# FAST DEPTH ESTIMATION ON MOBILE PLATFORMS AND FPGA DEVICES

*Marek Domański[1], Jacek Konieczny[2], Maciej Kurc[1], Adam Łuczak[1], Jakub Siast[1], Olgierd Stankiewicz[1], Krzysztof Wegner[1]*

[1] Poznań University of Technology, Chair of Multimedia Telecommunications and Microelectronics, Poznań, Poland

[2] Huawei Technologies, European Research Center, Munich, Germany

## ABSTRACT

In this paper, we propose a fast technique for real-time depth estimation that is implementable on mobile devices like smartphones and tablets. Moreover, the FPGA-based implementations of this technique are also reported. The idea of this technique is to use small-block matching and exploit the recently estimated disparity values in order to enhance spatial consistency of the output disparity map. Using the Middlebury stereoscopic test images, the proposed real-time technique is compared to other techniques known from the references. The obtained results demonstrate high efficiency of the proposed technique implemented both on mobile platforms and FPGA devices.

***Index Terms*** —depth estimation, mobile application, 3D imaging on mobile, disparity on mobile, FPGA-based disparity estimation

## 1. INTRODUCTION

Mobile devices like smartphones and tablets are going to be ready for 3D video applications. Some manufacturers have already started to equip them with autostereoscopic displays and two cameras at least. The prospective mobile applications are going far beyond acquisition and viewing of stereoscopic pictures, but they include, e.g. embedding virtual objects into real scenes or 3D model estimation of real objects. Therefore, the adoption of 3D video technology to mobile devices becomes an important research area. In particular, for most of 3D video applications, efficient real-time depth estimation is still a true challenge, especially when considering implementations on mobile devices with their limited complexity and power consumption. In order to meet these conditions of limited hardware performance and limited power consumption, the respective depth estimation techniques must be relatively simple but they still have to produce high-fidelity depth. Unfortunately, the existing solutions mostly suffer from important limitations.

Therefore, in this paper, we propose a fast technique for real-time depth estimation that is implementable on mobile devices like smartphones and tablets. For the proposed technique, we also demonstrate an efficient implementation on FPGA devices that may be used either directly or as a prototype for ASIC designs.

## 2. STATE OF THE ART

Disparity estimation on mobile devices was already mentioned in the papers dealing with 3D image and video-based applications (e.g. [1,2]).

In general, disparity estimation may be considered as minimization of a cost function that may be defined in various ways, e.g. in order to provide smoothness of the obtained disparity map. The disparity estimation methods [3] may be classified as global and local. The global methods exploit various techniques of global optimization of the whole disparity maps while the local methods provide the disparity maps using local optimization of the disparity map around a pixel only.

The widely used global methods include the graph cuts technique [4] and the belief propagation technique [5]. These contemporary algorithms provide relatively smooth disparity maps. Therefore, the real-time implementations of these methods are subject to extensive research. Yang et al [6] proposed an efficient real-time implementation of the hierarchical belief propagation using Graphics Processing Unit (GPU). Another interesting approach [7] uses simpler global reasoning algorithms based on dynamic programming in horizontal scan lines. Unfortunately, for the global methods, the complexity is rather high – they need both significant processing power and large memory volume. These high demands yield significant problems in mobile implementations. For example, belief-propagation algorithm used for depth estimation with VGA (640×480) resolution and 32 considered levels of disparities, might require as much as 80MB of space for message passing cache. This about of memory, although not significant from memory capacity point of view, has to be read and written-back in each iteration of BP algorithm. Memory bandwidth available on mobile devices and on FPGA devices, together with frame-rates required in real-time depth estimation and practical number of iterations of BP algorithm, therefore lower the usefulness of such approach.

Local methods use various types of block matching in order to find the disparity for each pixel independently. Large number of independent block matchings allow for massive parallelization of the local disparity estimation algorithms. This enables the usage of even large blocks (e.g. 35×35 pixels) that mostly provide disparity maps that are better compared to those obtained using small blocks. Unfortunately, large blocks are very computationally expensive. Hence, the local methods with large blocks are implementable in real-time using the abovementioned parallelization.

Therefore, the local depth estimation algorithms can be efficiently implemented on GPUs even with large windows. Most of them employ adaptively estimated weights in order to adjust to the image content. For example, in works [8-11] weights in the block matching cost are calculated by means of bilateral filtering. Unfortunately, bilateral filtering is computationally expensive and thus its various approximations are studied in the literature. Mattoccia et al [8] divide matching window into small regular blocks in which filter coefficients are kept constant. Wei et al [9] propose two algorithms which employ separable approximation of bilateral filtering and iterative calculation of the matching cost with exponential step size. Others, like [10,11] try to use guided filters for fast computation of the matching cost in block-size-independent *O(N)* time. Even with such mentioned optimizations, the memory bandwidth available on GPUs of mobile devices is still incomparably lower than those available in e.g. mainstream PC GPUs. Therefore, also this approach is inadequate for mobile devices considered in this work.

In this paper, we propose a depth estimation method that would have advantages of both local and global methods.

## 3. GENERAL IDEA

In the proposed algorithm, disparity for a given pixel is estimated using disparities from the previously processed pixels.

At the beginning, there are no previously processed pixels, e.g. in the first column of a picture. Therefore, for the pixels processed first, disparity is estimated locally. For each such a pixel, the disparity value $d$ is chosen, which minimizes the matching cost function $M(\delta)$ calculated with the use of neighboring pixels, for displacement values $\delta$ from a certain search range:

$$d = \underset{\delta}{\arg min} \, [M(\delta)] \qquad (1).$$

$M(\delta)$ can be any matching cost function, e.g. sum of absolute differences (SAD) calculated by block matching.

Such an estimated disparity value $d$ is used to enhance disparity estimations in subsequent pixels. In context of a subsequent pixel, there can be many disparity values estimated for previously processed pixels, therefore, such set will be denoted further as $\{d_{prev\,1}, \dots, d_{prev\,K}\}$. Calculation of the disparity $d$ from Eq. 1 is updated with a penalty value $C$ which is added to the matching cost of disparities not included in $\{d_{prev}\}$ set of disparities:

$$d' = \underset{\delta}{arg\,min} \left[ M(\delta) + \begin{cases} 0 & \text{if } \delta \text{ in } \{d_{prev\,1}, \dots, d_{prev\,K}\} \\ C & \text{otherwise} \end{cases} \right] \qquad (2).$$

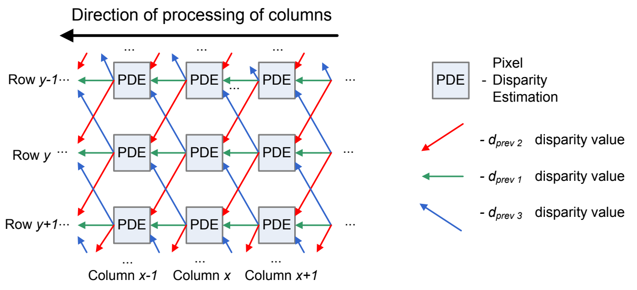
Such output disparity value $d'$ is fed to the subsequent pixels as one of $d_{prev}$ values.

The aim of usage of constant value $C$ is to penalize disparity changes between neighboring pixels, which result solely from noise in the input images. Thus, the value of $C$ can be adjusted experimentally accordingly to the level of noise aggregated within $M(\delta)$. It depends on the size of blocks used for calculation of $M(\delta)$, and also on acquisition parameters of the used stereo pair (e.g. sensitivity of the sensor, exposure time etc.). For pictures acquired with modern cameras those parameters can be read from meta data. Therefore it can be imagined that in a practical application, the penalty value $C$ can be calibrated with use of reference images for a given set of acquisition parameters and then chosen adequately. For the sake of brevity we omit the details on how to choose $C$ parameter.

## 4. DETAILED DESCRIPTION

We assume the usage of two views, which are used to generate dense disparity map for the left view. As for the matching cost $M(\delta)$ we used SAD of red, green and blue components and their horizontal and vertical gradients (9 components in total), calculated between pixels in the left and in the right view. SAD was aggregated over square blocks centered around the processed pixel. In the experiments (cf. Section 6), we used the blocks of size 1×1 to 9×9 pixels.

The order in which pixels must be processed in the proposed algorithm is enforced by dependency of disparity estimation on previously estimated $d_{prev\,1}, \dots, d_{prev\,K}$ disparities. Although in principle, number $K$ can be different for each pixel, we have chosen to use $K = 3$ disparities estimated at previously processed pixels: $d_{prev\,1}, \dots, d_{prev\,3}$, located respectively to the right, to the top-right, and to the bottom-right, from the currently processed pixel (Fig. 1). Consequently, the disparity map is generated in columns, in order from right to left. Such a scheme is beneficial due to occlusions, because we estimate depth for the left view and the right view is used as a reference.



Direction of processing of columns
Row y-1
Row y
Row y+1
PDE
Column x-1
Column x
Column x+1
PDE - Pixel Disparity Estimation
- d_prev 2 disparity value
- d_prev 1 disparity value
- d_prev 3 disparity value

Figure 1. Processing order of pixels (right-to-left) in the implementation of the proposed algorithm and the inference of $d_{prev}$ disparity values

In the presented scheme (Fig. 1), the disparity for each pixel is estimated in Pixel Disparity Estimation (PDE) module, which implements the proposed algorithm. It can be noticed, that, in a given column, estimation for each pixel is independent from each other pixel. This feature is used for parallelization of our algorithm. Of course, in practical implementation, the number of PDE modules is limited, e.g. to 8 on mobile devices or CPU or limited to 20 on FPGA.

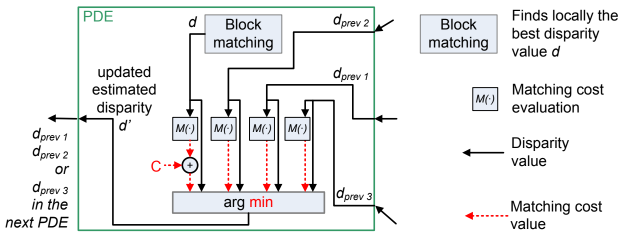


Figure 2. Implementation of the proposed disparity estimation algorithm for a single pixel in a Pixel Disparity Estimation (PDE) module

The PDE module (Fig. 2) implements calculation of Eq. 3, which is a synonymous formulation of Eq. 2 that employs the disparity value $d$ than can be computed with the use of Eq. 1:

$$d' = argmin\big[M(d) + C, M\big(d_{prev\,1}\big), \dots, M\big(d_{prev\,3}\big)\big] \qquad (3).$$

The minimal value of terms in $arg\,min[\cdot]$ is found and the corresponding function argument (e.g. $d$ or $d_{prev\,1}$) is assigned to $d'$. Such formulation allows two implementation optimizations, which we use for software and hardware.

In the first one, $M\big(d_{prev\,1}\big), \dots, M\big(d_{prev\,3}\big)$ values are calculated first. If any of them is lesser than penalty $C$, then calculation of $M(d) + C$ is redundant and can be skipped. Therefore, in such a case, disparity value $d$ is not needed and further block matching for a given pixel can be also skipped. We have used this optimization in our software implementation which yielded substantial speed-up of computations.

In the second optimization, which we have employed in FPGA, disparities $d$ and their corresponding costs $M(d) + C$ are computed first, before the actual algorithm starts. Thanks to that, block matching needed by $M(d)$ can be done very efficiently, because calculations of components of SAD sums can be shared between neighboring pixels. The actual usage of disparity values estimated in previously processed pixels, devised in the proposed algorithm, is performed later in an update step, described by Eq. 3 and depicted in Fig. 2.

## 5. IMPLEMENTATIONS

The proposed algorithm was implemented in software and in hardware (Fig. 3).

For software platform, we used two Android mobile devices equipped with ARM processor: Huawei Ascend P6 and Samsung Galaxy Note 3, and a PC equipped with Intel Core i7 as a reference. The software implementation has been prepared in native multithreaded C++ code, without any assembler subroutines. In all cases, GPU was not used for any computations.

The hardware implementation have been prepared using Verilog language. It runs on Xilinx Artix-7 device with 166 MHz clock. The FPGA implementation employs 20 parallelized PDE modules and requires 27 000 LUTs and 38 blocks of BRAM (36 kbit each).

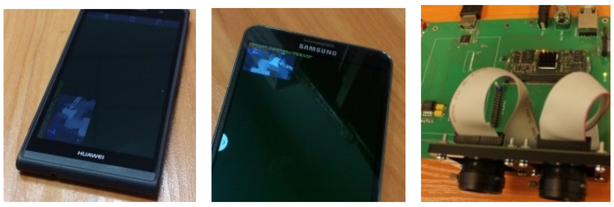


Figure 3. The proposed disparity estimation algorithm implemented on Android mobile devices (Huawei Ascend P6, Samsung Galaxy Note 3) and on the processing board developed by the authors equipped with Xilinx Artix-7 FPGA

## 6. RESULTS

For the quality evaluation of our algorithm, we have used Middlebury stereoscopic images [12]: Tsukuba Venus, Teddy and Cones (Fig. 4). A commonly used bad-pixel ratio [13] has been used as an objective quality index. Bad-pixel ratio presents the percentage of pixels for which disparity is estimated wrongly, in comparison to ground-truth disparity maps, with error greater than 1 disparity level. As usually, the percentage of bad pixels was calculated only for non-occluded regions of the images.

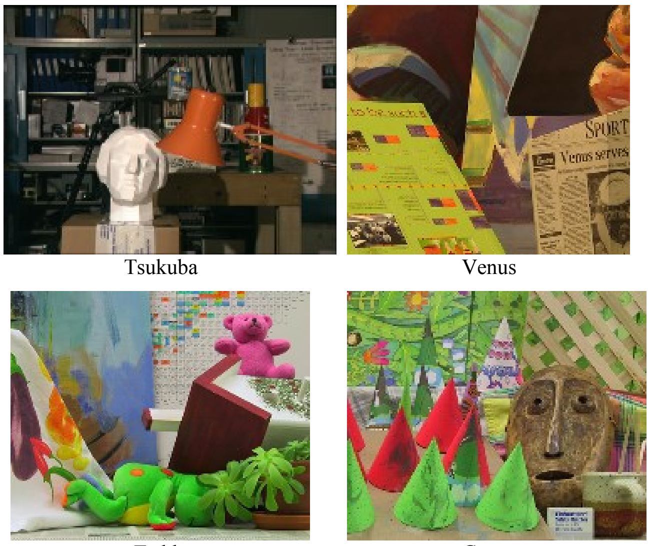


Figure 4. Images from Middlebury database [12] used for evaluation

Table I. Percentage of bad pixel for various block sizes and optimized penalty values *C*

| Block size | Penalty value *C* | Tsukuba | Venus | Teddy | Cones | Avg. |
|---|---|---|---|---|---|---|
| 1×1 | 72 | 17.4 | 10.8 | 9.8 | 25.4 | 15.8 |
| 3×3 | 189 | 10.4 | 7.9 | 8.7 | 17.4 | 11.1 |
| 5×5 | 225 | 8.5 | 7.6 | 7.7 | 16.4 | 10.1 |
| 7×7 | 343 | 6.5 | 8.1 | 7.5 | 16.0 | 9.5 |
| 9×9 | 405 | 6.4 | 8.7 | 6.9 | 16.1 | 9.5 |

Quality of the estimated disparity maps was tested using matching block of sizes in range of 1×1 to 9×9. For each block size we have selected a constant penalty C value that provides the lowest average percentage of bad pixels over four used stereoscopic images. The overall quality of the disparity maps estimated, attained by our algorithm, can be found in Table I.

The increasing of the block size results in reduced average percentage of bad pixels. The highest gains are observed in range between 1×1 to 3×3 and to 5×5. Further increase of the matching block size, over 5×5, provides minor improvements with a drawback of high complexity increase. Exemplary disparity maps estimated for Cones and Teddy images with the use of 5×5 matching block are presented in Fig. 5.

We have also assessed speed of the proposed algorithm in terms of the number of disparity maps estimated per second (frames per second - fps). As for the reference, we have chosen image of size 320×240 pixels and 32 tested disparity levels. For example, in such conditions, our FPGA implementation achieves 1193 fps (Table II). Of course, the provided numbers can be easily scaled to a different resolution, e.g. our FPGA implementation achieves 11 fps for 1920×1080 resolution with 128 tested disparity levels.

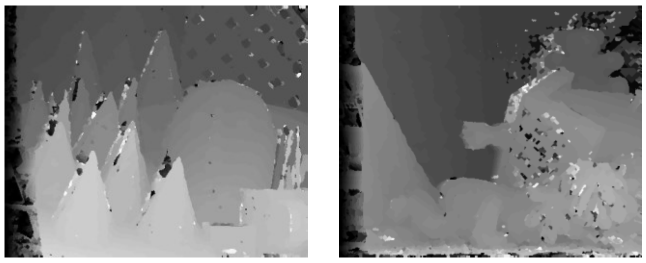


Figure 5. Disparity maps estimated with matching block size 5×5 and penalty value C = 225

The proposed algorithm has been compared to other real-time, state-of-the-art depth estimation techniques (Table II). In the first group, the real-time CPU implementations of our algorithm are shown. For those implementation, block size 3×3 has been used in order to meet real-time requirements. Unfortunately, in the literature we did not find any mobile real-time implementations of disparity estimation algorithms which we could compare to, in terms of bad-pixels and speed. What can be shown instead, is that our PC implementation is comparable with the state-of-the-art disparity estimation algorithms implemented on GPU on PC. Results of those methods, both global and local, are presented in the second and third group in Table II, respectively. The bad-pixel ratio attained by our algorithm (11.11%) is slightly worse, but it must be noted that GPU-implemented algorithms are able to use very big windows (e.g. 33×33). For that, GPU implementations employ hundreds of processors running in parallel, while our software implementation employs a very limited number of computing threads (e.g. 8) and still attains very comparable frame rate (16.7 fps).

As for the hardware implementation, the comparison is very difficult, because different FPGAs allow for vastly different degrees of parallelization etc. We have managed to adjust the results available in literature to conditions in which our hardware implementation was done. The results, presented in the fourth group of algorithms in Table II, show that the proposed algorithm is superficial related to the other presented algorithm in terms of attained frame rate, which is 1193 fps. As for bad-pixel ratio, the proposed algorithm is placed in-between the competitive algorithms implemented on FPGA.

Table II. Implementations of real-time depth estimation methods

| Algorithm / parallelization | Platform | Block size | Average bad pixel ratio [%] | Frames per second* |
|---|---|---|---|---|
| **Proposed algorithm - CPU on mobile devices and PCs** | | | | |
| 8 threads | Samsung Galaxy Note 3 | 3×3 | 11.11 | 5.6 |
| 4 threads | Huawei Ascend P6 | 3×3 | 11.11 | 2.3 |
| 8 threads | Intel Core i7 | 3×3 | 11.11 | 16.7 |
| **GPU on PC: global methods** | | | | |
| RealtimeBP [6] | GeForce 7900 GTX | unknown | 7.69 | 8 |
| RealTimeGPU [7] | Radeon XL1800 | 32×32 | 9.82 | 21 |
| **GPU on PC: local methods** | | | | |
| CostFilter [10] | GeForce GTX 480 | 36×36 | 5.55 | 24 |
| FastBilateral [8] | Tesla C2070 | 39×39 | 7.31 | 21 |
| RealtimeBFV [15] | GeForce 8800 GTX | 33×33 | 7.65 | 46 |
| ESAW [9] | GeForce 8800 GTX | 33×33 | 8.21 | 79 |
| DCBGrid [11] | Quadro FX 5800 | 35×35 | 10.90 | 10 |
| **FPGA** | | | | |
| Proposed, 20 threads | Artix-7 | 5×5 | 10.05 | 1193 |
| SGM [16] | Virtex-5 | 9×9 | 8.43 | 480 ** |
| SBASW [17] | Virtex-5 | 13×13 | 12.27 | 230 |

* - assumed 320x240 images with 32 disparity levels

** - not directly provided in paper; assumed Virtex-5 XC5LX220T with 112.3 MHz clock and design parameters pr = 5, dm = 2 given in the paper

## 7. CONCLUSSIONS

A lightweight depth estimation algorithm has been proposed. It employs a novel regularization method that uses the disparity values from previously processed pixels. These previously processed pixels are located in the neighborhood of the currently processed pixel i.e. to the right, right-top and right-bottom. The proposed regularization method is very simple but efficient. In contrary to e.g. dynamic programming, the cost function values from previously processed pixels are not used. Therefore, the algorithm is fast and eligible both for software implementation on mobile devices with limited processing power and for hardware implementation e.g. on FPGA devices. The results, presented for both platforms, show that the proposed algorithm provides good quality of the estimated depth maps, comparable with other state-of-the-art algorithms, while allowing real-time operation.

## 8. REFERENCES

[1] F. Navarro, S. Cancino, E. Estupinan, “Depth estimation for visually impaired people using mobile devices”, 2014 IEEE 5th Latin American Symp. Circuits Syst. (LASCAS), Santiago, Feb. 2014.

[2] H. Tse Kai, Y. Kawashima, T. Fujisawa, et al., “Implementation and optimization of software depth estimation for arm”, 3DTV-Conf.: The True Vision - Capture, Transm. Display of 3D Video (3DTV-CON), Zurich, 2012.

[3] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, et al. , “High-resolution stereo datasets with subpixel-accurate ground truth”, German Conf. Pattern Recognition (GCPR 2014), Münster, September 2014.

[4] M. Bleyer, M. Gelautz, “Graph-based surface reconstruction from stereo pairs using image segmentation”, SPIE Electronic Imaging Conf., San Jose, pp. 288–299, January 2005.

[5] T. Montserrat, J. Civit, O.D. Escoda, J.-L. Landabaso, “Depth estimation based on multiview matching with depth/color segmentation and memory efficient Belief Propagation”, IEEE Int. Conf. .Image Proc. (ICIP), pp. 2353-2356, Cairo, Nov. 2009.

[6] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, D. Nister, “Realtime global stereo matching using hierarchical belief propagation”, British Machine Vision Conference, pp. 989–998, Edinburgh, 2006.

[7] L. Wang, M. Liao, M. Gong, R. Yang, D. Nister, “High-quality realtime stereo using adaptive cost aggregation and dynamic programming”, Third Int. Symp. 3D Data Processing, Visualization, and Transmission (3DPVT’06), Washington, pp. 798–805, 2006.

[8] S. Mattoccia, M. Viti, and F. Ries, “Near real-time fast bilateral stereo on the GPU”, IEEE Comp. Society Conf. Computer Vision and Pattern Recog. Workshops (CVPRW 2011), pp. 136 –143, Colorado Springs, June 2011.

[9] W. Yu, T. Chen, F. Franchetti, J. C. Hoe, “High performance stereo vision designed for massively data parallel platforms,” IEEE Trans. Circuits Syst. Video Techn., vol. 20, pp. 1509–1519, Nov. 2010.

[10] A. Hosni, C. Rhemann et al, “Temporally consistent disparity and optical flow via efficient spatio-temporal filtering”, Advances in Image and Video Technology (Y.-S. Ho, ed.), vol. 7087, Lecture Notes in Comp. Science, pp. 165–177, Springer, 2012.

[11] C. Richardt, D. Orr, I. Davies, A. Criminisi, N.A.Dodgson, “Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid”, European Conf. Computer Vision (ECCV), Lecture Notes in Computer Science, pp. 510–523, September 2010.

[12] D. Scharstein, R. Szeliski, “High-accuracy stereo depth maps using structured light”, IEEE Comp. Society Conf. Computer Vision and Pattern Recogn. (CVPR 2003), vol. 1, pp. 195-202, Madison, June 2003.

[13] D. Scharstein, R. Szeliski. Middlebury stereo evaluation - version 2, 2010, http://vision.middlebury.edu/stereo/eval/.

[14] Qingxiong Yang, Liang Wang, Ruigang Yang, et al. “Realtime global stereo matching using hierarchical belief propagation”, British Machine Vision Conference, pp. 989–998, Edinburgh, 2006.

[15] Ke Zhang, Jiangbo Lu, Qiong Yang, G. Lafruit, et al. “Real-time and accurate stereo: A scalable approach with bitwise fast voting on CUDA”, IEEE Trans. Circuits Syst. Video Techno., vol. 21, pp. 867 –878, July 2011.

[16] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, P. Pirsch, "Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation”, Int. Conf. on Embedded Computer Systems (SAMOS), pp.93,101, July 2010.

[17] C. Ttofis, T. Theocharides, “Towards accurate hardware stereo correspondence: A real-time FPGA implementation of a segmentation-based adaptive support weight algorithm”, Europe Conf. & Exh. Design, Automation & Test (DATE), 2012, pp. 703-708, Dresden, March 2012.