aggregate cost values within the pixel neighborhoods defin by these windows. In 2005, Yoon and Kweon [4] proposed an adaptive matching cost aggregation scheme, which assigns a weight value to every pixel located in the support window of a given pixel of interest. The weight value is based on the spatial and color similarity between the pixel of interest and a pixel in its support window, and the aggregated cost is computed as a weighted average of the pixel-wise costs within the considered support window. The edge-preserving nature and matching accuracy of adaptive support weights have made them one of the most popular choices for cost aggregation in recently proposed stereo matching algorithms [3], [5]–[8].

Recently, Rheman *et al.* [9], [10] have revisited the co aggregation step of stereo algorithms, and demonstrated that cost aggregation can be performed by filtering of subsequent layers of the initially computed matching cost volume. In particular, the edge-aware image filters, such as the bilateral filter of Tomasi and Manducci [11] or the guided filter of He [12], have been rendered useful for the problem of matching cost aggregation, enabling stereo algorithms to correctly recover disparities along object boundaries. In fact, Yoon and Kweon's adaptive support-weight cost aggregation scheme is equivalent to the application of the so-called joint bilateral filter to the layers of the matching cost volume.

It has been demonstrated that the performance of ster algorithms designed to match a single pair of images can be adapted to take advantage of the temporal dependencies available in stereo video sequences. Early proposed solutions to temporal stereo matching attempted to average matching costs across subsequent frames of a video sequence [13], [14]. Attempts have been made to integrate estimation of motion fields (optical flow) into temporal stereo matching. The methods of [15] and [16] perform smoothing of disparities along motion vectors recovered from the video sequence. The estimation of the motion field, however, prevents real-time implementation, since state-of-the-art optical flow algorithms do not, in general, approach real-time frame rates. In a related approach, Sizintsev and Wildes [17], [18] used steerable filters to obtain descriptors characterizing motion of image features in both space and time. Unlike traditional algorithms, their method performs matching on spatio-temporal motion descriptors, rather than on pure pixel intensity values, which leads to improved temporal coherence of disparity maps at the cost of reduced accuracy at depth discontinuities.

Most recently, local stereo algorithms based on edge-awa filters were extended to incorporate temporal evidence into the matching process. The method of Richardt *et al.* [19] employs a variant of the bilateral grid [20] implemented on graphics hardware, which accelerates cost aggregation and allows for weighted propagation of pixel dissimilarity metrics from previous frames to the current one. Although this method outperforms the baseline frame-to-frame approach, the amount of hardware memory necessary to construct the bilateral grid limits its application to single-channel, i.e., grayscale images only. Hosni *et al.* [10], on the other hand, reformulated kernels of the guided image filter to operate on both spatial and temporal information, making it possible to process a tempor collection of cost volumes. The filtering operation was shown to preserve spatio-temporal edges present in the cost volumes, resulting in increased temporal consistency of disparity maps, greater robustness to image noise, and more accurate behavior around object boundaries.

III. Metho

The proposed temporal stereo matching algorithm is extension of the real-time iterative adaptive support-weight algorithm described in [3]. In addition to real-time two-pass aggregation of the cost values in the spatial domain, the proposed algorithm enhances stereo matching on video sequences by aggregating costs along the time dimension. The operation of the algorithm has been divided into four stages: 1) two-pass spatial cost aggregation, 2) temporal cost aggregation, 3) disparity selection and confidence assessment, and 4) iterative disparity refinement. In the following, each of these stages is described in detail.

A. Two-Pass Spatial Cost Aggregati

Humans group shapes by observing the geometric distan and color similarity of points in space. To mimic this visual grouping, the adaptive support-weight stereo matching algorithm [4] considers a support window Ω_p centered at the pixel of interest p, and assigns a support weight to each pixel $q \in \Omega_p$. The support weight relating pixels p and q is given by

$$w(p,q) = \exp\left(-\frac{\Delta_g(p,q)}{\gamma_g} - \frac{\Delta_c(p,q)}{\gamma_c}\right), \quad (1)$$

where $\Delta_g(p,q)$ is the geometric distance, $\Delta_c(p,q)$ is the c difference between pixels p and q, and the coefficients γ_g and γ_c regulate the strength of grouping by geometric distance and color similarity, respectively.

To identify a match for the pixel of interest p, the real-t iterative adaptive support-weight algorithm evaluates matching costs between p and every match candidate $\bar{p} \in S_p$, where S_p denotes a set of matching candidates associated with pixel p. For a pair of pixels p and $\bar{p}$, and their support windows Ω_p and $\Omega_{\bar{p}}$, the initial matching cost is aggregated using

$$C(p,\bar{p}) = \frac{\sum_{q\in\Omega_p, \bar{q}\in\Omega_{\bar{p}}} w(p,q)w(\bar{p},\bar{q})\delta(q,\bar{q})}{\sum_{q\in\Omega_p, \bar{q}\in\Omega_{\bar{p}}} w(p,q)w(\bar{p},\bar{q})}, \quad (2)$$

where the pixel dissimilarity metric $\delta(q,\bar{q})$ is chosen as sum of truncated absolute color differences between pixels q and $\bar{q}$. Here, the truncation of color difference for the red, green, and blue components given by

$$\delta(q,\bar{q}) = \sum_{c=\{r,g,b\}} \min(|q_c - \bar{q}_c|, \tau), \quad (3)$$

This limits each of their magnitudes to at most τ, which p vides additional robustness to outliers. Rather than evaluating Equation (2) directly, real-time algorithms often approximate