation. Cheng et al. [4] proposed a GPU-accelerated method that relies on an adaptive window to improve the generation of the initial cost map. In another work, Senoh et al. [16] introduced an algorithm to refine the initial cost map based on non-iterative edge-adaptive local cost optimization. The objective is reducing the complexity and hence the computation time without using an accelerator.

III. Algorithm and acceleration

Figure 1 shows a simplified view of DERS 8.0 structure [15]. As it can be observed, the application is divided into 7 stages, included in 3 different color boxes. Green-background represents the initialization step, gray-background the auxiliary processes and blue-background the core algorithm, respectively. They are further explained in the following sections.

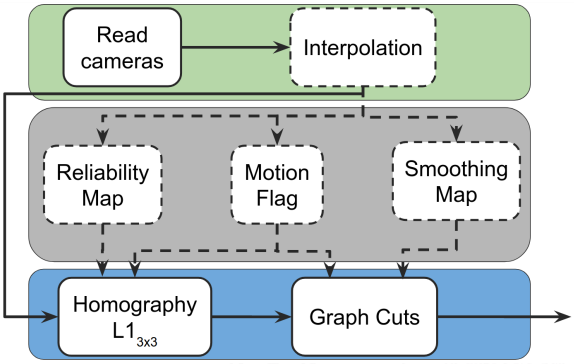


Fig. 1: DERS 8.0 structure: green background represents the initialization process; gray illustrate processes only using the main camera as input; and blue stages those that use all cameras as input. Solid lines represent mandatory whilst dashed lines optional processes, resp.

A. Read cameras and interpolation

The first step is the insertion of the camera information, which comprises RGB frames and intrinsic and extrinsic camera parameters [22]. Unlike DERS 8.0, GoRG allows an unbounded number of input cameras. The optional frame interpolation functionality in DERS 8.0 has not been implemented in this work.

B. Homography

This stage generates an initial cost volume for the main camera using all the cameras data. It is divided into two processes.

In the first process, for each main camera pixel, a number of corresponding pixels is calculated in other cameras. This is done by un-projecting rays from the main camera optical center passing through each pixel. Then, a limited space in z from a nearest Z_{near} and farthest plane Z_{far}, is divided into Z_n planes. The cross between the rays and these planes produce 3D points with a depth candidate associated each one. These 3D points are then projected to other cameras to obtain the corresponding projected pixels.

To implement this process, we construct the homography projection matrix in (1), which enables the calculation of a corresponding pixel $[u\ v]^T$ for a pair of cameras; given a pixel $[i\ j]^T$ and a depth candidate z. This way, the matrix is calculated in advance, reducing the per-pixel operation to only one matrix multiplication.

$$\frac{1}{s}\begin{bmatrix} u \\ v \\ s \\ 1 \end{bmatrix} = \begin{pmatrix} K_{cam}[R_{cam}|t_{cam}] \begin{pmatrix} K_{ref}[R_{ref}|t_{ref}] \\ \hline 0\ \ 0\ \ 0\ \ 1 \end{pmatrix}^{-1} \\ \hline 0 \quad 0 \quad 0 \quad 1 \end{pmatrix} \begin{pmatrix} i \\ j \\ 1 \\ 1/z \end{pmatrix} \quad (1)$$

where K refers to the intrinsic matrix, R to the rotation matrix, t to the translation vector and s to an undetermined constant.

In the second process, each pixel of the main camera is compared to its corresponding pixel, so that a cost per candidate is obtained. In this work, the cost is an addition of L1-norms of Y, U and V differences, using a weighted 3×3 window for Y.

Both corresponding pixels search and cost calculation are implemented in the GPU as a single kernel where, for each pixel, all the corresponding pixels for a certain camera are first calculated and then compared to the main camera. The independence between pixels and depth candidates, makes the storage of intermediate data in DRAM memory unnecessary.

The main problem with this implementation is the high number of memory reads: Y, U and V for all the pixels and depth candidates in a pair of cameras. This issue can be partially alleviated using GPU shared memory. As Y, U and V for the main camera are reused for every depth candidate, they can be stored in on-chip memory, reducing DRAM accesses drastically. This fact, along with an specific structure of overlapped thread blocks also solves the problem of the Y 3×3 window, reusing values of Y not only for a depth candidate, but also for near pixels. This scheme can not be applied to the second camera image, given that its accesses are computed in the process, hence unpredictable.

This kernel is used to calculate an initial cost volume (for pixel and depth candidate) only for a pair of cameras. Consequently, this process is repeated iteratively for all the cameras: whenever a camera produces a lower cost for a pixel and depth candidate, this value is updated in the previous cost volume. To improve this approach, each camera has a weight associated, based on its distance to the main camera.

C. Graph Cuts

Graph Cuts is a well-known global energy minimization algorithm adopted in computer vision to solve different problems, including depth estimation [3], [9]. It is based on the creation of a graph structure where a global-minimum energy cut is to be found. In GoRG, the energy function to be minimized is the following:

$$E(f) = \sum_{p \in \mathcal{P}} R(p) D(p, f_p) + \sum_{(p,q) \in \mathcal{N}} S_{p,q}(p,q) V_{p,q}(f_p, f_q) \tag{2}$$

where f is a labelling that generates a certain energy, a depth map in this case, $\mathcal{P}$ are the image pixels and $\mathcal{N}$ is a neighborhood of pixels.

The data term, $R(p)D(p, f_p)$, measures the cost of assigning one pixel to a certain depth using the information contained in the initial cost volume. In addition, it is scaled by the reliability map (see III-D).

The smoothing term, $S(p,q)V_{p,q}(f_p, f_q)$, measures the cost of associating a depth to a pixel, taking into account its neighbors depths and similarity; for this reason, this term also includes the smoothing map information (see III-D).

Graph Cuts is devised to minimize the energy of a binary-label function and extended it to a multi-label function employing the alpha expansion move method [9], illustrated in Algorithm 1. It begins with an empty depth map that is iteratively updated for each depth candidate Z_n. In every iteration, a graph is initialized and cut to calculate which pixels from the depth map belong to the current depth candidate.

Algorithm 1 Alpha expansion method.

1: *Initialize depth map()*
2: **for** $k = 0 \rightarrow Z_n$ **do**
3: **Step 1:** *Initialize graph()*
4: **Step 2:** *Graph Cut()*
5: **Step 3:** *Update depth map()*
6: **end for**

To develop this algorithm in the GPU, one implementation[1] of CUDA Cuts [19] have been employed. To the best of the authors' knowledge, this library is the most important Graph Cuts CUDA-based acceleration and hence is used.

CUDA Cuts was designed as an image segmentation application, where only one cut is needed and only the case of binary labelling is considered. We have integrated it into the alpha expansion method and optimized communications by removing unnecessary CPU-GPU memory transactions, drastically reducing the time spent managing memory. This also implies the creation of auxiliary GPU functions that allow processing inputs and outputs properly before and after using CUDA Cuts.

The library may yield sub-optimum results [12], which is not a major problem in image segmentation. However, when it is used within the alpha expansion loop, the error is accumulated iteration by iteration, largely impacting the resulting quality, specially in borders and narrow details. In order to overcome this problem, in GoRG, the initial cost volume is filtered layer by layer with a 3×3 gaussian filter and also, the depth map in the alpha expansion method is filtered by a 3×3 median filter every 30 depth steps.

D. Reliability, Smoothing and Motion Maps

To improve depth estimations, reliability, smoothing and motion maps are implemented using the main camera as input.

The reliability map provides an estimation of the texture level per pixel present in the main camera. The reason to do so is the difficulty of matching pixels from different cameras in textureless areas. This estimation is integrated into the initial cost volume, where the cost in textureless areas is magnified.

The smoothing map can be seen as the opposite of the reliability map: it provides an idea of how similar pixels in an area are. With this information, and assuming that, commonly, similar pixels present similar depths, this map is employed to improve the graph generated in the Graph Cuts stage. To do so, edges between similar pixels are reduced according to their level of similarity, favouring these areas to be assigned to the same depth plane.

The motion map is devised as a method to improve the algorithm performance when more than one frame is processed. It

[1] https://github.com/metthueshoo/graphCut

is based on the idea that if no motion exists from one frame to the next, depth values in those areas should remain equal. It is calculated comparing one main camera frame to the previous one, finding which pixels have differences above a threshold. Then, this mask is utilized in the initial cost cube to update only those pixels that have changed. It is also used in Graph Cuts, where only the motion pixels are set to 0 depth; the rest remains with the same depths as in the previous frame (although they can change during the process).

Reliability and smoothing maps are processed in a single, user parametrizable kernel where the differences for each pixel are calculated in a 3×3 window for horizontal and vertical directions. Given the operations independence and low number of memory accesses, which can also benefit from L2 cache due to its locality, this kernel presents a good GPU performance.

Motion map calculation is implemented in the GPU with a kernel that compares two consecutive frames using an user-defined window. It obtains a good GPU performance.

IV. Results and discussion

The results collected comprise the time needed to process the considered frames and the quality of the output for both DERS 8.0 and for GoRG. The former measurement is straightforward, whilst the latter is measured using the following depth-synthesis processing chain:

1) **Depth estimation:** for each camera in a sequence, depth estimations are calculated with DERS 8.0[2] and GoRG.
2) **Synthesis**: using Versatile View Synthesizer (VVS)[3], a virtual camera is set in the place of a real camera and then, a synthesized view is generated from the RGB images and depth of the surrounding cameras.
3) **Comparison:** the synthesized view is compared with the real camera RGB image in the same place using two metrics: WS-PSNR and SSIM.

The platforms used to conduct the experiments are: (i) for DERS 8.0, one node (2× Intel Xeon Gold 6230 processors, 20 cores each) of the Magerit-3 supercomputer of the Supercomputing and Visualization Center of Madrid (CeSViMa); and (ii) for GoRG, a desktop computer with an AMD Ryzen Threadripper 1950X (16 cores) with an NVIDIA Titan Xp GPU. The test material utilized, from the MPEG-I group, can be seen in Table I.

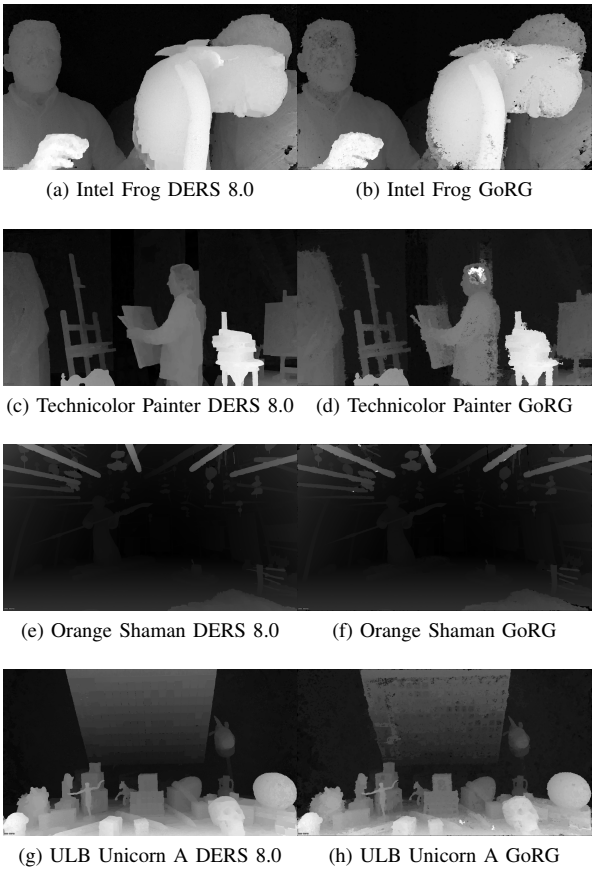
Sequences	Camera Configuration	Frames	Resolution
IntelFrog	linear 14 × 1	8	1920 × 1080
OrangeDancing	arc 14 × 3	8	1920 × 1080
OrangeKitchen	array 5 × 5	8	1920 × 1080
OrangeShaman	array 5 × 5	8	1920 × 1080
PoznanFencing2	arc 10 × 1	8	1920 × 1080
TechnicolorPainter	array 4 × 4	8	2048 × 1088
ULBBabyUnicorn3	array 5 × 3	8	3712 × 2064
ULBUnicornA	array 5 × 5	1	1920 × 1080
ULBUnicornB	array 5 × 3	1	1920 × 1080

TABLE I: Overview of the sequences [1], [14].

Quality and time results can be observed in Table II. Quality results are the product of an average process for all the frames and cameras for each sequence. Time results are the average time to process the number of frames considered for each sequence.

Regarding quality results, the PSNR obtained for GoRG is in average 1.6 dB worse than DERS, with important differences between sequences. For some of them, our results are around 1 dB worse, whilst for others almost 4 dB worse. In spite of that, subjective evaluations carried out by the authors suggest that the quality loss is not as high as might be inferred from PSNR results. For this reason, the SSIM metric is also calculated. Results show that SSIM is closer for all the sequences, with an average loss of 1.20 %.

These quality differences are due to the suboptimal results produced by Graph Cuts in every iteration. Effectively, this error is iteratively accumulated giving a partially incorrect depth map, specially in borders and small depth details. This also explain the better results in case of SSIM, which measures structure correlation and does not penalize erroneous borders as much as PSNR. This is the only source of differences between DERS and GoRG.A comparison of the generated depth maps in DERS 8.0 and GoRG can be seen in Figure 2.



(a) Intel Frog DERS 8.0 (b) Intel Frog GoRG

(c) Technicolor Painter DERS 8.0 (d) Technicolor Painter GoRG

(e) Orange Shaman DERS 8.0 (f) Orange Shaman GoRG

(g) ULB Unicorn A DERS 8.0 (h) ULB Unicorn A GoRG

Fig. 2: Example depth maps for 4 different sequences. On left column, DERS 8.0 estimations, on right column, GoRG estimations.

[2]http://mpegx.int-evry.fr/software/MPEG/Explorations/6DoF/DERS
[3]http://mpegx.int-evry.fr/software/MPEG/Explorations/6DoF/VVS

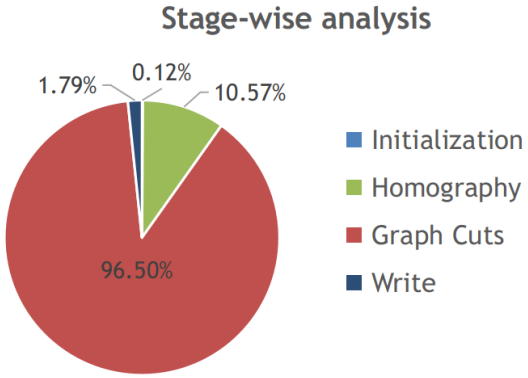


Fig. 3: GoRG stage-wise time analysis. Initialization refers to memory reads and smoothing, reliability and motion maps calculations.

Regarding time results, GoRG achieves an average speed-up of $\times 25$, which greatly outperforms the single-thread DERS 8.0. Nevertheless, our acceleration is far from generating depth estimations at a real-time video rate due to Graph Cuts, which takes more than 95% of the whole time, as can be seen in Figure 3. Although independent pixel operations like in the homography stage can benefit from a embarrassingly parallel acceleration, algorithms that require global synchronizations and memory movements like Graph Cuts can not seize this architecture, unless deeply modified. Furthermore, it is worth mentioning that the acceleration only comes from the GPU implementation, as all the algorithms remain the same.

Sequence	YPSNR(dB)		SSIM		Time/cam(s)	
	DERS	GoRG	DERS	GoRG	DERS	GoRG
IFrog	**27.52**	27.32	0.8232	**0.8312**	3840	**158**
ODancing	**32.45**	29.15	**0.9315**	0.8826	4440	**117**
OKitchen	**31.74**	31.00	**0.9663**	0.9573	1942	**148**
OShaman	**40.46**	36.66	**0.9648**	0.9393	1997	**93**
PFencing2	**27.23**	24.65	**0.8379**	0.8189	2820	**136**
TPainter	34.42	**35.22**	0.9224	**0.9294**	1726	**179**
UBabyUnicorn3	**28.44**	27.16	0.8433	**0.8505**	7324	**555**
UUnicornA	**30.88**	28.11	**0.9649**	0.9465	718	**22**
UUnicornB	**31.53**	30.86	**0.9672**	0.9637	810	**23**

TABLE II: Quality and time results for DERS and the proposal. Better resuls are underlined and bold. Sequence names are abbreviated.

V. Concluding Remarks and Future Lines

We have proposed a GPU-accelerated version of DERS 8.0, depicting the implementation details stage by stage. We demonstrate that most stages of DERS benefit from the GPU architecture due to the independence among pixels computations. The exception is Graph Cuts, which represents more than 95% of the consumed time. Although we also use the GPU for this stage, the constant need for synchronization during the alpha expansion method causes a loss of performance.

GoRG achieves a speed-up of $25\times$ at the cost of an average 1.6 dB loss in PSNR and 1.20% in SSIM. These loses considerably depend on the sequence tested, finding the worst cases when sequences have many small depth details. This is because the erroneous pixels are mainly found at borders and narrow regions.

This work is a first step in the reduction of the computation time of global depth estimation techniques in HSI intraoperative scenarios, where the computational time is expected to be similar or slightly higher, as adding new bands would increase only the processing time in the homography stage. In addition, it can become a valuable tool for researchers and developers as it improves their productivity: algorithm research on one side and the validation cycle of software updates on the other.

Future work will focus on the generation of different multi-view HS data-sets, including medical scenarios, in order to test GoRG with HS images. On the other hand, we will improve the quality of results, minimizing the errors that are mainly introduced in the Graph Cuts stage. As this consumes more than 95% of the computation time, we will explore new strategies for the alpha expansion method that search candidates more efficiently, e.g., using hierarchical or histogram-based methods, instead of the current naive iterative pattern.

VI. Copyrights

References

[1] Exploration Experiments for MPEG-I Visual: 6DoF [N18790]. ISO/IEC JTC1/SC29/WG11 (Oct 2019)

[2] Bonatto, D., Schenkel, A., Lenertz, T., Li, Y., Lafruit, G.: ULB High Density 2d/3d Camera Array data set, version 2 [M41083]. ISO/IEC JTC1/SC29/WG11 (Jul 2017)

[3] Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(9), 1124–1137 (Sep 2004). https://doi.org/10.1109/TPAMI.2004.60

[4] Cheng, F., Huang, K.: Real-time stereo matching for depth estimation using GPU. In: 2015 8th International Conference on Ubi-Media Computing (UMEDIA). pp. 3–6 (Aug 2015). https://doi.org/10.1109/UMEDIA.2015.7297418

[5] Doyen, D., Langlois, T., Vandame, B., Babon, F., Boisson, G., Sabater, N., Gendrot, R., Schubert, A.: Light field content from 16-camera rig [M40010]. ISO/IEC JTC1/SC29/WG11 (Jan 2017)

[6] Duan, X., Ye, X., Li, Y., Li, H.: High Quality Depth Estimation from Monocular Images Based on Depth Prediction and Enhancement Sub-Networks. In: 2018 IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6 (July 2018). https://doi.org/10.1109/ICME.2018.8486539

[7] Fabelo, H., Ortega, S., Szolna, A., Bulters, D., Piñeiro, J.F., Kabwama, S., J-O'Shanahan, A., Bulstrode, H., Bisshopp, S., Kiran, B.R., Ravi, D., Lazcano, R., Madroñal, D., Sosa, C., Espino, C., Marquez, M., De La Luz Plaza, M., Camacho, R., Carrera, D., Hernández, M., Callicó, G.M., Morera Molina, J., Stanciulescu, B., Yang, G., Salvador, R., Juárez, E., Sanz, C., Sarmiento, R.: In-vivo hyperspectral human brain image database for brain cancer detection. IEEE Access **7**, 39098–39116 (2019)

[8] Gopi, A., Reshmi, C.S.: A noninvasive cancer detection using hyperspectral images. In: 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET). pp. 2051–2055 (2017)

[9] Kolmogorov, V., Zabin, R.: What energy functions can be minimized via graph cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(2), 147–159 (Feb 2004). https://doi.org/10.1109/TPAMI.2004.1262177
[10] Mun, J., Ho, Y.: Depth Estimation from Light Field Images via Convolutional Residual Network. In: 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). pp. 1495–1498 (Nov 2018). https://doi.org/10.23919/APSIPA.2018.8659639
[11] Nathan, M., Kabatznik, A.S., Mahmood, A.: Hyperspectral imaging for cancer detection and classification. In: 2018 3rd Biennial South African Biomedical Engineering Conference (SAIBMEC). pp. 1–4 (2018)
[12] Peng, Y., Chen, L., Ou-Yang, F., Chen, W., Yong, J.: JF-Cut: A Parallel Graph Cut Approach for Large-Scale Image and Video. IEEE Transactions on Image Processing **24**(2), 655–666 (Feb 2015). https://doi.org/10.1109/TIP.2014.2378060
[13] Qin, Y., Jin, X., Dai, Q.: GPU-based depth estimation for light field images. In: 2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS). pp. 640–645 (Nov 2017). https://doi.org/10.1109/ISPACS.2017.8266556
[14] Schenkel, A., Bonatto, D., Fachada, S., Guillaume, H., Lafruit, G.: Natural Scenes Datasets for Exploration In 6DOF Navigation. In: 2018 International Conference on 3D Immersion (IC3D). pp. 1–8 (Dec 2018). https://doi.org/10.1109/IC3D.2018.8657865
[15] Senoh, T., Tetsutani, N., Yasuda, H.: Depth Estimation and View Synthesis for Immersive Media. In: 2018 International Conference on 3D Immersion (IC3D). pp. 1–8 (Dec 2018). https://doi.org/10.1109/IC3D.2018.8657842
[16] Senoh, T., Wakunami, K., Sasaki, H., Oi, R., Yamamoto, K.: Fast depth estimation using non-iterative local optimization for super multi-view images. In: 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP). pp. 1042–1046 (Dec 2015). https://doi.org/10.1109/GlobalSIP.2015.7418356
[17] Song, M., Kim, W.: Depth Estimation From a Single Image Using Guided Deep Network. IEEE Access **7**, 142595–142606 (2019). https://doi.org/10.1109/ACCESS.2019.2944937
[18] Stankiewicz, O., Domanski, M., Dziembowski, A., Grzelka, A., Mieloch, D., Samelak, J.: A Free-Viewpoint Television System for Horizontal Virtual Navigation. IEEE Transactions on Multimedia **20**(8), 2182–2195 (Aug 2018). https://doi.org/10.1109/TMM.2018.2790162, https://ieeexplore.ieee.org/document/8247262/
[19] Vineet, V., Narayanan, P.J.: CUDA cuts: Fast graph cuts on the GPU. In: 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. pp. 1–8 (June 2008). https://doi.org/10.1109/CVPRW.2008.4563095
[20] Weijtmans, P.J.C., Shan, C., Tan, T., Brouwer de Koning, S.G., Ruers, T.J.M.: A dual stream network for tumor detection in hyperspectral images. In: 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019). pp. 1256–1259 (2019)
[21] Xu, Y., Wang, Y., Guo, L.: Unsupervised ego-motion and dense depth estimation with monocular video. pp. 1306–1310 (10 2018). https://doi.org/10.1109/ICCT.2018.8600039
[22] Zhang, Zhengyou: Camera Parameters (Intrinsic, Extrinsic), pp. 81–85. Springer US, Boston, MA (2014). https://doi.org/10.1007/978-0-387-31439-6_152
[23] Zhou, W., Li, P., Lumsdaine, A., Lin, L.: Light-field flow: A subpixel-accuracy depth flow estimation with geometric occlusion model from a single light-field image. In: 2017 IEEE International Conference on Image Processing (ICIP). pp. 1632–1636 (Sep 2017). https://doi.org/10.1109/ICIP.2017.8296558