Journal of Real-Time Image Processing manuscript No.
(will be inserted by the editor)

Real-Time Shadow Detection using Multi-Channel Binarization and Noise Removal

Márcio C. F. Macedo · Verônica P. Nascimento · Antonio C. S. Souza

Received: date / Accepted: date

Abstract High-quality automatic shadow detection remains a challenging problem in image processing and computer vision. Existing techniques for shadow detection typically make use of deep learning strategies to obtain accurate shadow detection results, at the cost of demanding high processing time, making their use unsuitable for augmented reality and robotic applications. In this paper, we propose a novel approach to perform high-quality shadow detection in real time. To do so, we convert an input image into different color spaces to perform multi-channel binarization and detect different shadow regions in the image. Then, a filtering algorithm is proposed to remove the noisy false-positive shadow regions on the basis of their sizes. Experimental results evaluated in two different datasets show that the proposed approach may run entirely on the GPU, requiring only $\approx$ 13 milliseconds to detect shadows in an image with 3840×2160 (4k) resolution. That makes our approach about 1.8 (66 $\times$) to 4.6 (37,284 $\times$) orders of magnitude faster than related work for 4k resolution images, at the cost of only $\approx$ 5% of accuracy loss compared to the best results achieved for each dataset.

Keywords Shadow detection · Parallel processing · Binarization · Noise removal · Real time

1 Introduction

Shadow detection is desirable in several applications because it improves the visual understanding of a scene. Specifically for augmented reality applications, shadows must be detected in real time to assist in the estimation of the dynamic lighting conditions of a real scene, further enabling the photorealistic rendering of a virtual content in an augmented scene [16].

Shadow detection in single images is a difficult task because not only common images include limited information of a given scene, but these images may also include noise and distortion artifacts produced by the camera. However, despite these shortcomings, humans can easily detect shadows when looking at single images. In this sense, thanks to the collection and availability of different datasets for shadow detection [27,6,24,26], machine learning techniques have been largely applied to train the computer to be able to solve this task of shadow detection in single images (Section 2). While high accuracy rates have been achieved by the use of deep neural networks to train the algorithms, little effort has been given to the design of efficient, real-time solutions to the shadow detection problem.

In this work, we propose, to the best of our knowledge, the first accurate algorithm for real-time shadow detection that achieves a running time suitable for augmented reality and robotic applications. We show that, by converting the input image into distinct color spaces, that are used as a basis to a multi-channel binarization, we are able to detect the potential shadow regions located in the image. Then, the noisy false-positive shadow regions previously estimated are effectively detected and removed by the use of a new filtering algorithm.

In this sense, our main contributions can be summarized as follows:

1. A multi-channel binarization algorithm that locates potential shadow regions in an image;
2. A filtering algorithm that removes small, noisy false-positive shadow regions from a binary image, further

M. Macedo
Federal University of Bahia, Brazil
Tel.: +55-71-32836273
Fax: +55-71-32836308
E-mail: marciocfmacedo@gmail.com

V. Nascimento, A. Souza
Federal Institute of Bahia, Brazil

improving the accuracy of our multi-channel binarization solution;
3. An efficient shadow detection algorithm that makes use of multi-channel binarization and noise removal to achieve results five times faster than the state-of-the-art, while obtaining an accuracy loss of only $\approx 5\%$, in terms of Balance Error Rate (BER), as compared to best results achieved by related work;
4. An implementation of the proposed algorithm on the Graphics Processing Unit (GPU), outperforming state-of-the-art works in terms of processing time by 1.8 to 4.6 orders of magnitude for high-resolution images;

2 Related Work

In this section, we focus on the review of relevant related work that used a data-driven approach for single image shadow detection. An in-depth review of the existing shadow detection techniques for moving cast shadows is beyond the scope of this paper. We refer the reader to the surveys of Al-Najdawi et al. [1] and Sanin et al. [20].

Zhu et al. [27] were one of the first authors to handle shadow detection from single images using a data-driven approach. They used handcrafted shadow variant and shadow invariant features like intensity difference, local maximum intensity, smoothness, skewness, gradient similarity, texture similarity, discrete entropy and edge response to perform shadow detection on monochromatic images. Since no data set was available at the time of their work, they proposed a novel dataset, today called University of Central Florida (UCF) dataset.

Guo et al. [6,7] proposed that the shadow detection must be performed by comparing the illumination condition of different regions of the image. By using measurements like the distance between color and texture histograms, ratio of color intensity, chromatic alignment and Euclidean distance between different regions, the authors trained a Support Vector Machine (SVM) to improve the accuracy of the shadow detection on the UCF dataset and on the University of Illinois at Urbana Champaign (UIUC) dataset proposed by the own authors.

Vicente et al. [25] extended the work of Guo et al. by first performing superpixel segmentation to cluster the image into different regions. Then, a single SVM classifier [25] or a multi-kernel Least-Squares SVM classifier [22,23] with leave-one-out optimization could be used to improve the accuracy of the work of Guo et al. on the UIUC dataset. The drawback of these approaches is that they are not scalable for large training data. To solve this problem, the authors [24] extended their work to use stacked Convolutional Neural Networks (CNN) with a Fully Connected Network. This approach handled better the new large Stony Brook University (SBU) dataset proposed by the authors.

Despite the effort of the aforementioned techniques in the design of handcrafted features for shadow detection, with the advent and popularity of deep learning [15], many techniques have been proposed to use this technology to automatically learn the best features for shadow detection.

Khan et al. [11,12] were the first to make use of deep learning for shadow detection. They trained one CNN for detecting shadow regions over segmented superpixels and another CNN for detecting shadow boundaries, on the basis of boundary regions enhanced with bilateral filtering. Finally, an edge map computed according to the difference between adjacent pixels was used in an optimization problem solved with CRF to enforce local consistency.

Shen et al. [21] trained a CNN by modelling the shadow detection problem as an optimization function on the basis of shadow and brightness measurements computed between distinct regions of the image.

Nguyen et al. [18] adapted the use of a Conditional Generative Adversarial Network (CGAN) [4,17] to solve the task of shadow detection from single images. Afterwards, Wang et al. [26] showed that, by coupling shadow detection and shadow removal strategies to train Stacked CGAN on the newly Image Shadow Triplets (ISTD) dataset, they could improve the accuracy of the shadow detection.

Indeed, the use of deep learning for shadow detection greatly improved the accuracy of the shadow detection, allowing the generation of the current state-of-the-art results in all the four datasets already proposed in this field. However, as reported by the authors of related work, most of the existing solutions demand hours to train the classifiers and seconds to detect the shadows of a single image. Hence, until very recently, little effort has been given on the design of interactive/real-time solutions to the problem of shadow detection.

Hosseinzadeh et al. [9] segmented the image into superpixels, then trained an SVM with the handcrafted features proposed in related work [6,25] to estimate the probability that a given superpixel is a shadow region. Afterwards, such an estimate was used as an input for a patched CNN, that was used to predict the location of the shadow regions. In their work, the authors showed that this solution greatly decreases both training and testing times. However, the algorithm is less accurate than related work and still demands seconds to perform shadow detection on the available datasets.

Le et al. [14] made use of an adversarial shadow attenuation framework composed of a shadow detection and an attenuator network to achieve high-quality results at interactive frame rates.

Inspired by the spatial Recurrent Neural Networks, Hu et al. [10] designed a novel deep learning module to extract direction-aware spatial context features that allowed the evaluation of an input image in a global manner, improving the accuracy of the shadow detection.

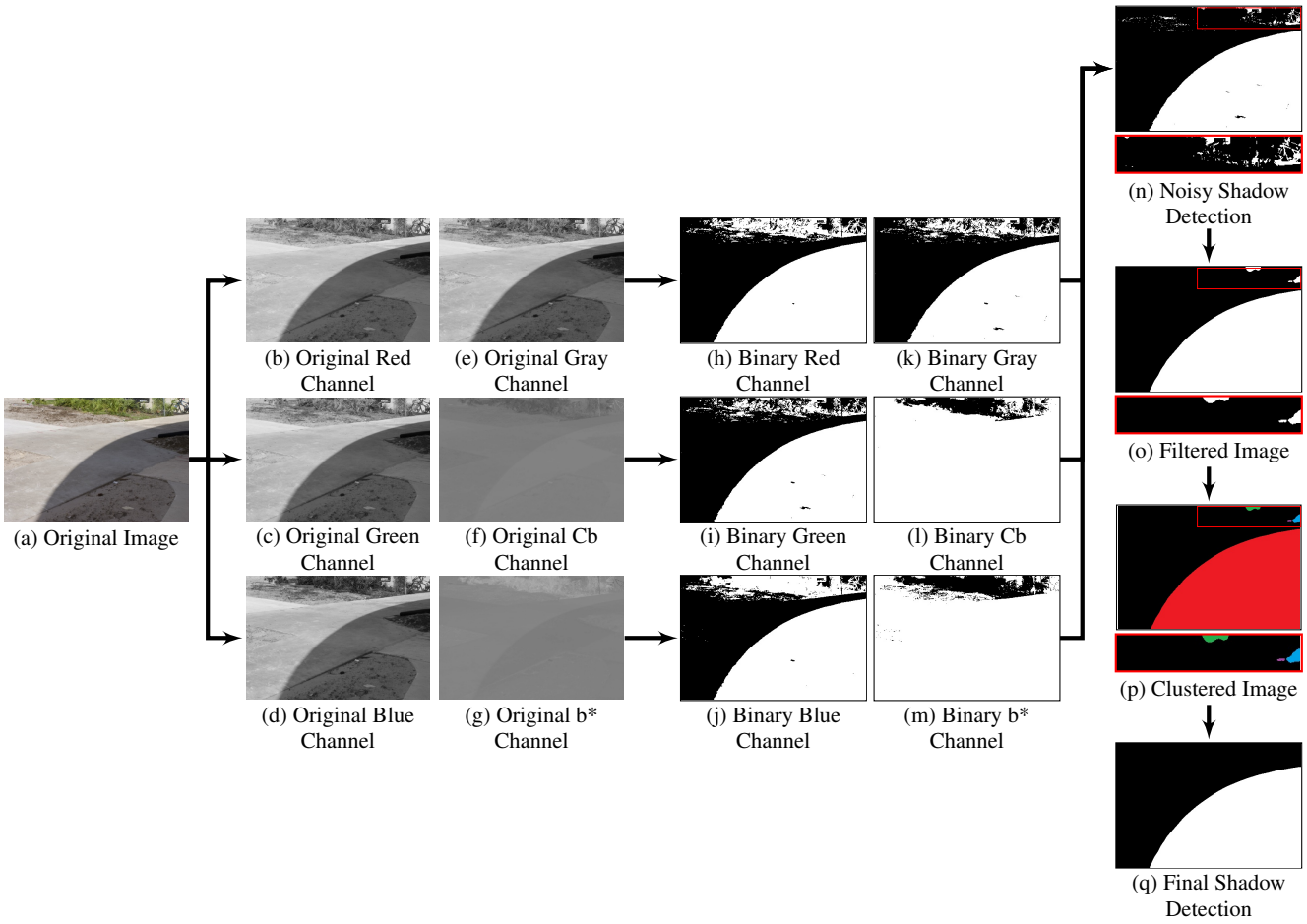


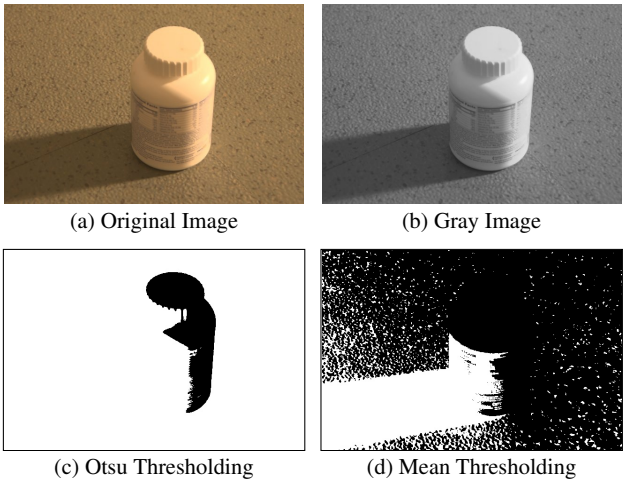
Fig. 1 An overview of the proposed real-time shadow detection algorithm. Given an RGB color image (a), we split the original image into six different channels: red (b), green (c), blue (d), gray (e), Cb (f) and b* (g). Then, we binarize each channel separately, according to the mean values estimated for each channel (h-m). Afterwards, we multiply some of the binary images previously computed to obtain a noisy shadow detection (n). To remove the false-positive shadow regions, we first apply a local filtering approach that removes isolated pixels labeled as shadow (o). Then, we apply a global filtering approach that clusters the remaining shadow regions and discards the false-positive ones that are too small in the image (p). With such an algorithm, we are able to generate high-quality shadow detection results (q).

Despite the promising results achieved by the aforementioned techniques, they are still inadequate for real-time applications because their shadow detection algorithms consume more than 30 milliseconds for a low-sized (*e.g.,* 640×480) image. A photorealistic augmented reality application, for instance, must take care of other costly operations, such as camera pose estimation, tracking, lighting estimation and virtual data rendering [16]. In this sense, the shadow detection step must run as fast as possible, consuming just a few milliseconds of the total frame time of the application.

In this work, we aim to solve this problem of real-time shadow detection by means of multi-channel binarization and noisy false-positive shadow removal. Without relying on the use of training classifiers nor deep learning, we design a practical approach that makes use of handcrafted features to achieve a processing time suitable for augmented reality and robotic applications. By implementing our technique fully on the GPU, we show that we can achieve results orders of magnitude faster than the fastest approaches proposed so far [9, 14, 10]. By evaluating the proposed approach on two of the largest datasets available in the literature, we show that our approach is not only real time, but is also accurate and generates visually pleasant shadow detection results.

3 Real-Time Shadow Detection

In this section, we introduce our approach for real-time shadow detection. An overview is shown in Figure 1. Taking a colored RGB image as input (Figure 1-(a)), we convert the image into different color spaces (Figures 1-(b, c, d, e, f, g)), which are used as a basis for a multi-channel binarization (Figures 1-(h, i, j, k, l, m)). That gives us a coarse estimate of where the shadows are located in the scene (Figure 1-(n)). To improve the accuracy of the shadow detection, local (Figure 1-(o)) and global filtering steps (Figure 1-(p)) are used to



(a) Original Image (b) Gray Image

(c) Otsu Thresholding (d) Mean Thresholding

Fig. 2 For an image (a) with a bright object (b), Otsu thresholding overestimates the presence of shadows (c). Mean thresholding is more robust to illumination conditions (d).

remove non-shadow regions from the final detection (Figure 1-(q)). These steps are detailed in the next subsections.

3.1 Color Space Conversion

Given a colored RGB image (Figure 1-(a)), the first step of our approach converts the colored image into a grayscale image, since shadows are typically located in the darker regions of the image, which can be easily detected by analyzing the gray component of an image. Afterwards, we proceed with the conversion of the input colored image to other two color spaces, namely: YCrCb and CIE L*a*b*. As already stated by related work [6,7,25,23,9], the CIE L*a*b* color space is useful for shadow detection, mainly because the b* channel is not invariant to the presence of shadows. Since the b* channel handles the blue-yellow color spectrum and is efficient for shadow detection, we also compute the Cb channel in the YCrCb color space to take into account the blue difference in the chroma components for shadow detection.

After the conversion of the original image to different color spaces, we split the relevant channels (red, green, blue, gray, Cb, and b*) into separate images, as shown in Figures 1-(b, c, d, e, f, g). The separate use of red, green and blue channels allows us to discard non-shadow regions that have high intensity in only one of those channels (*e.g.*, brick, sky, sea, tree, grass).

3.2 Multi-channel Binarization

To detect where the shadow is located in the image, we need to binarize each of the channels previously computed on the basis a threshold. Otsu thresholding is a common

Algorithm 1 Shadow detection by multi-channel binarization

Input: I : the set of binary images,
μ : the set of mean values
w : image's width,
h : image's height,
β : a threshold value
Output: $\mathsf{I}_{\text{shadow}}$: an image with shadows detected

procedure DETECTSHADOW(I, μ, w, h, β)
  $\mathsf{I}_{\text{shadow}} \leftarrow \mathsf{I}_{\text{gray}} \mathsf{I}_{\text{Cb}}$;
  if $\mu_{\text{r}} > \mu_{\text{g}}$ or $\mu_{\text{r}} > \mu_{\text{b}}$ **then**
    $\mathsf{I}_{\text{shadow}} \leftarrow \mathsf{I}_{\text{shadow}} \mathsf{I}_{\text{r}}$;
  end if
  if $\mu_{\text{g}} > \mu_{\text{r}}$ or $\mu_{\text{g}} > \mu_{\text{b}}$ **then**
    $\mathsf{I}_{\text{shadow}} \leftarrow \mathsf{I}_{\text{shadow}} \mathsf{I}_{\text{g}}$;
  end if
  if $\mu_{\text{b}} > \mu_{\text{r}}$ or $\mu_{\text{b}} > \mu_{\text{g}}$ **then**
    $\mathsf{I}_{\text{shadow}} \leftarrow \mathsf{I}_{\text{shadow}} \mathsf{I}_{\text{b}}$;
  end if
  $\mathsf{I}_{\text{diff}} \leftarrow |\mathsf{I}_{\text{shadow}} - \mathsf{I}_{\text{b*}}|$;
  $c \leftarrow$ count zero values from I_{diff};
  if $\frac{c}{wh} > \beta$ **then**
    $\mathsf{I}_{\text{shadow}} \leftarrow \mathsf{I}_{\text{shadow}} \mathsf{I}_{\text{b*}}$;
  end if
  return $\mathsf{I}_{\text{shadow}}$;
end procedure

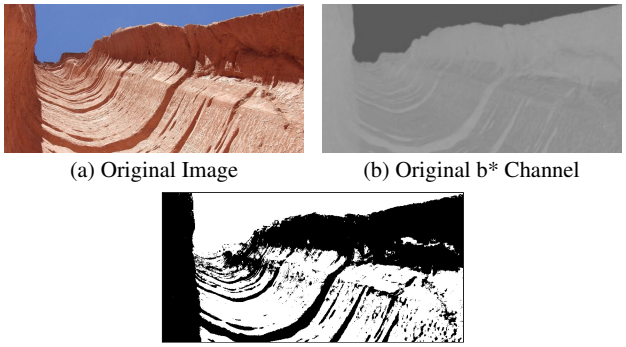
alternative for accurate binarization, however, in practice, we have found that this technique tends to overestimate the presence of shadows in images with bright objects (Figure 2-(c)). Hence, we opted to perform binarization via mean thresholding. Let us define the set $\mathsf{I} = \{\mathsf{I}_{\text{r}}, \mathsf{I}_{\text{g}}, \mathsf{I}_{\text{b}}, \mathsf{I}_{\text{gray}}, \mathsf{I}_{\text{Cb}}, \mathsf{I}_{\text{b*}}\}$, composed of the red, green, blue, gray, Cb and b* single-channel images with width w and height h, respectively. So, for each one of the six channel images, we estimate the mean values $\mu = \{\mu_{\text{r}}, \mu_{\text{g}}, \mu_{\text{b}}, \mu_{\text{gray}}, \mu_{\text{Cb}}, \mu_{\text{b*}}\}$ of their pixels and scale each mean value by a small scale factor $\alpha = \{\alpha_{\text{r}}, \alpha_{\text{g}}, \alpha_{\text{b}}, \alpha_{\text{gray}}, \alpha_{\text{Cb}}, \alpha_{\text{b*}}\}$, that is measured experimentally. Then, for each pixel $\mathsf{I}(i,j)$ of the set I, we make use of the corresponding mean value μ and scale factor α to perform traditional binarization as follows

$$\mathsf{I}(i,j) = \begin{cases} 0 & \text{if } \mathsf{I}(i,j) > \mu\alpha, \\ 1 & \text{otherwise,} \end{cases} \tag{1}$$

where 0 means that the pixel is potentially located out of shadow and 1 otherwise. Specially for the Cb image, we change Eq. (1) to output 1 if $\mathsf{I}(i,j) > \mu\alpha$ and 0 otherwise, because this channel tends to brighten shadow regions in the image. This effect is visible in Figure 1-(f).

By using the solution proposed in Eq. (1), whose result is visible in Figures 1-(h, i, j, k, l, m), we are able to minimize the shadow overestimation problem produced by Otsu thresholding (Figure 2-(d)), while keeping the binarization process simple and fast.

Given the six binary images generated with mean thresholding, we need to integrate them into a single binary image



(a) Original Image (b) Original b* Channel

(c) Binary b* Channel

Fig. 3 Pixels located in the bright sky (a) tend to be darker than shadow pixels for the b* channel (b), affecting the shadow detection procedure (c).

I_{shadow} with the shadows detected. An algorithmic overview of this step is listed in Algorithm 1. Rather than producing the single binary image as a direct per-pixel product between six binary images, we check the relevance of four of the six binary images before including them into the final product. I_{gray} and I_{Cb} images are relevant to shadow detection, being always used in the per-pixel product (Line 2 of Algorithm 1). We have found that, out of the RGB channels, only the most relevant ones should be used for shadow detection. This relevance is again measured in terms of mean values. The channel with the lower mean value is discarded from shadow detection, because we consider that their influence in the visual perception of the image is lower than the other channels, and, due to the low intensity of pixels of this color channel, shadow pixels may be incorrectly discarded during the binarization process (Lines 3-11 of Algorithm 1). With respect to the b* channel, we are careful before including their influence into the final product. For images with a visible bright sky, the binarization of I_{b*} produces the inverse of the expected effect, because the sky region tends to be darker than the shadow region (Figure 3). To solve this problem, we first check whether I_{b*} and I_{shadow} are relatively similar to each other. To do so, we take the absolute difference between both images (Line 12 of Algorithm 1) and count the number of pixels that have the same intensity in both images (Line 13 of Algorithm 1). If the similarity between images is above a threshold β, we include I_{b*} in the per-pixel product to generate the final image I_{shadow} (Lines 14-16 of Algorithm 1).

3.3 Noise Removal

As shown in Figure 1-(n), the output of the previous step is the generation of a single image that represents potential shadow pixels with the white color. However, images captured with conventional cameras, such as webcams and built-in cameras of smartphones, typically contain noise and other artifacts that may affect the quality of the shadow detection procedure. In this sense, the multi-channel binarization of Algorithm 1 may incorrectly label several small regions of the image as false-positive shadow regions. In order to solve this problem and minimize the false-positive shadow regions of I_{shadow}, we have designed a two-step filtering algorithm, that is described as follows.

Due to the noise present in the original image, some pixels incorrectly assumed to be in shadow are isolated inside a non-shadow region (see the shadow pixels in close up of Figure 1-(n)) and vice-versa (see the non-shadow pixels in the shadow quadrant shown in Figure 1-(n)). To make the visibility condition of these isolated pixels agree with the one of their surrounding neighbours, we apply a local filter that analyzes the neighbourhood region centered at each pixel and determines whether the visibility condition of the center pixel must be changed to agree with the visibility condition of its neighbours.

Following the definition of Eq. (1), let us recall that non-shadow and shadow pixels store 0 and 1 intensity values, respectively. Then, for a filter with kernel order of $\kappa \times \kappa$ centered at each pixel of I_{shadow}, we can detect whether the region inside the local filter is dominated by non-shadow or shadow pixels simply by counting which one of the values is more present in the filter kernel, 0 or 1. Considering I_{input} and I_{output} as the input and output images of the proposed filter, this process can be done efficiently, for each pixel $I_{output}(i,j)$, with box filtering

$$I_{output}(i,j) = \frac{\sum_{x=i-\kappa}^{i+\kappa}\sum_{y=j-\kappa}^{j+\kappa} I_{input}(x,y)}{\kappa^2}, \quad (2)$$

$$I_{output}(i,j) = \begin{cases} 0 & \text{if } I_{output}(i,j) \leq \phi, \\ 1 & \text{otherwise.} \end{cases}$$

In Eq. (2), we use a simple box filter to estimate the average intensity inside the filter kernel. Then, if the output intensity of the box filter is larger than an intensity value ϕ, or, in other words, if more than a pre-defined percentage of the pixels located inside the kernel have intensity 1, the filter outputs that the pixel is located in a shadow region and must be put in shadow. Likewise, if the filter outputs that the average intensity of the filter region is lower or equal than ϕ, the pixel is located in a non-shadow region and must be considered as a non-shadow pixel.

As shown in Figure 1-(o), by computing Eq. (2) over I_{shadow}, we can minimize the false-positive shadow regions generated by the multi-channel binarization algorithm listed in Algorithm 1. However, as shown in the closeup of Figure 1-(o), the use of a local filter works well for isolated false-positive pixels, but does not work well in isolated false-positive shadow regions, with dozens of false-positive pixels inside them. To further suppress the remaining noisy false-positive shadow regions, the second step of our noise re-

moval approach makes use of a global filtering algorithm that analyzes the size of each shadow region located in the image to determine whether the shadow region is prominent from noise and must be removed. We draw this strategy from the observation that shadow regions estimated from noise artifacts are generally small, since noise artifacts tend to be small and distributed over the image.

To determine the size of a shadow region, we first cluster each shadow region on the basis of a connected components labeling algorithm. Connected components labeling aims to cluster different, connected regions of an image, associating a unique identifier (ID) to each one of them (Figure 1-(p)). To perform the clustering of the pixels that belong to the same shadow regions, we have used the fast and accurate connected components labeling algorithm proposed in [5]. In this algorithm, labeling is performed in two steps, in a block-based manner. In the first step, shadow pixels located in the same 2×2 block belong to the same shadow region and are associated with a unique label. Then, a decision table is used to check whether different labels are equivalent to the same shadow region. Finally, in the second step of the algorithm, each pixel in a shadow region is assigned to an ID that is representative of the shadow region in which the pixel belongs to. A false-color result of the application of this connected components labeling over a binary image is shown in Figure 1-(p).

After locating the pixels that belong to the same shadow region, we can count the number of pixels of each shadow region and discard the noisy false-positive shadow regions whose sizes are lower than ω percent of the image, shadow regions that possibly contain noisy pixels incorrectly estimated to be in shadow. As shown in Figure 1-(q), this solution effectively suppresses remaining artifacts, enhancing the final shadow detection.

4 GPU-Based Shadow Detection

In this section, we show how the pipeline depicted in Figure 1 can be fully parallelized for GPU architectures.

The first step of color space conversion (Section 3.1) can be easily implemented in a single kernel on the GPU. Since the color space conversion operation is pixel independent, each thread is able to compute gray, Cb and b* intensities of the corresponding pixel in the input RGB image in parallel. Also, each thread may copy the corresponding individual red, green and blue channels of the input RGB image into separate images, resulting the generation of the images shown in Figures 1-(b, c, d, e, f, g). To further optimize this solution, all the images may be stored in pitched memory, favoring coalesced memory access.

The next step consists in the mean intensity estimation for each one of the six images previously computed. To solve this problem, efficient parallel prefix sum [8] can be used to sum the pixel intensities of each image and estimate their mean values.

Similarly to color space conversion, mean thresholding (Section 3.2) is inherently a pixel-independent operation that can be performed in a single kernel. In this case, each thread may perform in-place binarization of its corresponding pixel in each one of the six channels in parallel, according to Eq. (1).

The multi-channel method for shadow detection (Algorithm 1) is easily parallelizable, since the algorithm mostly consists of pixel-independent operations between distinct binary images (Lines 2-12, 14-16 of Algorithm 1). The exception to this rule happens in the counting step of Line 13 of Algorithm 1, that requires the use of parallel prefix sum for efficient GPU computation.

As we show in Section 3.3, our filtering algorithm consists of local and global filtering strategies for false-positive shadow region removal. As for the local filtering algorithm that aims to minimize noisy false-positive shadow and non-shadow pixels, a simple box filtering is used in Eq. (2) to determine whether a region inside the filter kernel is dominated by shadow pixels and change the intensity of the center pixel accordingly. Considering that box filter is a separable spatial linear filter, efficient solutions that use shared memory and coalesced memory access already exist for optimized filtering [19].

Our global filtering strategy to minimize false-positive shadow regions determines the size of the shadow regions by means of connected components labeling and removes the shadow regions whose sizes are smaller than a pre-defined threshold. Rather than adapting the CPU-based solution proposed in [5] to the GPU, we make use of the optimized GPU-based connected components labeling strategy proposed in [3]. In this algorithm, the image is divided into separate blocks. Then, each pixel located in a shadow region is compared to its neighbours in the same block in a row-column order to determine whether they are in the same shadow region. The value and the label of each pixel are loaded into the shared memory to speed up the local labeling process. In the next step, the algorithm scans the pixels located in the boundary of each block to solve label equivalences to the same shadow region. In the final step, the algorithm associates each pixel to the unique ID of the shadow region where it is located.

The GPU-based connected components labeling algorithm outputs, for each pixel located in the shadow region, the ID of the shadow region in which the pixel is located. The shadow region ID is typically the position of a pixel that represents the shadow region. In this sense, special care must be taken on how to use this data to perform noisy false-positive shadow region removal entirely on the GPU. A naïve approach would be to calculate a histogram over

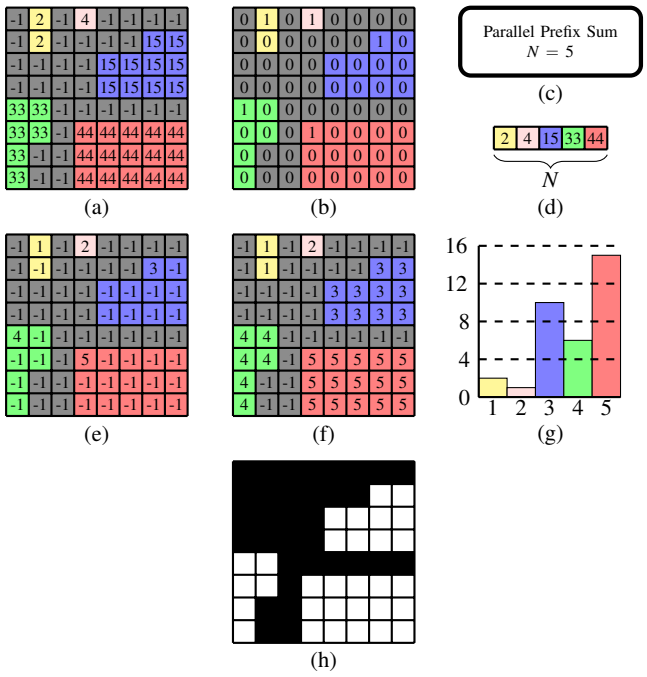


Fig. 4 Given a clustered image with non-normalized shadow region IDs (a), we first binarize such an image by assigning to intensity 1 the representative pixels of each shadow region (b). Then, the parallel prefix sum is computed over the image to estimate the number of shadow region N in the image (c), to allow the building of a vector with only shadow region IDs (d). Next, a hash image is built to assign the new shadow region ID of the representative pixels (e). Afterwards, this hash image is used to update the original clustered image with normalized shadow region IDs (f). Finally, a histogram (g) is computed over the normalized shadow region IDs, to allow noisy false-positive shadow region removal (h).

the clustered image (*i.e.,* the image with each shadow region represented by a unique ID), in order to obtain, for each shadow region ID, the number of pixels located inside it. Without knowing in advance the number of shadow regions N in the image, the range of values to be considered by the histogram would be to the order of the image size, increasing the memory requirements and decreasing the processing time of the histogram calculation. To optimize this approach, let us call representative pixels the pixels that are located in a position that matches a shadow region ID. By the binarization of the clustered image (Figure 4-(a)), assigning the value 1 for the representative pixels of the shadow regions (Figure 4-(b)), and 0 otherwise, we can estimate N by running the parallel prefix sum over such a binary cluster image (Figure 4-(c)). Afterwards, we copy only the shadow region IDs to a smaller vector, whose size is N (Figure 4-(d)). Such a smaller vector is used to build a hash structure, whose key is the previous shadow region ID, and whose value is the new normalized ID, that goes from 1 to N (Figure 4-(e)). On the basis of this hash structure, we update the clustered image (Figure 4-(f)), and calculate the histogram of the image for N bins (Figure 4-(g)). Finally, for each thread, in parallel,

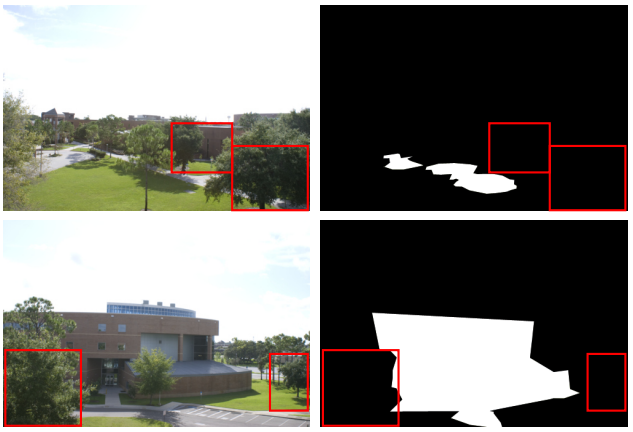


Fig. 5 The regions inside the red rectangles show shadow pixels (a) that are not included in the ground-truth shadow masks (b). These images (a) are incorrectly labeled (b) in the UCF dataset.

the algorithm checks if a given pixel falls in a shadow region whose histogram returns that the shadow region size occupies less than ω percent of the image. In this case, the pixel is considered to be located in a false-positive shadow region and is further discarded from the image (Figure 4-(h)).

As we show in the next section, the proposed GPU solution is able to speed up the processing time of the shadow detection, mainly for high-resolution image sizes.

5 Results and Discussion

In this section, we evaluate different shadow detection techniques in terms of accuracy and processing time. As shown in Section 2, a large amount of works has been proposed in this field. However, we restrict our evaluations to the works that have been evaluated in the most recent, large-scale, data sets available in the literature. In this sense, we compare our real-time shadow detection approach, hereafter named as RTSD or RTSD-CPU/RTSD-GPU to differentiate CPU and GPU versions of our algorithm, with the traditional Unary-Pairwise technique of Guo et al. [6] and the latest state-of-the-art techniques: Stacked-CNN of Vicente et al. [24], Patched-CNN of Hosseinzadeh et al. [9], scGAN of Nyugen et al. [18], D-Net of Le et al. [14], ST-CGAN of Wang et al. [26] and DSC of Hu et al. [10].

5.1 Experimental Setup

All the running time measurements shown in this section were performed on a personal computer equipped with an NVIDIA GeForce GTX Titan X graphics card and an Intel Core^{TM} i7-3770K CPU (3.50 GHz), 8GB RAM. Processing time was evaluated only for the techniques whose source codes were gently provided by the authors of related work.

We have implemented our approach using OpenCV 2.3.1 [2] and CUDA 4.2 [13] with the NPP library[1]. Thrust library[2] was used to perform the vector compression shown in Figure 4-(d).

As discussed in Section 2, four datasets have already been proposed in the literature for shadow detection evaluation:

- **UCF dataset** [27] is a manually-labeled dataset, that contains a lot of shadow pixels who were not properly labeled in the ground-truth mask (Figure 5). Moreover, authors of related work diverge in the number of testing images that can be used in the UCF dataset (for instance, Zhu et al. [27] used 120 images for test, Shen et al. [21] used 245 images, Khan et al. [11,12] used 255 images, and Le et al. [14] used 111 images), making their accuracy estimates inconsistent;
- **UIUC dataset** [6] is a small-scale dataset labeled by checking the difference between shadow and non shadow images. Most of the images in the dataset contain shadows cast by a single object. Also, the dataset contains only 76 images for shadow detection evaluation. Moreover, as shown in Figure 6, this dataset also contains images with inaccurate shadow labeling [22,23]. That is why recent approaches are not using UIUC dataset for validation of their works [9,18,14];
- **SBU dataset** [24] is a large-scale dataset labeled using lazy annotation and contains 638 testing images. The dataset contains images that cover a large range of scenarios, such as beach, mountain, road, and snow;
- **ISTD dataset** [26] is a large-scale dataset that enables simultaneous shadow detection and removal. The dataset contains 540 testing images that vary mainly in terms of illumination and shadow shapes;

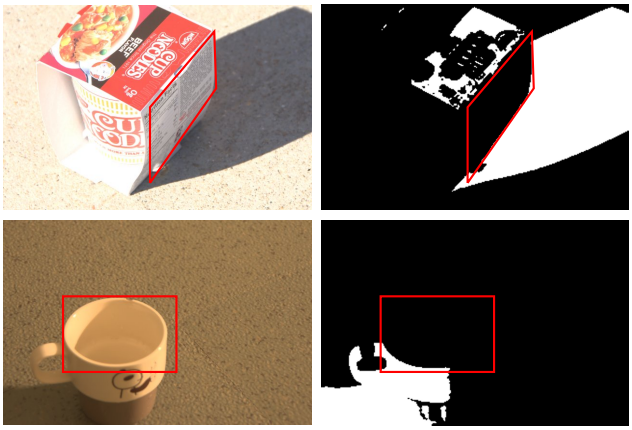
In this sense, we have chosen to evaluate our approach in the SBU and ISTD datasets, since they are the most recent datasets, provide more than 500 images for shadow detection evaluation, and contain less annotation errors than the other UCF and UIUC datasets.

To perform the quantitative evaluation, we compare the binary shadow masks generated by our approach with the ground-truth masks provided by the datasets. Accuracy is evaluated in terms of mean shadow pixel accuracy, mean non-shadow pixel accuracy, and Balance Error Rate (BER) metric

$$\text{BER} = 50\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right), \quad (3)$$

where TP, TN, FP and FN values represent the total number of true positive, true negative, false positive and false negative pixels. Natural images typically contain much more non-shadow pixels than shadow pixels. In this sense, the BER metric is less biased than the mean pixel accuracy metric, because it provides a more balanced evaluation between both shadow and non-shadow classes.

[1] https://developer.nvidia.com/npp
[2] https://thrust.github.io/



(a) Original Images (b) Incorrect Shadow Masks

Fig. 6 The regions inside the red rectangles show shadow pixels (a) that are not included in the ground-truth shadow masks (b). These images (a) are incorrectly labeled (b) in the UIUC dataset.

Table 1 The list of parameters used by our approach.

Parameter	Value
α_r	1.0
α_g	0.9
α_b	1.0
α_{gray}	0.9
α_{Cb}	0.9
α_{b*}	1.05
β	0.25
κ	16
ϕ	0.49
ω	0.6

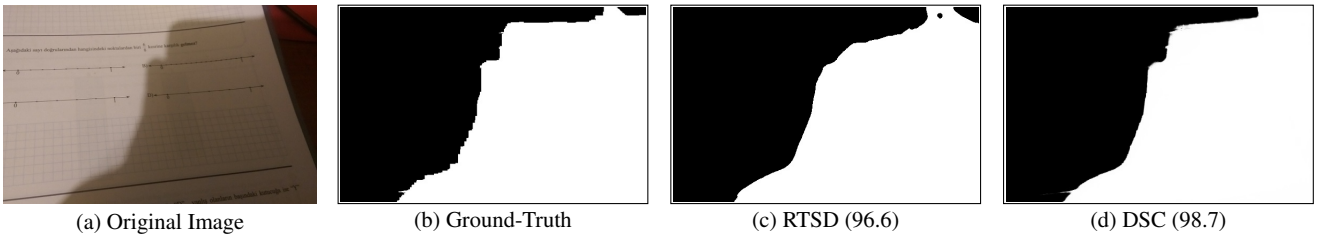
With respect to performance, we evaluate the processing time of the different techniques for conventional image resolutions typically found in augmented reality applications: $480p$, $720p$, $1080p$ and $2160p$.

In Table 1, we show the value of the parameters described in Section 3. These values are the ones that provided us the best results in the training sets of both SBU and ISTD datasets and were used by our algorithm in the test sets of these datasets.

Finally, in addition to the results shown in this section, we show the temporal coherence obtained by our approach in the supplementary video.

5.2 Accuracy Evaluation

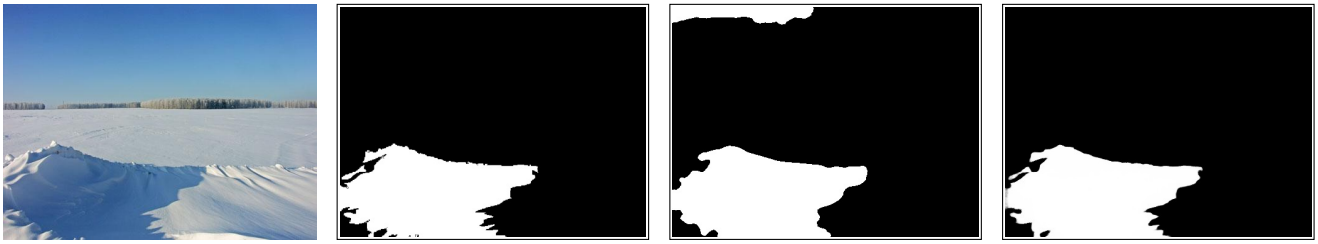
In this section, we evaluate the accuracy obtained by our shadow detection algorithm both quantitatively and qualitatively, as described in the next subsections.



(a) Original Image (b) Ground-Truth (c) RTSD (96.6) (d) DSC (98.7)

(e) D-Net (98.4) (f) scCGAN (84.1) (g) Stacked-CNN (95.6) (h) Patched-CNN (57.0)

Fig. 7 A visual comparison between our approach (RTSD) and different shadow detection techniques for an image with a shadow projected on a paper. Name and BER accuracy are displayed for each method. Images in columns (d), (f), (g) and (h) are courtesy of [10].



(a) Original Image (b) Ground-Truth (c) RTSD (96.2) (d) DSC (98.9)

(e) D-Net (99.2) (f) scCGAN (95.5) (g) Stacked-CNN (95.0) (h) Patched-CNN (84.2)

Fig. 8 A visual comparison between our approach (RTSD) and different shadow detection techniques for a snow scenario. Name and BER accuracy are displayed for each method. Images in columns (d), (f), (g) and (h) are courtesy of [10].

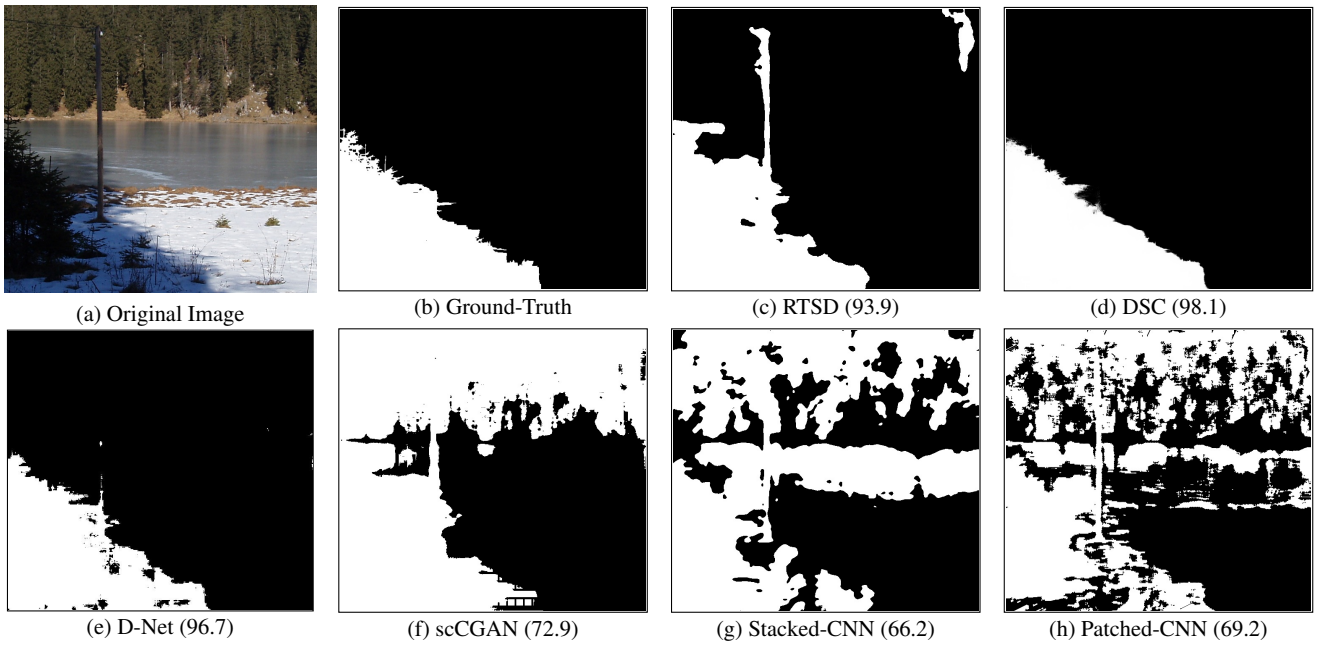
Table 2 Ranking of several shadow detection techniques on the SBU dataset. All methods are trained on the SBU training set, and evaluated on the SBU testing set. BER, mean shadow pixel, mean non-shadow pixel metrics are evaluated in terms of percentage of accuracy, where the higher the value is, the better the shadow detection result is. NRA - Not reported by the authors

Method	BER	Shadow	Non-Shadow
Unary-Pairwise [6]	86.0	NRA	NRA
Patched-CNN [9]	88.8	89.9	87.7
Stacked-CNN [24]	89.0	90.4	87.5
RTSD	**89.4**	**90.2**	**88.7**
scCGAN [18]	90.9	92.2	89.6
ST-CGAN [26]	91.9	96.3	87.5
D-Net [14]	94.3	93.8	94.8
DSC [10]	**94.4**	NRA	NRA

5.2.1 Quantitative Evaluation

In Table 2, we compare the accuracy obtained by the proposed approach with respect to related work on the SBU dataset. Our technique is more accurate than the previous techniques that used handcrafted features [6,9] and that one of the first techniques that used deep learning for shadow detection [24]. Moreover, our approach is only 5% less accurate than the state-of-the-art technique evaluated in this dataset [10].

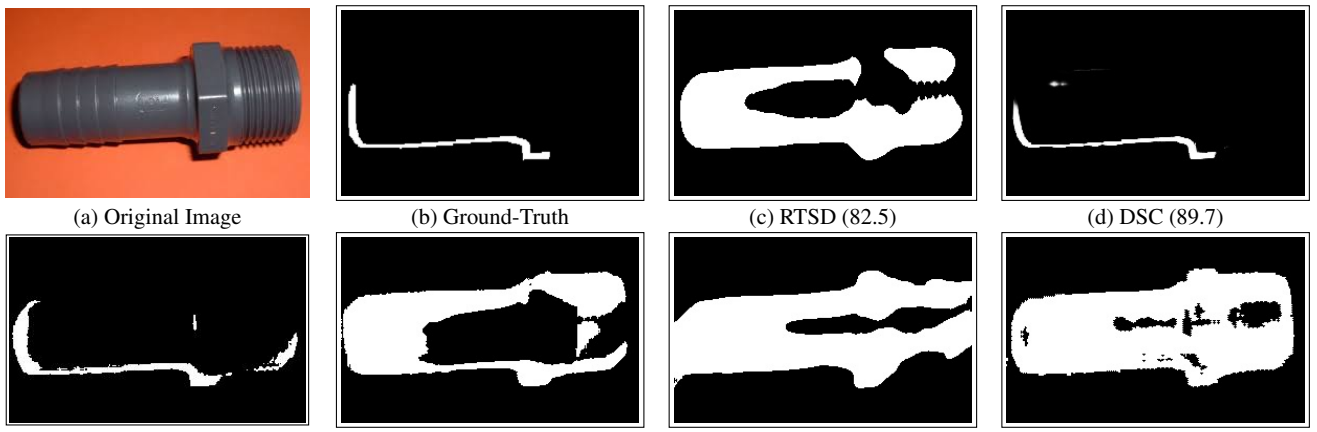
In Table 3, we compare the accuracy obtained by the proposed approach on the very recent ISTD dataset. As shown in Table 3, a few techniques have been evaluated in the ISTD dataset. Despite this fact, our approach is again more accurate than the Stacked-CNN technique [24], in terms of BER and non-shadow pixel metrics. Moreover, our approach is



(a) Original Image (b) Ground-Truth (c) RTSD (93.9) (d) DSC (98.1)

(e) D-Net (96.7) (f) scCGAN (72.9) (g) Stacked-CNN (66.2) (h) Patched-CNN (69.2)

Fig. 9 A visual comparison between our approach (RTSD) and different shadow detection techniques for a challenging scenario with trees and sand. Name and BER accuracy are displayed for each method. Images in columns (d), (f), (g) and (h) are courtesy of [10].



(a) Original Image (b) Ground-Truth (c) RTSD (82.5) (d) DSC (89.7)

(e) D-Net (96.6) (f) scCGAN (83.9) (g) Stacked-CNN (70.7) (h) Patched-CNN (73.4)

Fig. 10 A visual comparison between our approach (RTSD) and different shadow detection techniques for a challenging scenario with shadows projected by a gray object. Name and BER accuracy are displayed for each method. Images in columns (d), (f), (g) and (h) are courtesy of [10].

Table 3 Ranking of several shadow detection techniques on the ISTD dataset. All methods are trained on the ISTD training set, and evaluated on the ISTD testing set. BER, mean shadow pixel, mean non-shadow pixel metrics are evaluated in terms of percentage of accuracy, where the higher the value is, the better the shadow detection result is.

Method	BER	Shadow	Non-Shadow
Stacked-CNN [24]	91.4	92.0	90.8
RTSD	**91.5**	**89.8**	**93.2**
scCGAN [18]	95.3	96.8	93.8
ST-CGAN [26]	**96.2**	**97.9**	**94.5**

only 4.7% less accurate than the state-of-the-art technique on the ISTD dataset [26].

Table 4 The contribution of each step of the proposed approach in the final accuracy of the solution, measured by the BER metric for both SBU and ISTD datasets.

Step	BER (SBU)	BER (ISTD)
Gray-channel binarization	85.86	87.6
RGB-channel binarization	(+1.09) 86.95	(+0.09) 87.69
Cb-channel binarization	(+0.65) 87.6	(+0.04) 87.73
b*-channel binarization	(+0.4) 88.0	(+0.49) 88.22
Noise removal	(+1.41) **89.41**	(+3.25) **91.47**

In Table 4, we analyze the improvement, in terms of BER metric, provided by each one of the steps used in our real-time shadow detection algorithm. The mean binarization of the gray channel provides a good shadow detection

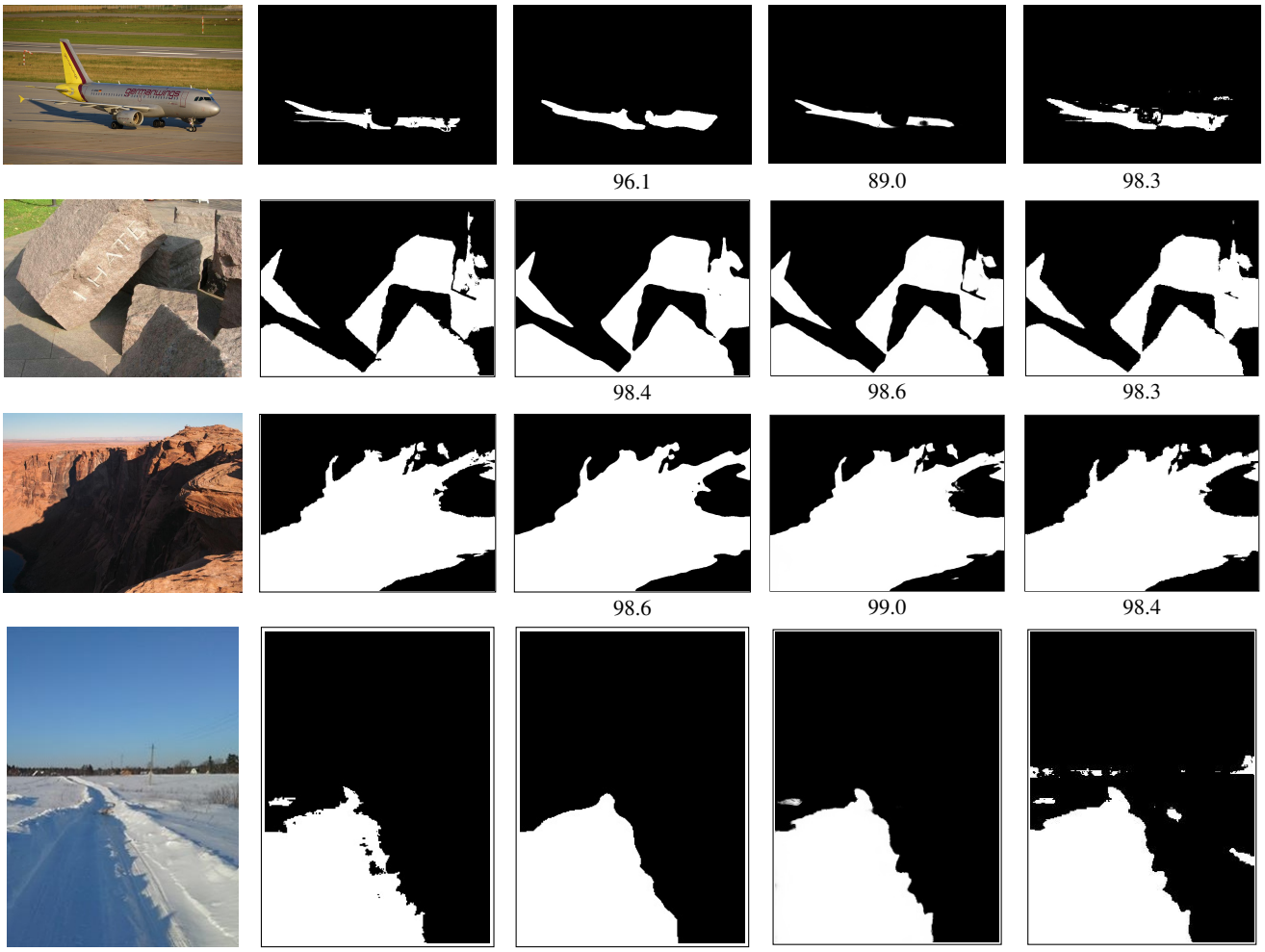


Fig. 11 A visual comparison between our approach (RTSD) and state-of-the-art shadow detection techniques for different scene conditions. BER accuracy is displayed below each image.

rate, being as accurate as the solution proposed by Guo et al. [6] (see BER metric for Unary-Pairwise technique in Table 2). The use of multi-channel binarization improves the accuracy of the shadow detection mostly for the SBU dataset, because this dataset contains more scenarios with blue skies, trees, and other regions whose one of the R, G, B, Cb and b* channels are too high. The ISTD dataset, otherwise, contains a lot of images captured with a typical street with gray floor. Our approach for noisy false-positive shadow region removal works well for both datasets, greatly reducing the noise artifacts generated by the multi-channel binarization and further improving the accuracy of the solution.

5.2.2 Qualitative Evaluation

In Figures 7, 8, 9, 10, we compare our approach with different shadow detection techniques proposed in the literature. For the scenarios shown in Figure 7 and 8, our approach (Figures 7-(c) and 8-(c)) is able to detect the shadows accurately, being competitive against most of related work (Figures 7-(d, e, f, g) and 8-(d, e, f, g)). Patched-CNN fails for both scenarios (Figures 7-(h) and 8-(h)). For the more complex scenario shown in Figure 9, with distinct objects (*e.g.,* trees, sand) present in the same scene, our approach fails to detect the light post as a non-shadow region (Figure 9-(c)). Despite this fact, our approach is still more accurate than some of the related work that incorrectly detect the trees as shadow regions (Figure 9-(f, g, h)). The major drawback of our approach is that it is not able to differentiate shadows from shadow casters, when the shadow casters are objects as dark as the shadows they cast in the scene. This fact is mainly visible in Figure 10, where the object that casts the shadow is mostly gray, slightly brighter than the shadow that it casts. Since the shadow region has a similar intensity than the shadow caster, our algorithm incorrectly assumes the shadow caster as shadow (Figure 10-(c)). We could not define a criteria able to easily handle this scenario, while keeping the real-time performance of the solution. On the other

Table 5 Processing time of each individual step of the proposed shadow detection technique for different image resolutions and processing units.

		Output Resolution (ms)			
Unit	**Step**	**480p**	**720p**	**1080p**	**2160p**
CPU	Color conversion	2.45	7.97	17.35	68.11
	Mean estimation	0.38	1.18	2.50	9.86
	Multi-channel bin.	1.41	4.08	9.43	35.37
	Noise removal	2.13	6.11	14.73	51.75
	Total	**6.37**	**19.34**	**44.01**	**165.09**
GPU	Color conversion	0.38	0.43	0.88	2.06
	Mean estimation	1.72	1.76	1.91	2.05
	Multi-channel bin.	1.00	1.10	1.16	1.99
	Noise removal	1.74	1.95	3.07	6.83
	Total	**4.84**	**5.24**	**7.02**	**12.93**

hand, the same problem is shared by most of related work (Figure 10-(f, g, h)), except for the state-of-the-art shadow detection techniques [14, 10], that are able to solve this problem accurately (Figure 10-(d, e)).

Aided by the results shown in Table 2 and visible in Figures 7, 8, 9, 10, we can further state that Stacked-CNN [24] and Patched-CNN [9] are less accurate than our approach because, even for simpler scenarios such as the one shown in Figure 8, they fail to detect non-shadow regions, overestimating the presence of shadows in the scene. This fact can be seen in Figures 8-(h) and 9-(g, h). The approach of sc-CGAN [18] is competitive against our approach, since we share the same limitation of detecting dark shadow casters as shadows, as shown in Figure 10-(f). Finally, D-Net [14] and DSC [10] are the state-of-the-art shadow detection techniques because their deep learning architectures were well designed to detect shadows even for challenging scenarios such as the one shown in Figure 10. Nevertheless, as visible in Table 2, our approach is only 5% less accurate than both techniques. Moreover, as we show in Figure 11, our approach (Figure 11-(c)) is able to perform shadow detection as accurately as the state-of-the-art techniques (Figure 11-(d, e)) for a variety of scene conditions. These include scenes with small (shadows cast by the airplane in Figure 11-top), large (desert-like region in Figure 11-middle) portions of shadows present in the scene and different weather conditions (snow in Figure 8 and 11-bottom). These results leverage the importance of the proposed algorithm, in the sense that we can achieve high accurate shadow detection results while keeping the real-time performance of the technique.

5.3 Processing Time Evaluation

In Table 5, we compare the processing time of our approach for each step of both CPU and GPU implementations for varying image resolutions. Our CPU implementation provides real-time performance for images with resolution up to $720p$. On the other hand, our GPU implementation provides real-time performance (more than 80 frames per second) even for a $2160p$ image resolution, being one order of magnitude, or, more exactly, 12 times faster than the CPU implementation for the same image resolution. So, our GPU implementation is more scalable than the CPU implementation for high image resolutions, providing real-time performance regardless of the image resolution used. In Table 5, we can also see that, in both processing units, noise removal is one of the slowest steps of our approach, because it requires a relative high number of memory accesses to perform the computations. On the other hand, as we can see in Table 4, this step is essential to improve the accuracy of the shadow detection.

Table 6 Ranking of the processing time (in seconds) obtained by several shadow detection techniques for images with different resolutions.

	Output Resolution (s)			
Method	**480p**	**720p**	**1080p**	**2160p**
Stacked-CNN [24]	69.91	78.11	375.68	484.70
Unary-Pairwise [6]	19.80	50.74	94.94	205.80
Patched-CNN [9]	1.37	1.43	6.12	7.53
D-Net [14]	0.032	0.096	0.21	0.87
RTSD - CPU	0.006	0.019	0.044	0.165
RTSD - GPU	**0.0048**	**0.0052**	**0.007**	**0.013**

In Table 6, we provide a comparison between the processing times obtained by our approach and related work. Stacked-CNN [24] is the slowest shadow detection technique because it uses a Fully Connected Network (FCN) to predict a shadow probability map from the original image, merges such a map with the original image, then divides both into overlapping patches of size 32×32, that are used as input for a CNN to predict local shadow pixels. Finally, the final shadow prediction of each pixel is given by a weighted average over the prediction outputs for the patches that contain each pixel. The use of a Fully Connected Network together with a CNN that runs for every overlapping 32×32 patch of the coupled original + shadow probability images makes the approach too costly, even for interactive applications, easily achieving more than 50 seconds for a low-sized image resolution of $480p$. Only for reference, our GPU-based real-time shadow detection that relies mostly on pixel independent operations is 14,564 times faster than the Stacked-CNN approach, being up to 4.6 orders of magnitude (37,284 times) faster than Stacked-CNN for a high-resolution $2160p$ image.

Unary-Pairwise [6] is faster than Stacked-CNN because, rather than running a CNN for every overlapping 32×32 patch of an image, the algorithm uses handcrafted features extracted from superpixels to predict a shadow probability map from an SVM. Then, another SVM is used to compare the shadow probabilities estimated by different regions of the image to ensure that regions with the same material are

classified in the same way. However, as shown in Tables 6 and 2, Unary-Pairwise is still too slow and inaccurate for interactive applications.

To optimize the processing time of Stacked-CNN, the Patched-CNN [9] merged the solutions proposed by Stacked-CNN and Unary-Pairwise techniques: replaced the FCN by the use of an SVM with handcrafted features. Also, rather than running the CNN for every pixel inside the 32×32 patches, the network is run on the basis of superpixels previously segmented. As shown in Table 6, the use of superpixels for CNN prediction greatly reduces the processing time demanded by the shadow detection. However, even in this case, the technique is still inadequate for real-time applications, demanding more than 1 second to perform shadow detection even for low-sized images.

D-Net [14] is faster than related work by the use of an adversarial shadow attenuation network for shadow detection. This network is pretty much smaller than the ones based on CNN because it requires only four modules with three layers each to provide accurate shadow detection results. D-Net technique is able to detect shadows in less than 1 second, but is ≈ 5 times slower than the CPU implementation of our algorithm, 6.6 times slower than our GPU implementation for a small image resolution (*e.g.,* $480p$), and 1.8 orders of magnitude (66 times) slower than our GPU implementation for a high image resolution (*e.g.,* $2160p$).

The DSC algorithm [10] extracts multi-scale features of an input image and fed these features to a spatial Recurrent Neural Network, that predicts shadow probability maps to produce the final shadow detection. Although we have not tested the DSC algorithm [10] for different image resolutions, it was reported by the authors that the algorithm takes 0.175 seconds to run their algorithm for an image with input size of 400×400 in an NVIDIA GeForce Titan X (Pascal) and an Intel Core i7. In this sense, their algorithm, running on a low-sized image, is slower than both CPU and GPU implementations of our approach when running at a $2160p$ image resolution (see the performance of RTSD in the last column of Table 6).

As can be seen in Table 6, by relying mostly on pixel-wise independent operations, only our GPU implementation consumes less than 15 milliseconds of processing time for the typical image resolutions evaluated. Hence, for high image resolutions, our approach provides a huge speedup over related work, being adequate for real-time applications that demand the shadow detection step to run as fast as possible.

6 Conclusion and Future Work

In this paper, we have presented a real-time algorithm for shadow detection that can be fully implemented in both CPU and GPU architectures. By converting the input colored image into different color spaces, we are able to perform a multi-channel binarization that gives a coarse estimate of where the shadow regions are located in the image. To further refine the shadow detection and minimize the presence of noisy, false-positive shadow regions previously detected, we employ an efficient noise removal algorithm that uses local and global strategies to remove small-sized incorrect shadow regions. We could show that, with our approach, we provide shadow detection as accurate as related work, but at a significantly less processing time. In this sense, we believe that our approach is ready to be used for robotic and augmented reality applications that require the shadow detection to run in real time.

In terms of accuracy, our approach is not able to differentiate dark objects from their shadows. For future work, one may investigate the use of alternative features to improve the shadow detection, while keeping the real-time performance of the solution. Also, our work could be further extended to support real-time shadow detection and shadow removal fully parallelized on the GPU.

Acknowledgements We are thankful to Guo et al. [6], Hosseinzadeh et al. [9] and Le et al. [14] for gently sharing the source code of their shadow detection algorithms. This research is supported by the scholarship program of Coordenação de Aperfeiçoamento de Pessoal do Nível Superior (CAPES). The hardware used for processing time evaluation was provided by NVIDIA Corporation, through the GPU Education Center. This is a post-peer-review, pre-copyedit version of an article published in Journal of Real-Time Image Processing. The final authenticated version is available online at: `http://dx.doi.org/10.1007/s11554-018-0799-3`

References

1. Al-Najdawi, N., Bez, H.E., Singhai, J., Edirisinghe, E.: A survey of cast shadow detection algorithms. Pattern Recognition Letters **33**(6), 752 – 764 (2012). DOI https://doi.org/10.1016/j.patrec.2011.12.013
2. Bradski, G., Kaehler, A.: Learning OpenCV: Computer Vision in C++ with the OpenCV Library, 2nd edn. O'Reilly Media, Inc. (2013)
3. Chen, J., Nonaka, K., Watanabe, R., Sankoh, H., Sabirin, H., Naito, S.: Efficient Parallel Connected Components Labeling with a Coarse-to-fine Strategy. In: arXiv preprint. arXiv:1712.09789 (2017)
4. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 27, pp. 2672–2680. Curran Associates, Inc. (2014)
5. Grana, C., Borghesani, D., Cucchiara, R.: Optimized block-based connected components labeling with decision trees. IEEE Transactions on Image Processing **19**(6), 1596–1609 (2010). DOI 10.1109/TIP.2010.2044963
6. Guo, R., Dai, Q., Hoiem, D.: Single-image shadow detection and removal using paired regions. In: Proceedings of the CVPR, pp. 2033–2040 (2011). DOI 10.1109/CVPR.2011.5995725
7. Guo, R., Dai, Q., Hoiem, D.: Paired Regions for Shadow Detection and Removal. IEEE Transactions on Pattern Analysis and

Machine Intelligence **35**(12), 2956–2967 (2013). DOI 10.1109/TPAMI.2012.214
8. Harris, M., Sengupta, S., Owens, J.D.: Parallel prefix sum (scan) with CUDA. GPU gems **3**(39), 851–876 (2007)
9. Hosseinzadeh, S., Shakeri, M., Zhang, H.: Fast Shadow Detection from a Single Image Using a Patched Convolutional Neural Network. In: arXiv preprint. arXiv:1709.09283 (2017)
10. Hu, X., Zhu, L., Fu, C.W., Qin, J., Heng, P.A.: Direction-aware spatial context features for shadow detection. In: Proceedings of the CVPR (2018)
11. Khan, S.H., Bennamoun, M., Sohel, F., Togneri, R.: Automatic Feature Learning for Robust Shadow Detection. In: Proceedings of the CVPR, pp. 1939–1946 (2014). DOI 10.1109/CVPR.2014.249
12. Khan, S.H., Bennamoun, M., Sohel, F., Togneri, R.: Automatic Shadow Detection and Removal from a Single Image. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(3), 431–446 (2016). DOI 10.1109/TPAMI.2015.2462355
13. Kirk, D.B., Hwu, W.m.W.: Programming Massively Parallel Processors: A Hands-on Approach, 2 edn. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2013)
14. Le, H., Vicente, T.F.Y., Nguyen, V., Hoai, M., Samaras, D.: A+D-Net: Shadow Detection with Adversarial Shadow Attenuation. In: arXiv preprint. arXiv:1712.01361 (2017)
15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015). DOI 10.1038/nature14539
16. Liu, Y., Granier, X.: Online Tracking of Outdoor Lighting Variations for Augmented Reality with Moving Cameras. IEEE Transactions on Visualization & Computer Graphics **18**, 573–580 (2012). DOI 10.1109/TVCG.2012.53
17. Mirza, M., Osindero, S.: Conditional generative adversarial nets. In: arXiv preprint. arXiv:1411.1784 (2014)
18. Nguyen, V., Vicente, T.F.Y., Zhao, M., Hoai, M., Samaras, D.: Shadow detection with conditional generative adversarial networks. In: Proceedings of the ICCV, pp. 4510–4518 (2017)
19. Podlozhnyuk, V.: Image convolution with CUDA. NVIDIA Corporation White Paper **2097**(3) (2007)
20. Sanin, A., Sanderson, C., Lovell, B.C.: Shadow detection: A survey and comparative evaluation of recent methods. Pattern Recognition **45**(4), 1684 – 1695 (2012). DOI https://doi.org/10.1016/j.patcog.2011.10.001
21. Shen, L., Chua, T.W., Leman, K.: Shadow Optimization From Structured Deep Edge Detection. In: Proceedings of the CVPR, pp. 2067–2074 (2015). DOI 10.1109/CVPR.2015.7298818
22. Vicente, T.F.Y., Hoai, M., Samaras, D.: Leave-One-Out Kernel Optimization for Shadow Detection. In: Proceedings of the ICCV, pp. 3388–3396 (2015). DOI 10.1109/ICCV.2015.387
23. Vicente, T.F.Y., Hoai, M., Samaras, D.: Leave-one-out Kernel Optimization for Shadow Detection and Removal. IEEE Transactions on Pattern Analysis and Machine Intelligence **PP**(99), 1–1 (2017). DOI 10.1109/TPAMI.2017.2691703
24. Vicente, T.F.Y., Hou, L., Yu, C.P., Hoai, M., Samaras, D.: Large-Scale Training of Shadow Detectors with Noisily-Annotated Shadow Examples. In: B. Leibe, J. Matas, N. Sebe, M. Welling (eds.) Proceedings of the ECCV, pp. 816–832. Springer International Publishing, Cham (2016)
25. Vicente, T.F.Y., Yu, C.P., Samaras, D.: Single Image Shadow Detection Using Multiple Cues in a Supermodular MRF. In: Proceedings of the BMVC. BMVA Press, Bristol UK (2013)
26. Wang, J., Li, X., Yang, J.: Stacked Conditional Generative Adversarial Networks for Jointly Learning Shadow Detection and Shadow Removal. In: Proceedings of the CVPR (2018)
27. Zhu, J., Samuel, K.G.G., Masood, S.Z., Tappen, M.F.: Learning to recognize shadows in monochromatic natural images. In: Proceedings of the CVPR, pp. 223–230 (2010). DOI 10.1109/CVPR.2010.5540209