Detecting, segmenting and tracking unknown objects using multi-label MRF inference

Mårten Björkman, Niklas Bergström, Danica Kragic

*Computer Vision and Active Perception lab (CVAP)
School of Computer Science and Communication (CSC)
Royal Institute of Technology (KTH)
Stockholm, Sweden*

Abstract

This article presents a unified framework for detecting, segmenting and tracking unknown objects in everyday scenes, allowing for inspection of object hypotheses during interaction over time. A heterogeneous scene representation is proposed, with background regions modeled as a combinations of planar surfaces and uniform clutter, and foreground objects as 3D ellipsoids. Recent energy minimization methods based on loopy belief propagation, tree-reweighted message passing and graph cuts are studied for the purpose of multi-object segmentation and benchmarked in terms of segmentation quality, as well as computational speed and how easily methods can be adapted for parallel processing. One conclusion is that the choice of energy minimization method is less important than the way scenes are modeled. Proximities are more valuable for segmentation than similarity in colors, while the benefit of 3D information is limited. It is also shown through practical experiments that, with implementations on GPUs, multi-object segmentation and tracking using state-of-art MRF inference methods is feasible, despite the computational costs typically associated with such methods.

Keywords: Figure-ground segmentation, Active perception, MRF, Multi-object tracking, Object detection, GPU acceleration

1. Introduction

Objects play a central role in computer vision, and different fields are dedicated to *recognize* or *classify* objects in images, or *track* such objects over time. The former tasks assume models of objects or classes of objects to be learned from sets of training examples. Given a test image, extracted features are associated to the learned models, with the goal of deducing whether a particular object or class exists in the image or not. Object tracking is a general problem, but can be facilitated by taking advantage of similar models, if the tracked object is previously known. While much focus has been given to these fields during the past decades, less focus has been given to that of discovering unknown objects in scenes and tracking these over time. Such a problem assumes that hypotheses of what could constitute real physical objects are first generated from images and then modeled, so that tracking can be initiated. Once tracked the hypothesis of an object can then either be confirmed or rejected, given multiple observations in sequence.

Motivations for such an active approach include: 1) allowing for finding and modeling of objects outside currently learned classes, and 2) assisting classification and recognition problems by limiting the search space. The task is related to the fields of visual attention and segmentation. Computational models of visual attention can be used to find regions of interest in images [1], regions that potentially have some semantic meaning to the observer. With these models, however, objects are localized, but rarely segregated from the surrounding scene. Work in object segmentation [2, 3] on the other hand, aim to segregate foreground objects from their backgrounds. In most cases,

however, no real *concept* of an object exists. Instead, a user is needed to indicate the object in the image, by for instance framing it [3]. The presented work is more related to recent work by Mishra and Aloimonos [4], where the concept of an object is more central. In their work they segment “simple” objects defined as compact regions enclosed by edge pixel that arise either due to discontinuities in depth or from contact with supporting surfaces. In this work we define an object as something that occupies a portion of 3D space, and exhibits some continuity in appearance and shape. Compared to [4] our framework has three main advantages. First, it runs in real time, thus enabling tracking. Second, as a side effect of the object definition, it produces a model of an object, in terms of its size, rough shape and color distribution. Third, it allows for simultaneous segmentation and tracking of multiple objects, not just a single one.

A motivating goal of the presented work is a system that facilitates active scene and object understanding in realistic indoor settings [5, 6]. Given an observed scene containing unknown objects of interests, the system should allow for objects to be modeled and refined over sequences of observations, while the camera pose changes or objects are interacted with by e.g. a robotic manipulator. In such a scenario, segmentation and tracking serve little purpose in themselves, but are used as a means to extract attributes for object understanding over time and guide exploratory actions. The system should, however, be open also to other applications, such as semi-autonomous annotation of image and video data, an application that also requires precision and high speed. Care has thus been taken not to introduce assumptions specific to particular applications.

Tracking previously unseen objects has gained some attention lately [7, 8]. Similarly to our work, these methods create models of foregrounds and backgrounds, and segment objects based on measurements of e.g. image intensities and positions. Unlike our work, however, they use level sets for object tracking, and gain their speed from propagating only contours around objects, something that is possible even for multiple objects in real time, as demonstrated in [9]. However, they cannot effortlessly be used in unsupervised scenarios, as they require initial boundaries for initialization and are sensitive with respect to how these boundaries are drawn [10]. We instead take a graph based approach, similar to [4], which allows for more robust segmentation, while enabling unsupervised initialization. Our method performs tracking by modeling and segmenting each frame using not just boundary pixels, but every pixel, and letting model parameters evolve as functions of estimates from previous frames. This enables robustness to changes in object appearance, shape and topology. For additional robustness, in particular for cases when the boundary between an object and its supporting surface is ambiguous, we propose an heterogeneous scene representation that uses a combination of flat surfaces and random clutter for background modeling, with foreground objects modeled as 3D ellipsoids.

The main contributions of this work is a unified framework for principled *active object segmentation* by modeling the problem over a Markov Random Field (MRF) that

- allows for multi-object detection, modeling and tracking,
- considers all image pixels for classification, not just those around objects of interest, and
- has minimal requirements on user input for initialization.

In an effort not to sacrifice accuracy for speed, we present a thorough analysis of alternative MRF inference methods for segmentation. We study these in terms of both segmentation performance and computational costs, in particular when implemented on massively parallel GPU architectures. We also evaluate different tracking scenarios, where model parameters are predicted and tracked using sets of Kalman filters, and by doing so demonstrating the feasibility for tracking.

The outline of the presentation is as follows. In Sec. 2 we give an overview of the work related to our study. The theoretical basis for the tested MRF inference methods is given in Sec. 3. A heterogeneous scene representation for segmentation is presented in Sec. 4, with foreground objects modeled as 3D ellipsoids and background as a combination of planar surfaces and uniform clutter. In Sec. 5 the initialization procedure is described and in Sec. 6 the framework is extended for tracking, using Kalman filters for forward predictions of model parameters. A large series of off-line experiments, testing alternative MRF inference methods, are presented in Sec. 7, whereas in Sec. 8 these are studied from a computational point of view. Finally, the presentation is concluded in Sec. 9 with a discussion on future work.

2. Related work

2.1. Tracking and segmentation

There exists a vast amount of work on tracking of objects through sequences of images. Many of these deal with tracking of specific classes of objects, classes like human body parts [11, 12] and vehicles [13, 14]. Common for these is that they are all limited to one particular class, and are often optimized with respect to certain characteristics of this class. Model-based methods try to optimize some objective function on the current observation and a hypothesis created from the model. Other methods extract features from the appearance of an object and try to find a mapping to a database including a variety of poses of that object. In this work we aim for more generic solution that allows tracking of object of any kind, as long as these fit our definition of objects, which means that class specific methods are not applicable to our case.

Methods for figure-ground segmentation can be roughly divided into either variational or combinatorial methods. Variational methods assume that a functional is optimized over some continuous space. These can be expressed in terms of either object contours [15, 16] or image regions [17, 18]. Some methods can be adapted for specific object classes [19, 20, 21], but most are implemented as generic segmentation and tracking methods. The contour methods typically base their optimization on image gradients, which can be inherently hard to robustly extract. Our work is more related to the regions methods, where functionals are instead based on how well areas inside and outside an object contour fit some model of respective regions.

What information you use for optimization may vary, depending on use cases and need for robustness. Chan and Vese [18] model foregrounds and backgrounds with only mean intensities, whereas Freedman and Zhang [22] allow arbitrary distributions, but only for foreground regions, while Bibby and Reid [7] use color histograms for both foregrounds and backgrounds [7]. Similarly to our work, Chockalingam et al. [8] model both appearances, spatial positions and region extents. They do this in 2D, however, while the system proposed in this work exploits also 3D data, if such data is available.

Unlike the variational methods, the combinatorial ones typically define the segmentation problem as a labeling problem on a Markov Random Field (MRF), where each pixel is represented by a node that is connected to some local neighborhood. The segmentation is then based on minimizing an energy function (or maximizing a probability) over functions on individual nodes and their neighborhoods using methods like *iterated conditional modes* (ICM) [23], *graph cuts* [24, 25], *tree-reweighted message passing* [26, 27] or *loopy belief propagation* [28, 29]. Using graph cuts, global minima can be found for binary problems (one foreground and one background), but even for non-convex multi-label problems good approximations exist [27, 30].

There are relatively few comparisons made between alternative labeling methods, in particular for multi-object segmentation. Tappen and Freeman [31] compare loopy belief propagation to graph cuts for the purpose of stereo matching. Szeliski et al. [32] study most of the above mentioned methods for a

variety of problems, including interactive segmentation, stereo matching, image stitching and denoising. In their work, however, they focus on the methods' ability to minimize the energy, rather than the quality of the end results, which is not necessarily the same thing. In our work we look into the modeling problem of finding the most likely object models for image segmentation and tracking, and benchmark in terms of the quality of the final segmentation. This is done over time, which means that the energy function is in constant change, and so is the minimum energy.

A popular way of minimizing the energy functionals is formulating the problem with level-sets, introduced in [18]. In this case, a function is defined on the image region, and the object boundary is defined as the zero crossing of this function. The methods search the function that minimizes the energy by propagating this boundary, which is often done using a gradient descent approach. Herein lies two weaknesses with level set based methods; it is easy to get stuck in a local minima, and only small steps can be taken why convergence can be slow. As was shown by Grady et al. in [10], if the energy is defined over a graph instead, global methods like Graph-Cuts [33] can be used which gives a solution with a lower energy in most cases as well as finding it in orders of magnitude fewer iterations and less time. Furthermore, they demonstrate that Graph-Cut based minimization is less sensitive to difference in initialization, something that is desirable for active scene and object understanding. Although real time implementations of level set trackers have been proposed [9], the fact still remains that the solution might be a local minima. In this work we define the segmentation and tracking problem on a graph, reducing the risk of the segmentation getting stuck in a local minima.

2.2. Initialization and interaction

For initialization most figure-ground segmentation methods require some form of input from a human operator, a constraint that is critical in particular for autonomous application. Different approaches have been proposed, such as scribbling in the image [2, 34], framing the object of interest with a rectangle [3, 35], or just indicating object positions with single points [36, 37]. Scribbling allows you to directly build appearance models from the scribbled areas. In [38] users interact with an image by defining a rough, wide border around the foreground object and building a color model from the inside and outside of that border. In [2] the marked regions are used for two purposes: as hard constraints during the energy minimization, and as source pixels for intensity histograms that serve as foreground and background models. The user is further allowed to correct faulty segmentation by scribbling in misclassified areas for gradual improvements. Scribbles were also used in [34] to co-segment several images with the same foreground object. Like [2] an operator is allowed to correct for incorrect segmentation. However, rather than having the user manually search for possible improvements, the method identifies ambiguous areas and suggests where to draw new scribbles.

Both [34] and [39] stress the importance of quick user feedback in interactive scenarios. To speed up the following graph cut, these methods rely on an initial over-segmentation of the

image and create an MRF on the segmented regions, rather than on the individual pixels. This means, however, that the final segmentation becomes highly dependent on the quality of the over-segmentation and on how well it obeys the true object boundaries, something that is hard to guarantee in practice. If the over-segmentation is incorrect, there is no way to recover. In our work we thus avoid such over-segmentation, but are still able to reach speed allowing interactive scenarios, by a careful choice of methods based not just on accuracy, but also on how well they can be implemented on parallel hardware.

There are methods that require less input from the user, than those of scribbles. In [3, 35], an object is indicated by simply drawing a bounding rectangle around it. This puts other demands on the models, since the only thing known is that the exterior is part of the background – no hard constraints are provided for foreground pixels. Model parameters can therefore not be fixed like the case with scribbles, but have to be optimized along with the labeling, either using an iterative scheme [3] or jointly with the segmentation [35]. In this work we take the approach in [3] and iteratively segment and update model parameters, but do so without even having hard constraints for the background pixels, i.e., no single pixel is a-priori assigned to either foreground or background.

Even if framing an object is a quick method for providing user input, it still requires a mouse interface for interaction, which limits its applicability for autonomous systems. The simplest possible method for initialization is that of using just a single seed point. After all, the system needs to know which image region is the one to be regarded as foreground for any figure-ground segmentation to be initiated. Psychophysical studies on visual attention [40, 41] have long served as inspiration for computational models [42, 43], out of which many have been used in robotics for applications such as gaze control [44]. Despite differences in features used and level of biological relevance, these models all generate image points or regions that are conspicuous with respect to its surroundings, so as to direct computational resources towards regions that are more likely to be of importance to the overall system. There is no guarantee, however, that generated attention points are in fact located on objects of interest. Efforts have been made to bias attention models towards image points where physical objects are more likely to reside, using e.g. contrasts [1], contextual priming [45] or symmetries [46]. In this study we find seed points by detecting high density cluster centers in 3D image disparity space using a combination of Gaussian filtering and mean-shift. Even if this approach resembles the above mentioned attention models, it was not designed to be biologically constrained.

Being given only a single point puts additional requirements on the initialization of foreground models. Mishra and Aloimonos [36] create a log-polar image representation around a selected point and run the segmentation in two passes. The first pass only uses a measure of edgeness [47] for segmentation. In the second pass the segmentation is improved from color models obtained after the first pass. Despite the impressive results, the high computational cost of the edge detector makes it impractical for interactive scenarios. Similarly to Mishra and Aloimonos we apply stereo cues to assist initialization using

only single points as foreground seeds. However, while they use stereo to enhance discontinuity edges, we apply them to prune background points from the vicinity of seed points.

3. Multi-label segmentation

The segmentation framework used in this study contains a finite set of models, indexed by $\mathcal{L} = \{1, 2, \dots, K\}$, representing a number of foreground object regions and a background. Let $\mathcal{S} = \{1, 2, \dots, N\}$ be a set of pixel-wise indices and $\mathbf{m} = \{m_i; m_i \in \mathcal{M}, i \in \mathcal{S}\}$ a set of measurements, measurements that are observed values of a random field $M = \{M_i; i \in \mathcal{S}\}$, where $\mathcal{M}$ is the space of measurements. Also let $L = \{L_i; i \in \mathcal{S}\}$ be a random field of labels, with realizations $\mathbf{l} = \{l_i; l_i \in \mathcal{L}, i \in \mathcal{S}\}$ denoting from which particular model, among the finite set of models, the corresponding measurements are sampled from.

Assume that L is a *Markov random field* (MRF), which means that dependencies between labels are only local with respect to some neighborhood system $\mathcal{N}$, and that the state of L is unobservable. Further assume that the random field M is observable and that given a particular labeling $\mathbf{l}$, every measurement m_i follows a conditional probability distribution $p(m_i | l_i)$ of the same known analytic form $f(m_i; \theta_{l_i})$, where $\theta = \{\theta_l; l \in \mathcal{L}\}$ is the set of model parameters for the foreground object regions and background. Given that the random variables M_i are conditionally independent, that is

$$p(\mathbf{m} | \mathbf{l}) = \prod_{i \in \mathcal{S}} p(m_i | l_i), \quad (1)$$

M can be recognized as a *hidden Markov random field* (HMRF) [48]. The segmentation problem can be summarized as that of finding an estimate $\hat{\mathbf{l}}$ of the true labeling $\mathbf{l}^*$, both of which are realizations of the unobservable L , which is related to the observable M through conditional distributions, where M depends on an unknown parameter set θ .

Since both labels and model parameters are unknown and interdependent, the problem of recovering the labeling also involves that of estimating the parameters. Similar problems are often expressed as that of finding a joint maximum a posteriori (MAP) solution to $p(\theta, \mathbf{l} | \mathbf{m})$, often using graph cuts in an iterative framework [3, 35]. However, by doing so only one particular labeling contributes to the estimation of the model parameters, even if there might be many alternative labelings of almost equal probability. The estimated labeling might well be an artifact due to imperfect modeling and a poor representative of the distribution as a whole.

3.1. Optimization framework

Thus instead of searching for a joint MAP estimate of labels and parameters, one might seek a MAP estimate of the parameters θ through marginalization over all unknown labelings,

$$\hat{\theta} = \arg \max_{\theta} p(\theta | \mathbf{m}) = \arg \max_{\theta} \sum_{\mathbf{l} \in \mathcal{L}^N} p(\mathbf{m}, \mathbf{l} | \theta) p(\theta). \quad (2)$$

This can be done by first applying *expectation-maximization* (EM) for parameter estimation, as will be detailed below, and then find a labeling estimate in a second MAP step by

$$\hat{\mathbf{l}} = \arg \max_{\mathbf{l} \in \mathcal{L}^N} p(\mathbf{m}, \mathbf{l} | \hat{\theta}) = \arg \max_{\mathbf{l} \in \mathcal{L}^N} p(\mathbf{m} | \mathbf{l}, \hat{\theta}) p(\mathbf{l}). \quad (3)$$

By summing up the influence of all possible labelings in the estimation of the true parameter set θ^* , one explicitly takes the uncertainties in the data into consideration. If two different interpretations are of equal probability, they both contribute to $\hat{\theta}$ with equal weight. Thus in cases where precision and robustness are hard to simultaneously achieve, precision is traded for robustness. In that sense $\hat{\mathbf{l}}$ can be seen as the best representation of the full distribution of labels, instead of being one particular labeling that happens to maximize the joint probability distribution.

3.2. Expectation-maximization

For L we assume a neighborhood system $\mathcal{N}$ based on four neighbors. We further assume that the dependencies between labels $\mathbf{l}$ do not in turn depend on the parameters θ . This means the distribution of labels can be written as

$$p(\mathbf{l}) = \frac{1}{Z} \prod_{i \in \mathcal{S}} f(l_i) \prod_{j \in \mathcal{N}_i} f(l_i, l_j). \quad (4)$$

Here we have also assumed that the label priors $f(l_i)$ are also independent on θ and the same for all pixels. More details on these priors will be given later in Section 4. Due to the conditional independence of M_i in the HMRF, measurements are only dependent on their respective labels, which means that we have

$$\begin{aligned} p(\mathbf{m}, \mathbf{l} | \theta) &= p(\mathbf{m} | \mathbf{l}, \theta) p(\mathbf{l}) = \\ &= \frac{1}{Z} \prod_{i \in \mathcal{S}} p(m_i | l_i, \theta) f(l_i) \prod_{j \in \mathcal{N}_i} f(l_i, l_j). \end{aligned} \quad (5)$$

The first half on the right side of Eq. (5) represents the first order cliques of the MRF, whereas the second half represents the second order cliques. Like many others we apply a contrast sensitive Potts model [49, 50, 2, 51] for the inter-label dependencies $f(l_i, l_j)$.

The EM algorithm used to find the estimate $\hat{\theta}$ in Eq. (2) maximizes $\log p(\theta | \mathbf{m})$ with respect to θ , which is equivalent to a maximization of $p(\theta | \mathbf{m})$. This is done iteratively by updating a lower bound $G(\theta) = Q(\theta | \theta') + \log p(\theta) \leq \log p(\mathbf{m} | \theta) + \log p(\theta)$, where

$$Q(\theta | \theta') = \sum_{\mathbf{l} \in \mathcal{L}^N} p(\mathbf{l} | \mathbf{m}, \theta') \log p(\mathbf{m}, \mathbf{l} | \theta) \quad (6)$$

and θ' is the parameter estimate from the previous iteration. Given an initial parameter estimate θ^0 , the algorithm alternates between two steps until convergence. In the first step, the *expectation* step, the expectation $\mathcal{E}[\log p(\mathbf{m}, \mathbf{l} | \theta)]$ is evaluated over the conditional distribution of labels $p(\mathbf{l} | \mathbf{m}, \theta')$ given the measurements $\mathbf{m}$ and the current parameter estimate θ' . In the second step, the *maximization* step, a new parameter set

$$\theta^{new} = \arg \max_{\theta} G(\theta) \quad (7)$$

is sought as the set that maximizes the current lower bound. Note that unlike typical maximum likelihood formulations of EM, a prior $\log p(\theta)$ is added to the lower bound $G(\theta)$. This prior can be used to include dependencies on object specific knowledge or historical estimates.

Unfortunately, since there are K^N possible labelings in total, the integration over labelings in Eq. (6) becomes computationally infeasible. However, if gradient descent is applied to maximize the lower bound $G(\theta)$ according to Eq. (7), the bound does not have to be explicitly computed, only its gradient with respect to θ . Given that the $p(\mathbf{l})$ in Eq. (5) is assumed to be independent on θ , its terms disappear when the derivative of $G(\theta)$ is computed. Thus the objective function $Q(\theta | \theta')$ can be replaced by

$$Q_{\text{mrg}}(\theta | \theta') = \sum_{\mathbf{l} \in \mathcal{L}^N} (p(\mathbf{l} | \mathbf{m}, \theta') \sum_{i \in \mathcal{S}} \log p(m_i | l_i, \theta)) = \quad (8)$$

$$= \sum_{i \in \mathcal{S}} \sum_{l_i \in \mathcal{L}} \log p(m_i | l_i, \theta) \sum_{\mathbf{l} \setminus \{l_i\} \in \mathcal{L}^{N-1}} p(\mathbf{l} | \mathbf{m}, \theta') = \quad (9)$$

$$= \sum_{i \in \mathcal{S}} \sum_{l_i \in \mathcal{L}} \log p(m_i | l_i, \theta) p(l_i | \mathbf{m}, \theta') \quad (10)$$

The second equivalence is due to the fact that the terms in the Eq. (8) involve only one label each. The order of summations can thus be changed, so that an inner summation is done over all labels but l_i , while an outer summation is done over l_i . This inner-most summation can be identified as the distribution $p(l_i | \mathbf{m}, \theta')$ with the remaining labels marginalized out, as shown in Eq. (10).

Since $Q_{\text{mrg}}(\theta | \theta')$ only involves KN terms, instead of K^N , the summation is computationally tractable. Note that the only assumption added to those of the HMRF is that higher-order cliques need to be independent on the parameters θ to be estimated in the EM loop. In [52], where parameters controlling the dependency between labels were also estimated by EM, an approximation based on a MAP estimation of labels computed in the previous update had to be applied in order to reach similar tractability.

3.2.1. Sum Product Belief Propagation (LBP-S)

The marginal distributions $p(l_i | \mathbf{m}, \theta')$ in Eq. (10) can be estimated with *loopy belief propagation* using sum-product [28, 29], a message passing method that will be referred to as LBP-S in the experiments below. A message $m_{i \rightarrow j}(l_j)$ can be interpreted as the influence a node i has on a neighboring node j . All messages are initialized to 1 and then iteratively passed in parallel in both direction using the update rule

$$m_{i \rightarrow j}(l_j) \leftarrow \sum_{l_i} p(m_i | l_i, \theta) f_i(l_i) f(l_i, l_j) \prod_{k \in N_i \setminus j} m_{k \rightarrow i}(l_i). \quad (11)$$

After convergence the belief

$$b_i(l_i) = p(m_i | l_i, \theta) f_i(l_i) \prod_{k \in N_i} m_{k \rightarrow i}(l_i) \quad (12)$$

can be computed, which in case of graphs without loops can be shown using induction to be the exact marginal distribution $p(l_i | \mathbf{m}, \theta')$. Even if the procedure is not guaranteed to converge for graphs with loops, it usually does [53, 29].

3.3. MAP based segmentation

With marginalization, as described above, no hard decisions are imposed on the labeling until after EM convergence, since the modeling is done through a marginalization over all labelings. This might be attractive, if the modeling is more important than the segmentation, and you wish uncertainties in the segmentation to be reflected in the models, instead of letting the models rely on the particular labeling that maximizes the joint probability. However, if the opposite is true and the most probable segmentation is the end goal, such an approach might not be ideal.

An alternative to marginalization is to instead use maximization for the labels in the *expectation* step of the EM algorithm. By exploiting the local characteristics of MRFs we can write the distribution of labelings in Eq. (9) as

$$p(\mathbf{l} | \mathbf{m}, \theta') = \frac{p(\mathbf{m} | \mathbf{l}, \theta') p(\mathbf{l})}{p(\mathbf{m})} = \prod_{i \in \mathcal{S}} p(l_i | l_{N_i}, m_i, \theta'), \quad (13)$$

where

$$p(l_i | l_{N_i}, m_i, \theta') = \frac{p(m_i | l_i, \theta') p(l_i | l_{N_i})}{p(m_i)} \quad (14)$$

and l_{N_i} represents the neighboring labels of l_i . Zhang et al. [48] proposed a method, called HMRF-EM, that first computes a MAP estimate of the labeling, using the current estimate of the model parameters θ' and then uses the conditional probabilities $p(l_i | l_{N_i})$, where l_{N_i} is assumed fixed by the MAP estimate. In practice this can be done by replacing the marginal distributions $p(l_i | \mathbf{m}, \theta')$ by the conditionals $p(l_i | l_{N_i}, m_i, \theta')$ in Eq. (10), leading to an objective function

$$Q_{\text{max}}(\theta | \theta') = \sum_{i \in \mathcal{S}} \sum_{l_i \in \mathcal{L}} \log p(m_i | l_i, \theta) p(l_i | l_{N_i}, m_i, \theta') \quad (15)$$

that is maximized in each EM iteration, given the previous estimate of the parameter set θ' .

In order to find a MAP estimate, we will explore four different energy minimization methods. These are all based on an energy formulation that can be written as the negative logarithm of the joint probability $p(\mathbf{m}, \mathbf{l} | \theta)$ in Eq. (5), that is

$$E(\mathbf{l}) = \sum_{i \in \mathcal{S}} \psi_i(l_i) + \sum_{i \in \mathcal{S}} \sum_{j \in N_i} \psi(l_i, l_j), \quad (16)$$

where $\psi_i(l_i) = -\log(p(m_i | l_i, \theta) f_i(l_i))$ and $\psi(l_i, l_j) = -\log f(l_i, l_j)$ are the negative logarithms of respective first and second order clique functions. The problem of minimizing the energy $E(\mathbf{l})$ is equivalent to that of maximizing $p(\mathbf{m}, \mathbf{l} | \theta)$, with respect to the labeling $\mathbf{l}$.

3.3.1. Iterated Conditional Modes (ICM)

The first maximization method to be tested, Iterated Conditional Modes [54], is a very simply and fast approach that operates locally by updating each individual label, based on the current estimate of its neighbors. Labels are initiated to those that maximizes the respective data terms $\psi_i(l_i)$. In each subsequent iteration each label is updated by maximizing $\psi_i(l_i) + \sum_{j \in N_i} \psi(l_i, l_j)$, where l_j are the neighboring labels from

the previous iteration. It will later be shown in the experimental section that due to its local nature ICM quickly falls into a local minimum, often with an energy higher than those reached by other methods. However, it still serves the purpose as an easily implemented baseline method.

3.3.2. Graph Cuts (GC)

Graph cuts have been successfully used for a large range of labeling problems, such as image restoration [55], stereo matching [56, 25] and segmentation [2]. For labeling problems with only two labels, maximum-flow methods can provide exact solutions [55]. For problems with more than two labels, the (α, β) -swap and α -expansion algorithms provide approximate solutions [25], if an edge cost, i.e. a second order clique function $\psi(l_i, l_j)$, constitute a metric. This is true in the case for the contrast sensitive Potts model that we use. Even if graph cut based methods have the weakness of not allowing arbitrary cost functions, they have the strength of permitting earlier results to be exploited for faster convergence. This is particularly beneficial for sequences, where results from previous updates can be reused.

The α -expansion algorithm starts from an arbitrary labeling, which is conveniently given by minimizing the unitary cliques $\psi(l_i)$ at each point. Then the method cycles through every label α and tests whether the current label should be changed to α . Thus at each stage there are two possible hypotheses per image point, either keep the current label or change the label to α . Using a maximum-flow method (such as *push-relabel* [57]) this binary labeling problem is found at each stage. Since energy is guaranteed never to increase, the overall method converges. For the later experiments we use Fast-PD [30], a primal-dual alternative to α -expansion, that improves speed by exploiting information from the dual problem to limit the size of graphs in later iterations.

3.3.3. Max Product Belief Propagation (LBP-M)

The message passing method max-product [58] is a modification of sum-product in which the summation in Eq. (11) is replaced by a maximization. The computed beliefs approximate the max-marginals $\max_{\mathbf{l}} p(\mathbf{l} \mid l_i, \mathbf{m}, \theta')$, instead of the marginal distributions $p(l_i \mid \mathbf{m}, \theta')$ in Eq. (10). For graphs without loops the max-marginals can be computed exactly, using the fact that any distribution of such a graph can be factorized in terms of max-marginals [59]. By maximizing the distribution for each label, a MAP solution to the labeling problem given the parameters θ' can be obtained and used in Eq. (15). With the energy formulation of Eq. (16), we get the message update function and beliefs

$$m_{i \rightarrow j}^{t+1}(l_j) = \min_{l_i} (\psi_i(l_i) + \psi(l_i, l_j) + \sum_{k \in N_i \setminus j} m_{k \rightarrow i}^t(l_i)) \quad (17)$$

and

$$b_i(l_i) = \psi_i(l_i) + \sum_{k \in N_i} m_{k \rightarrow i}(l_i). \quad (18)$$

In this formulation the maximization is changed to a minimization, which motivates min-sum being used as an alternative

name to sum-product. Even if convergence can only be guaranteed for graphs without loops, similarly to sum-product, max-product has been successfully used for applications such as decoding of turbo codes [60], super-resolution, shading and reflectance estimation [53], photo-montages and stereo matching [32].

For the experiments in Sec. 7, we used messages represented as integers and adopted modifications suggested by Felzenszwalb and Huttenlocher [61], to speed-up computations for both LBP-S and LBP-M. Using the fact the Potts model [49] only considers the equality and inequality of neighboring labels, Eq. (17) can be rewritten into two stages making the computational cost linear to the number of states, rather than quadratic. Memory requirements were also reduced by observing that with 4-connected labels, the graph is bipartite, which means that the two disjoint label sets can be updated in sequence, without additional temporary storage.

3.3.4. Sequential Tree-Reweighted Message Passing (TRW-S)

Unlike tree structures, there is no guarantee that max-product will converge to the true max-marginals and a labeling of lowest energy for graphs with loops. Tree-reweighted max-product methods try to overcome this by representing the graph as a convex combination of trees, with the goal of finding a labeling that is simultaneously optimal with respect to each such tree [26]. This is typically done by solving the dual problem of maximizing the lower bound of the energy, using a linear programming relaxation of the problem.

Given that there is a finite set of possible labels, the energy formulation in Eq. (16) can be rewritten as a linear combination of a set of node and edge parameters $\psi = \{\psi_{s,i}, \psi_{st,ij}\}$, and indicator functions

$$E(\mathbf{l}) = \sum_{s \in S} \sum_{i \in \mathcal{L}} (\psi_{s,i} \delta_i(l_s) + \sum_{t \in N_s} \sum_{j \in \mathcal{L}} \psi_{st,ij} \delta_i(l_s) \delta_j(l_t)). \quad (19)$$

Here the indicator function $\delta_i(l_s)$ is equal to 1 if $l_s = i$ and 0 otherwise. Since the representation above is overcomplete, a node parameter can be changed by updating the connected edge parameters accordingly, without affecting the energy function. This process is called reparametrization. The message updates in max-product can be shown to be such a reparametrization, with parameters converging towards the max-marginals for trees [26]. With tree-reweighted message passing, similar updates are done for each tree and in a second operation the parameters are averaged, forcing the parameters of each tree to the same limit point.

In this study we have used the sequential tree-reweighted algorithm for this, since this algorithm guarantees the lower bound never to decrease during averaging [27]. Trees are created in terms of monotonic chains, with respect to some ordering of nodes, which is fortunately trivial for regular structures like images, with only one chain covering each edge. Doing so the update function becomes very efficient,

$$m_{i \rightarrow j}^{t+1}(l_j) = \min_{l_i} (\frac{1}{n_i} b_i(l_i) + \psi(l_i, l_j) - m_{j \rightarrow i}^t(l_i)) \quad (20)$$

where

$$b_i(l_i) = \psi_i(l_i) + \sum_{k \in N_i} m_{k \rightarrow i}^t(l_i) \quad (21)$$

and n_i is the number of chains passing node i . Messages are iteratively passed between nodes, first according to the ordering of nodes and then in the opposite direction. Similarly to LBP-M, a locally optimal MAP solution to the labeling problem is then given by maximizing the beliefs $b_i(l_i)$ after convergence, a solution that is used in Eq. (15) for updating the model parameters.

4. Scene part modeling

For scene modeling we propose a heterogeneous scene representation comprised of 3D ellipsoids, planar surfaces and a uniform clutter model. The ellipsoids represent physical foreground objects in 3D space and are each modeled by a parameter set θ_f . Backgrounds are assumed to contain combinations of bounding planes and uniform clutter, modeled by parameters θ_p and θ_c respectively. A bounding plane is defined as a plane that limits the extent of the scene, such as walls, floors or tabletop surfaces. All modeled foreground objects are assumed to be placed on the sides of such planes facing the camera. Additional details on the definition of the complete parameter set $\theta = \theta_f \cup \theta_p \cup \theta_c$ will be given below. For brevity the notations assume that there is only one foreground object and a single bounding plane.

In the presented implementation the measurements at pixel i , $m_i = (p_i, c_i)$, consists of two components, a spatial components and a color component. The spatial component $p_i = (x_i, y_i, d_i)$ is given by image point positions (x_i, y_i) and binocular disparities d_i . It is assumed that disparities can be undefined, i.e. accurate disparities do not have to exist for every individual image point. In lack of disparities, inference will be based on (x_i, y_i) only. The color component $c_i = (h_i, s_i, v_i)$ is given in HSV space, where the three values denote hue, saturation and luminance respectively.

A foreground scene part is represented by a 3D ellipsoid and assumed to be normally distributed in image-disparity space, $P(p_i | l_i = l_f, \theta) = n(p_i; p_f, \Delta_f)$. Here $n(x; \bar{x}, \Delta)$ denotes a normal distribution with mean $\bar{x}$ and covariance Δ . The disparities of the planar background parts are modeled as a normal distribution, with the mean linearly dependent on the image coordinates, $P(d_i | l_i = l_p, \theta) = n(d_i; a_p x_i + b_p y_i + d_p, \Delta_p)$. Such a plane can be shown to correspond to a planar surface also in metric 3D space. The plane, however, has some 'thickness' given by Δ_p , which can be arbitrarily large. For the clutter part the disparities are modeled as a normal distribution, $P(d_i | l_i = l_c, \theta) = n(d_i; d_c, \Delta_c)$. Finally, image point positions are assumed to be uniformly distributed for both planar and clutter parts, $P(x_i, y_i | l_i = l_p, \theta) = P(x_i, y_i | l_i = l_c, \theta) = 1/N$, where N is the total number of points in image space.

The color distribution of each scene part is represent by normalized 2D histograms of hue and saturation; $p(h_i, s_i | l_i = l_f, \theta) = H_f(h_i, s_i)$, $p(h_i, s_i | l_i = l_p, \theta) = H_p(h_i, s_i)$ and $p(h_i, s_i | l_i = l_c, \theta) = H_c(h_i, s_i)$. With these histograms stacked into vectors c_f , c_p and c_c , the complete set of model parameters

is given by

$$\begin{aligned}\theta_f &= \{p_f, \Delta_f, c_f\}, \\ \theta_p &= \{a_p, b_p, d_p, \Delta_p, c_p\}, \\ \theta_c &= \{d_c, \Delta_c, c_c\}.\end{aligned}\tag{22}$$

These parameters are iteratively estimated using the MRF inference methods described in Section 3 above, where the unitary terms $p(m_i | l_i, \theta)$ $f(l_i)$ are given by the label priors $f(l_i)$ and the measurement conditionals that can be summarized as

$$\begin{aligned}P(m_i | l_i = l_f, \theta) &= n(p_i; p_f, \Delta_f) H_f(h_i, s_i), \\ P(m_i | l_i = l_p, \theta) &= N^{-1} n(d_i; a_p x_i + b_p y_i + d_p, \Delta_p) H_p(h_i, s_i), \\ P(m_i | l_i = l_c, \theta) &= N^{-1} n(d_i; d_c, \Delta_c) H_c(h_i, s_i).\end{aligned}$$

For single images the prior distribution $f(l_i)$ is assumed uniform. However, in the case of tracking they are changed based on the areas of segments from the previous update. For inter-label dependencies we use a contrast sensitive Potts model [49, 50]

$$f(l_i, l_j) = \exp(-\gamma[l_i \neq l_j] \exp(-\frac{(v_i - v_j)^2}{2\sigma_v^2})), \tag{23}$$

where $[\phi]$ is an indicator function equal to 1 if ϕ is true and 0 otherwise, $\{v_i; i \in S\}$ the image intensities for the corresponding labels, σ_v^2 the variance of intensity gradients between neighbors across the whole image and γ is a constant typically set to about 50.

5. Initialization

Initialization of scene parts relies on two sources of information provided either by a human operator or some other system; 1) the number of foreground parts to be instantiated and, 2) the expected 3D size of these parts. There is no requirement on either framing of parts or scribbling within parts, both of which would require a human operator and a mouse interface. The intention has been to keep the required information limited, making the applicability of the system as wide as possible. In an earlier version of the system [62], single foreground parts were always expected to be found in the center of view. This was possible by letting an attention mechanism control the camera system placing the detected regions of interest in the center after a view change. For multi-object scenarios this is no longer convenient and we thus use a different approach.

The initialization begins with all image points assigned to the clutter part that is represented by θ_c . From these clutter points, planar surfaces are hypothesized through random sampling and linear fitting in (x, y, d) -space. For those plane hypotheses that have a high enough number of supporting 3D points, the total number of points on either side of corresponding plane is counted. If the vast majority of points are on the side facing the camera system, which means that the plane is placed along the boundaries of the visible scene, this plane is kept, while those planes that cut through the scene are disregarded. In indoor settings there are rarely more than three such bounding planes, typically corresponding to walls, floors, ceilings and table tops.

Planes are then selected in a greedy manner, in the order of number of supported 3D points. For each selected plane a scene part is instantiated and represented by a θ_p set, with the corresponding clutter points reassigned to this new part. The process is repeated, until no more bounding planes can be found.

5.1. Foreground insertion

Once planar scene parts have been initiated, only clutter points located inside the boundaries of the observed scene remain. Foreground object parts can be inserted in many different ways, e.g., by a human operator or through some external trigger mechanism. For the later experiments, however, an automatic procedure was used instead. The remaining points are first sliced up into a number of overlapping intervals in depth, each represented by a binary mask of those points that exist in each slice [63]. Each slice is then blurred by a 2D Gaussian kernel that has a projective size given by the expected 3D size and the mean depth of the slice. The resulting maxima after blurring of slices serve as seed points for a mean-shift operation [64] that iteratively searches for high density cluster centers. The kernel used for mean-shift is similar to the one used for blurring, but applied to the 3D cloud of clutter points, rather than in 2D within each slice. After mean-shift has converged, centers are greedily selected based on the highest number of supporting 3D points. Since points located close to the camera are more densely distributed, there is thus a natural bias towards nearby parts, such that these are selected first.

Foreground scene parts are finally instantiated, and represented by a θ_f set each, with the selected cluster centers as their initial positions p_f . A small 3D ball is placed around each center and points located within such a ball are reassigned to the corresponding part. Like the kernels used for blurring and mean-shift, the size of these balls is also given by the expected 3D size. However, to prevent nearby scene parts from affecting the initialization, the ball size is set to a smaller fraction of the expected size. Even if the resulting initial segments do not cover the full parts, they will grow once the EM loop is started. In the beginning, the disparity cue tends to be dominating, while the color cue is able to fill the gaps for those points that lack disparities. Some experiments are given in Sec 7, illustrating the trade-off between having large balls, to cover most of the target scene parts, and small balls, to prevent neighboring parts from interfering.

6. Integration over time

To extend the scene part modeling and segmentation framework to tracking, we apply temporal filters to predict the model parameters between subsequent image frames. Thus instead of directly using the results from a previous frame to initialize the EM iteration of the next, we perform the initialization using a set of predicted parameters. These parameters are also used for queries by external processes, such as the motor controller of a robotic head system. Since information survives over time, only a few EM iterations are necessary for each new frame, which allows for segmented objects to be tracked in real-time, using an

implementation such as the one described in Sec. 8. Instead of a typical 20 iterations to reach convergence in off-line use, 3 iterations are usually enough in on-line use to reach acceptable results.

6.1. Intrinsic parameters

We differentiate between model parameters that are considered as *intrinsic* or *extrinsic*. Intrinsic parameters are those that are assumed constant over time, whereas extrinsic parameters change as objects and cameras are moving from one frame to the next. Among the parameters in Eq. (22) we regard the covariance matrix of foreground points Δ_f , the 'thickness' of planar surfaces Δ_p , the background clutter parameters d_c and Δ_c , and the three color models c_f , c_p and c_c , as intrinsic parameters. All these are filtered element-wise by an exponentially decaying kernel, which in effect is the same as Kalman filtering where the filtered parameter is assumed constant over time. For the experiments in Sec. 6.3 we use a kernel that has decayed to 50% of its maximum value after 10 frames. Temporal filtering is particularly efficient for the color models, since it allows sampling over time and prevents sparsity in the population of histogram bins.

6.2. Extrinsic parameters

Among the extrinsic parameters we count the center of gravity of foreground objects in 3D image-disparity space (x_f, y_f, d_f) and the parameters of the planar surfaces (a_p, b_p, d_p) , parameters that relate disparities to image coordinates according to $d(x, y) = a_p x + b_p y + d_p$. When cameras or objects are moving, these parameters change. For integration of information over time we apply extended Kalman filters (EKF). Such filters have two stages, a *predict* stage and an *update* stage, that can be summarized in two equations.

$$\begin{aligned}\mathbf{x}^k &= f(\mathbf{x}^{k-1}, \mathbf{u}^{k-1}) + \mathbf{w}^{k-1} \\ \mathbf{z}^k &= h(\mathbf{x}^k) + \mathbf{v}^k\end{aligned}$$

Here $\mathbf{x}^k$ denotes the state of a process governed by a transition function $f(\mathbf{x}^k, \mathbf{u}^k)$, where $\mathbf{u}^k$ is an external control input and $\mathbf{z}^k$ is the measurement at frame k . In our case the observation function $h(\mathbf{x}^k)$ is always assumed to be an identity function with respect to the parameters of the model in question. The random variables $\mathbf{w}^k$ and $\mathbf{v}^k$ represent the process and measurement noise, and are assumed to be zero mean and normal distributed, with covariances $\mathbf{W}$ and $\mathbf{V}$ respectively.

As illustrated in Fig. 1, the current state $\mathbf{x}^k$ after the predict stage is used as the initial parameters θ_{pred} of the segmentation system, while the resulting model parameters θ after segmentation are used as measurements $\mathbf{z}^k$ in the update stage. Note that the segmentation system does not use the filtered parameters during the EM iterations, but only for initialization, which is its most critical phase. This allows for the overall system to cope with situations such as when individual images are corrupted by motion blur or when nearby objects are temporarily merged due to poor contrasts, without making the system overly slow at responding to rapid changes.

Figure 1: The upper shaded area represents the inner segmentation and scene part modeling loop, which runs multiple times per frame, whereas the lower area represents the outer Kalman filtering loop, that is run once per frame. Labeling is done using model parameters from either initialization θ_{init} , from the predicted parameters based on previous frame θ_{pred} , or from the last update of the current frame θ . The Kalman filtered parameters θ_{filt} are used for external processes, such as motor control. The prediction over time relies on optical flow estimates $\mathbf{u}$.

For both foreground objects and planar surfaces, predictions of parameters over time are guided by estimates of the dominating optical flow. Scale-invariant SIFT features [65] are tracked between frames and for each component an estimate $\mathbf{u} = (u_x, u_y)$ is given as the median x-wise and y-wise flow, computed over the corresponding region from the segmentation of the previous frame. For real-time performance feature extraction, as well as matching, can be done using a GPU implementation such as the one detailed later in Sec 8. With the median flow used as a control input $\mathbf{u}^k$ to the respective EKF, the process noise $\mathbf{w}^k$ expresses the uncertainties in how well this median flow can predict the change in parameters.

Planar surface EKF. The effect a moving camera has on the planar surface parameters, depends on the nature of the motion. Rotations around the normal of the plane leave all the parameters unchanged. Similarly, only translations along the normal affect the parameters and do this as a uniform scaling of all three parameters. In most typical scenarios, however, this translation is insignificant compared to a rotation, especially if the surface is horizontal and the camera keeps a fixed distance from the ground.

Since depths are unaffected by rotations, the disparity is constant for an image point that has moved due to a rotation. Furthermore, if rotations around the camera axis are assumed to be small, the optical flow induced by a rotation can be approximated by a constant flow across the whole image. With (a'_p, b'_p, d'_p) being the parameters after a rotation and (u_x, u_y) the median flow estimated by feature tracking, the disparity at point (x, y) can thus be written as

$$\begin{aligned} d(x, y) &= a'_p x + b'_p y + d'_p = a_p(x - u_x) + b_p(y - u_y) + d_p = \\ &= a_p x + b_p y + (d_p - a_p u_x - b_p u_y). \end{aligned}$$

This leads us to the process transition function of the corresponding EKF that, using a constant velocity assumption for the surface parameters, is given by

$$f(\mathbf{x}_p, \mathbf{u}) = [a_p + \dot{a}_p, b_p + \dot{b}_p, d_p + \dot{d}_p - a_p u_x - b_p u_y]^\top, \quad (24)$$

where the observation function is

$$h(\mathbf{x}_p) = [a_p, b_p, d_p]^\top. \quad (25)$$

The state consists of the three surface parameters and their corresponding velocities, that is $\mathbf{x}_p = [a_p, b_p, d_p, \dot{a}_p, \dot{b}_p, \dot{d}_p]^\top$. Note here that $\dot{d}_p$ does not really correspond to the velocity of d_p , but to a systematic bias between the change of d_p and what can be predicted by $a_p u_x + b_p u_y$. Due to the orientation of the plane and the centroid of the tracked features, this bias changes over time.

For the experiments given below we use a covariance matrix for the process

$$\mathbf{W}_p = \begin{pmatrix} \mathbf{A}/3 & \mathbf{A}/2 \\ \mathbf{A}/2 & \mathbf{A} \end{pmatrix},$$

with $\mathbf{A} = \text{diag}(0.0004, 0.0025, 4)$, which can be seen as the velocity covariance due to acceleration. For the measurement noise we use a covariance matrix $\mathbf{V}_p = \text{diag}(0.001, 0.001, 4)$. The process noise was estimated by minimizing the prediction errors over a series of recorded image sequences, while the measurement noise was estimated from multiple images of the same static scenes.

Foreground object EKF. The combined effect of a camera rotation and translation on the foreground object parameters is a translation in the image plane. Typically, the rotational flow components dominates the translational one. Only if the camera approaches the object, do the disparities change. Given that the control input, i.e. the estimated median flow (u_x, u_y) computed with feature tracking, gives a good prediction of the change in image positions, the process function

$$f(\mathbf{x}_f, \mathbf{u}) = [x_f + u_x, y_f + u_y, d_f + \dot{d}_f]^\top, \quad (26)$$

is used for prediction of the parameters, where the state is given by $\mathbf{x}_f = [x_f, y_f, d_f, \dot{d}_f]^\top$ and the observation function by

$$h(\mathbf{x}_f) = [x_f, y_f, d_f]^\top. \quad (27)$$

Since each object only covers a limited portion of image space and since objects are typically fronto-parallel as seen in the images, a bias similar to that of planar surface, is insignificant for most foreground objects and can thus be disregarded.

Measurement and process covariance matrices for the foreground object were found using a method similar to that of the planar surfaces. The matrices used in the experiments below were respectively set to $\mathbf{V}_f = \text{diag}(4, 4, 1)$ and

$$\mathbf{W}_f = \begin{pmatrix} 64 & 0 & 0 & 0 \\ 0 & 64 & 0 & 0 \\ 0 & 0 & 0.12 & 0.18 \\ 0 & 0 & 0.18 & 0.36 \end{pmatrix}.$$

6.3. Validation

Fig. 2 shows some images with segmentation from a sequence for which the motion is of the most problematic kind, i.e. a translation along the normal of the plane, combined with

Figure 2: Sequence from a hand-held Kinect camera in which five objects and a flat table top are tracked over time. The camera is moved (about 20cm) up and down along to normal of the plane, while being rotated to make object remain in view. The images (from left to right) correspond to the 1st image of the sequence and its disparity map, as well as the 1st, 10th, 20th and 30th images with segmentation.

Figure 3: Sequence from a hand-held Kinect camera in which five objects and a flat table top are tracked over time. The camera is moved (about 40cm) from left to right in the plane, while being rotated to make object remain in view. The images (from left to right) correspond to the 1st image of the sequence and its disparity map, as well as the 1st, 6th, 17th and 23rd images with segmentation.

a rotation that generates a similarly directed optical flow. The camera is elevated up and down about 20cm, but to prevent the segmented objects from falling out of view, the camera undergoes some quick compensatory rotations around frame 20. The sequence in Fig. 3, shown for comparison, involves lateral motion over the table-top, a motion significantly easier to cope with, since the surface parameters only change due to noise.

The changes of the surface parameter d_p obtained after segmentation, using the EKF prediction for initialization of each new image frame, is illustrated by the thick solid lines in Fig. 4 for the corresponding two sequences. Predictions using the full filter are shown as thin solid lines. The figure also shows predictions without a constant velocities assumption (dashed lines), instead assuming that parameters change only due to noise, and predictions without control inputs (dash-dotted lines). For each version of filters, the optimal covariance matrices were sought using the same method. From the left graph, representing the more problematic sequence, it can be concluded that the velocity assumption is essential to prevent lag in the predictions and that the control inputs reduce over-shoots during accelerations.

The average prediction errors using the full filter were 1.03 and 0.60 pixels over the full sequences, while for the predictions without control inputs, the average errors respectively in-

Figure 4: Predictions of the flat surface parameter d_p (thick solid) during the two sequences in Fig. 2 (left) and Fig. 3 (right), using either a Kalman filter with a constant velocity assumption and control input (thin solid), a static assumption and control input (dashed) or a fixed velocity assumption without control input (dash-dotted). As can be seen from the graphs, the control inputs are able to capture rapid disturbances, while the constant velocity assumption prevents lag.

Figure 5: Six consecutive frame from a sequence processed and recorded live in about 13 Hz, while the camera is rapidly moving back and forth, resulting in significant motion blur, as can be seen around the keyboard in the fourth frame.

creased to 1.33 and decreased to 0.45 pixels. This can be explained by the fact that the flow estimation process itself is noisy. Thus control inputs only benefit in cases of large disturbances, whereas the constant velocity assumption is important to prevent lag and keep the average errors low. For the same two sequences, the average image position errors of the foreground objects were 0.95 and 0.81 pixels, which can be compare to 3.55 and 1.98 without control inputs.

The last set of images in Fig. 5 show the system running close to its breaking point with the camera moving rapidly back and forth. The blue and green areas show detected planar surfaces, with a spray bottle in the center of view as foreground. With segmentation and SIFT features computed using a GPU, as will be explained later in Sec. 8, it runs in about 13 Hz. In particular the fourth image show significant motion blur that affects the segmentation of the spray bottle. The maximum change in the center position is about 50 pixels from one frame to the next, i.e., a full image width in one second.

7. Segmentation experiments

In Sec 3 five different MRF inference based methods for segmentation were described. The first one, LBP-S, updates scene part models based on a marginalization over labelings, whereas the remaining four uses the MAP estimate of the labeling using the current set of model parameters in each iteration. LBP-S

can thus be expected to be more tolerant to poor initializations, since it considers the full space of possible labelings, while the maximization method only uses the one that happens to maximize the a-posteriori probability.

In this section we evaluate the inference methods and scene part models by performing a series of experiments on object segmentation. The focus is quite different from those in [31, 32], where energy minimization methods were benchmarked based on their convergence properties and abilities to minimize a global energy. For an iterative procedure where labeling and scene part modeling are interleaved the dynamic changes, compared to cases with fixed objective functions. The experiments in this section address the effect of this dynamic. We will later return to the methods in Sec 8 and study how well they can be adapted for parallel processing in real-time.

A customary stereo camera system with two conventional cameras was used for the experiments, with binocular disparities provided by correlation based stereo matching using OpenCV. Correlation is fast enough for the intended application areas, but has the weakness of not being able to provide reliable disparities in image regions without texture. Thus large areas within the tested images contain pixels without 3D measurements, a fact that complicates initialization in particular, which is illustrated by the two examples in Fig. 6. Despite the availability of better performing method based on global optimization, such methods were not considered, due to their increased requirements on computational cost.

For initialization of scene part models we used the procedure described in Sec 5, with some modifications to facilitate benchmarking with respect to the provided ground truth. After foreground models have been automatically initialized, the resulting segments, referred to as a target points below, are paired up with the ground truth ones in the order of decreasing overlap, so that each foreground model is benchmarked against its own ground truth segment. This is true even if a target point does not intersect any of the ground truth segments, which is thus reflected in the reported precision and recall scores.

Figure 6: Two example of disparity maps and resulting segmentation using LBP-M. In the first case no planar background model is found in the initialization and in the second case the white jug fails after initialization. Since false positive disparities have a severe effect on segmentation, only the most confident measurements are used. This leads to sparsity in the disparity maps, especially in cases of limited texture, such as in these two examples.

7.1. Single object segmentation

To test the relative performance of the five methods for single object segmentation, we performed a series of experiments in 100 different scenes, where an object is placed on a table top, initially without the influence of any nearby disturbing objects. Half of these scenes have a textured table top, while in the remaining the table top is uniformly colored. Using an ball with a diameter equivalent to about 20% of the image height around the point for initialization, we got the results shown in Fig. 7.

	LBP-S	LBP-M	TRW-S	GC	ICM
F1 score	0.900	0.882	0.890	0.895	0.868
precision	0.904	0.925	0.932	0.923	0.908
recall	0.902	0.862	0.867	0.875	0.853
energy	4.648	4.622	4.629	4.634	4.672

Figure 7: Single object segmentation performance in scenes with one object.

As can be seen from the results, all five methods give similar results, with one clear exception. Whereas LBP-S has a better recall rate, the maximization based methods lead to a higher average precision. Due to the marginalization, LBP-S can allow segments to grow and eventually cover regions that were initially labeled as part of the background. However, this comes at the cost of sometimes including parts that are in fact not part of the foreground object. Worth noting is that despite the fact that LBP-S and LBP-M do not aim at minimizing the total energy, i.e. the negative logarithm of the joint probability, they still end up at similar minima. Furthermore, since ICM performs optimization locally, it is most likely to get stuck in local minima and thus results in the highest average energy.

A similar set of experiments were conducted on 141 scenes that each involved three objects in close proximity, often in direct contact. Target points were selected one object at the time and the resulting segmentations were studied. As can be seen in Fig. 8, the performance deteriorates for LBP-S in particular. The original seed often grows until more than one objects is covered by the foreground hypothesis, with decreasing precision as a result. Some characteristic examples of this are shown in Fig. 9. For examples with ICM a strong bias towards rectangular segments can be observed, which is a result of the local nature of the method and the fact that we are using 4-neighbors. In terms of F1 scores, LBP-M, TRW-S and GC lead to very similar results, slightly worse than the single object case.

	LBP-S	LBP-M	TRW-S	GC	ICM
F1 score	0.796	0.852	0.855	0.851	0.831
precision	0.745	0.867	0.858	0.849	0.850
recall	0.906	0.865	0.878	0.878	0.840
energy	5.036	4.945	4.941	4.962	5.000

Figure 8: Single object segmentation performance in scenes of three objects, with scores given in relation to a ground truth segment selected one at the time.

Since a target point is not necessarily located within the object targeted, it could happen that the object actually segmented

Figure 9: Characteristic examples of segmentation using LBP-S, LBP-M, TRW-S, GC and ICM respectively.

	LBP-S	LBP-M	TRW-S	GC	ICM
F1 score	0.830	0.875	0.878	0.876	0.854
precision	0.782	0.895	0.878	0.884	0.878
recall	0.919	0.880	0.898	0.891	0.856

Figure 10: Single object segmentation performance in scenes of three objects, with scores given in relation to the best matching ground truth segment.

is not the one originally targeted. This is particularly common for objects occluded by a more distinct foreground object. Such unintended changes in focus account for 11 test cases with LBP-S and 6 cases with the other four methods, which partly explains the lower observed F1 scores. In order to study how well the methods segment any one of the three objects, not necessarily the one originally targeted, we computed the average scores, precision and recall rates for the best matching ground truth segments out of the three possible. These results are shown in Fig. 10. Now LBP-M, TRW-S and GC perform almost as well as in the single object case and still without much differentiating the three methods. ICM lags behind slightly due to its higher sensitivity to local minima. LBP-S also improves, even if it still suffers from segments involving multiple objects, as indicated by the low precision scores.

7.2. Multi-object segmentation

We next performed the same experiments as before, but with three simultaneously segmented foreground hypotheses, with one target position selected per object. As can be seen in Fig. 11, the scores improve considerably, in particular for LBP-S. There are two reasons for this. First, due to the introduction of two more seeds, the dynamic of the segmentation loop changes. If a method gets stuck in a poor local minima in one iteration, it is likely to fall out of this minima in the next, due to the interaction of neighboring segments. Second, with three foreground hypotheses, a segment that would otherwise have grown to include multiple objects is prevented from doing so. This is particularly true for LBP-S that despite a higher energy has an average score competitive to those of the maximization based methods. Some characteristic examples of multi-object segmentation are shown in Fig. 12. The results are very similar for all methods, besides ICM that still exhibits bias towards

rectangular shapes.

	LBP-S	LBP-M	TRW-S	GC	ICM
F1 score	0.878	0.882	0.897	0.885	0.869
precision	0.856	0.935	0.946	0.931	0.912
recall	0.921	0.851	0.863	0.857	0.841
energy	4.800	4.689	4.696	4.714	4.748

Figure 11: Multi-object segmentation performance in scenes of three objects.

In many practical situations the above experiments can hardly be considered as realistic, since it is assumed not just that the number of available objects is known, but also that a target position is given for each object. For cases in which the number of objects is unknown, the scores in Fig. 10, with examples in Fig. 9, are more relevant, as they indicate the performance of segmentation one hypothesis at the time.

7.3. Effect of initial target size

Most critical to typical systems for foreground segmentation is the initialization phase. What occurs in the first few iterations, directly affects the end result. Unless some post-processing is later applied to assess the quality of segmentation, there is usually no turning back after initialization. As mentioned earlier in Sec 5, initialization is done by placing a 3D ball around a density peak in 3D space. The distance to and the projective size of this ball is given by a mean shift procedure. Points within the ball after convergence are assumed to belong to the initial foreground segment. Unfortunately, if the observed objects are unknown, then so are their respective sizes. In many cases, however, the sizes are known to some degree. For example, for the experiments performed here, objects are assumed to be measured in decimeters, with observation distances given in meters, which limits the range of possible projective sizes.

The results in Fig. 13 show how the different methods perform while varying the projective diameters of the initialization balls, in essence the size of the mean shift kernel. Worth noting is that while LBP-M, TRW-S and GC perform best with the optimal ball size, they are less robust to balls that become too small, GC slightly more robust than the message passing methods. The maximization based methods tend to miss small

Figure 12: Characteristic examples of segmentation using LBP-S, LBP-M, TRW-S, GC and ICM respectively.

differently colored object regions, if they are not initially covered by the ball. LBP-S shows the best ability to gradually grow from a small seed to include also such regions, as can be seen in the example in Fig. 14. However, the ability to grow comes at the cost of sometimes growing beyond the extent of the segmented object, leading to relatively low segmentation scores on average.

Initial size	LBP-S	LBP-M	TRW-S	GC	ICM
$0.05 \times H$	0.810	0.719	0.747	0.783	0.675
$0.10 \times H$	0.822	0.814	0.819	0.805	0.766
$0.20 \times H$	0.830	0.875	0.878	0.876	0.854
$0.40 \times H$	0.794	0.860	0.855	0.857	0.823

Figure 13: The effect of varying initial target sizes on F1 scores with disparities on single object segmentation in scenes of three objects, with 3D balls used for initialization.

Figure 14: The 1st, 4th and 15th EM iteration using an initial target size of $0.05 \times H$ with the methods LBP-S (upper) and GC (lower). Maximization methods, such as GC, easily get stuck in uniformly colored initial regions.

The benefit of disparities in the initialization is that points originating from neighboring objects can be pruned. This becomes an issue the larger the target ball is, which can be concluded from the table in Fig. 15, where depth information has been ignored and only 2D discs are used for initialization. Surprisingly, for small disc sizes the segmentation may even improve without disparities. The reason is because, for some uniformly colored objects, the number of accurately computed disparities become too few, if the target ball is too small, which

limits the number of samples that can be used for initialization of color histograms.

Initial size	LBP-S	LBP-M	TRW-S	GC	ICM
$0.05 \times H$	0.809	0.733	0.758	0.746	0.697
$0.10 \times H$	0.826	0.833	0.827	0.821	0.787
$0.20 \times H$	0.798	0.848	0.844	0.844	0.821
$0.40 \times H$	0.708	0.725	0.729	0.724	0.706

Figure 15: The effect of varying initial target sizes on F1 scores without disparities on single object segmentation in scenes of three objects, with 2D discs used for initialization.

7.4. Importance of modeling components

As mentioned in Sec 4, the background model consists of two kinds of components, a uniform component and planar ones. The planar components are intended to capture the existence of dominating planes, such as walls, floors and table tops, while the uniform component should capture all remaining non-foreground points, including scene objects currently not targeted. The reason for introducing planar hypotheses were to better separate objects from the supporting surfaces that objects are placed upon. The importance of planar hypotheses was studied by repeating the experiments with these hypotheses removed. By comparing the results in Fig. 16 to those in Fig. 10, one can conclude that without planar hypotheses somewhat better recall scores come at the cost of a substantially lower precision. Foreground segments tend to grow and capture also parts of the supporting surfaces. This is particularly true for objects with color distributions similar to those of the surfaces. Since the surfaces is in direct contact with the objects, the spatial information does not provide enough evidence to separate the two. This effect occurs regardless of method, but it most critical for LBP-S.

What frequently occurs in practice is that, despite the lack of a distinct dominating plane in the scene, a planar hypothesis is still found and included to the model. This is often the case for table-top scenes, where objects are located within a narrow range just above a supporting surface, and a large enough number of 3D points from different objects are located on a plane intersecting the scene and bounding the rest of the scene. As can

be seen in Fig. 17, the proposed model is able to cope also with such situations. The detected planar region will grow in thickness after initialization and eventually capture a set of background objects in the scene (shown in gray), however, without affecting the quality of the segmented foreground object. In fact, this happens for many cases in the quantitative experiments when the background is uniformly colored. The incorrectly detected plane becomes just another background component.

	LBP-S	LBP-M	TRW-S	GC	ICM
F1 score	0.755	0.822	0.831	0.826	0.767
precision	0.661	0.786	0.797	0.786	0.709
recall	0.942	0.907	0.911	0.913	0.894

Figure 16: Results without using planar hypotheses in scenes of three objects. Compared to the results in Fig. 10, the precision score are considerably lower, due to object segments growing onto their supporting surfaces.

Figure 17: Segmentation using LBP-M with a planar background component in the model, despite the lack of a dominating planar surface in the scene. The planar component captures some background objects (shown as gray regions), but without affecting the foreground segment.

It can be argued whether disparities are really necessary for successful segmentation. After all, most methods for foreground segmentation do not assume disparities and by expecting disparities, one is constrained to set-ups with multiple cameras or time-of-flight cameras. Disparities can in fact do more harm than good in scenes with large textureless areas, where accurate disparities are hard to determine. Stereo matching may also suffer from fattening of textured objects placed in front of uniformly colored backgrounds. As shown by the results in Fig. 18, where disparities have been disregarded, segmentation is helped by disparities, even if the impact is moderate compared to the scores in Fig. 10.

More important is the color information, as can be seen from the results in Fig. 19, where such information has been ignored and only luminance is used, together with 3D distance measures. Both precision and recall deteriorate to a similar extent for all methods. In a concluding experiment we try the same methods with only color information and ignore the distance measures altogether. Here the GC case is very similar to Grab-

	LBP-S	LBP-M	TRW-S	GC	ICM
F1 score	0.790	0.835	0.841	0.833	0.798
precision	0.739	0.844	0.831	0.836	0.809
recall	0.883	0.856	0.876	0.853	0.818

Figure 18: Segmentation results without disparities (with 2D distance measures and colors) in scenes of three objects.

Cut [3], but with histograms, rather than mixtures of Gaussians for color modeling. From the poor results in Fig. 20, the importance of distance measures for segmentation is highlighted. For all five methods recall suffers considerably, which is an indication that without such measures they all fail to grow beyond the original seeds, when different parts of objects are of colors not included in the initial foreground segment.

	LBP-S	LBP-M	TRW-S	GC	ICM
F1 score	0.720	0.785	0.793	0.777	0.743
precision	0.676	0.821	0.822	0.805	0.775
recall	0.820	0.801	0.814	0.803	0.771

Figure 19: Segmentation results without colors (with 3D distance measures) in scenes of three objects.

	LBP-S	LBP-M	TRW-S	GC	ICM
F1 score	0.575	0.668	0.643	0.628	0.634
precision	0.626	0.797	0.761	0.839	0.756
recall	0.580	0.627	0.598	0.593	0.588

Figure 20: Segmentation results with colors only in scenes of three objects.

7.5. Concluding remarks

The question is which method to select for object segmentation, considering the differences and similarities in performance and implementation complexities. From the quantitative experiments it can be concluded the most simplistic method, ICM, is only competitive in the case of single object segmentation without any nearby obstructing objects. Among the top contenders, LBP-M, TRW-S and GC, no single method showed to be superior, which is somewhat contradictory to the study in [32] that suggests that LBP-M ought to be less competitive to the other two, at least when benchmarked in terms of energy minimization. Due to the dynamics of the segmentation loop, LBP-M is helped from being stuck in local minima. The marginalization based method, LBP-S, show somewhat different properties than the other methods. Despite lower scores in almost all experiments, LBP-S is by far the most robust method to too small initialization seeds. In fact, in most cases LBP-S is able to grow from only a handful of pixels, while the maximization methods easily get stuck.

More important than the choice of method is the scene part modeling. Planar background models help to separate foreground objects from the surfaces they are placed on in cases of similarities in color. Even if a false planar model is instantiated, it becomes just another background model, without affecting the foreground object segments. When it comes to the

importance of cues for segmentation, disparities are less valuable than colors and image positions. Its most important benefit is in pruning of neighboring pixels in the initialization phase, the most critical part of a complete system. Color information is important, but only when combined with image distance measures, as can be seen by comparing Fig. 18 and Fig. 20.

8. Real-time system

In this section we study the alternative methods in the context of a complete segmentation and tracking system for real-time use. As illustrated in Fig. 1 the system consists of four steps running in two loops, an inner segmentation loop with Labeling and Modeling, and an outer loops that also include the Kalman filter steps Update and Predict. In terms of computational cost these two last steps are insignificant. However, the Predict step also relies on control inputs $\mathbf{u}^k$, that originate from SIFT features being tracked over time. One might argue against SIFT for this purpose, given that scale invariance is hardly a necessity under the given conditions. The reason for using SIFT anyway, is to allow for future applications that include on-line object recognition as a mean to initialize new segments as soon as previously seen objects are detected. This is, however, not covered within the scope of this paper.

The inner segmentation loops is heavily dominated by the Labeling step that applies MRF inference for segmentation. The table in Fig. 8 includes a summary of average computational costs on a four-core Intel Core i7-860 CPU for the five tested methods, using a representative set of 27 test images with varying number of labels. The GC method, the fastest among the five, uses Fast-PD [30], a primal-dual energy minimization method that improves the speed, compared to α -expansion, by exploiting information from the dual problem to limit the size of graphs in later iterations. For 7-label problems the improvement in average speed is about 2.5 times on our particular test set, but is likely to be larger for problems with more labels.

From the table it can be concluded that the benefit of multiple threads for message passing with LBP-M and TRW-S is smaller than one would hope for. This is due to the sheer amount of messages and beliefs that has to be loaded and stored in each iteration, which makes the methods highly bandwidth limited. Since data are accessed sequentially and are more than what the cache hierarchy allows, the memory bandwidth is exhausted already with a few threads. Further optimization using e.g. assembly intrinsics do not lead to any improvements in speed. Furthermore, with multiple threads neither LBP-M nor TRW-S can cut the memory consumption for messages by half, by reusing the memory used for incoming and outgoing messages, which is possible for single threaded implementation [61, 27], since the ordering of processing cannot be guaranteed with multiple threads. With push-relabel [66] applied for graph cuts, GC does a much better job at limiting the amount of accessed memory.

Due to its better convergence properties, the sequential tree-reweighted message passing algorithm [27] is usually superior to max-product [58]. However, when run in a coarse-to-fine hierarchy [61] with only a few iterations per level, this benefit

is not as clear. In fact, with the same number of iterations per level LBP-M is slightly faster than TRW-S, because TRW-S not only updates the messages in each iteration, but also the beliefs (21). In our case, where we used 5-level hierarchies, we were unable to show that TRW-S would need fewer iterations than 8 per level to reach convergence.

Unfortunately, none of the tested methods showed to be fast enough for real-time use cases, at least for multi-object tracking that requires an update rate of at least 10 Hz. As a consequence of this, implementations on graphics cards (GPUs) were considered for the purpose. Even if it has been argued that the benefits of GPUs for general purpose programming have been exaggerated [67], there are few applications that can benefit from GPUs as well as image processing applications, as long as they are complex enough to motivate the cost of transferring images from main to GPU memory. SIFT feature extraction [65] is one such application and so is MRF inference in regular grids. Thus to show that the work presented in this paper can feasibly be run in real-time, we implemented the most time-consuming parts of the complete system on a GPU, an NVidia GeForce GTX 580, and compare the performance to that of the CPU implementations.

8.1. GPU implementations

GPUs from NVidia, as well as from AMD, consist of a set of multiprocessors (MP) that each has a number of stream processors (SP) working in parallel. The GTX 580 GPU, for example, has 16 MPs with 32 SPs each. SPs act in a simple-instruction multiple-thread (SIMT) fashion, which is very similar to SIMD with a few exceptions. If all SPs in a MP access the same memory address or if they access a sequence of consecutive addresses, this can be exploited by the MP for maximum bandwidth. GPU software is written as *kernels* that express work in terms of thousands of *threads*, with multiple threads grouped into *blocks*. During execution each block is scheduled to a MP, with its threads mapped to the SPs of that MP. To hide memory latencies and keep all SPs fully active, multiple blocks can simultaneously be scheduled to the same MP.

Thanks to the mass of computational power available by the total set of SPs, the real challenge of GPU programming is understanding the memory architecture and making sure that each SP is working at its full potential. GPUs typically have a multi-level memory architecture, much like of conventional cache hierarchies of CPUs. Each MP has a couple of hundred kilobytes of high-speed local memory available, some of which is shared between its SPs, while the rest is local to each SP. Maximum performance is achieved, if implementations are designed such that most accesses are limited to this local memory, without relying on the slower, but larger, off-chip global memory that is available to all MPs.

8.1.1. MRF inference

In Sec 7 it was shown, that for multi-object segmentation, LBP-M, TRW-S and GC all lead to very similar end results, despite their differences in ability to minimize the objective functions. Among these, GC was the fastest method on the CPU

	LBP-S 1 thread	LBP-M		TRW-S		GC 1 thread	ICM 1 thread
		1 thread	4 threads	1 thread	4 threads		
3 labels	570	180	140	260	190	160	100
5 labels	1110	520	230	570	320	290	150
7 labels	1630	770	330	920	480	410	200

Figure 21: Computational cost (in milliseconds) for the five tested MRF inference methods, using a set of representative images in VGA with a varying number of labels per pixel. GC was run until convergence, while the other methods used similar 5-level hierarchies, with 8 iterations per level. All methods, except LBP-S, were implemented using integer arithmetics.

and was thus first to be considered for GPU implementation. GC uses Fast-PD that in turn relies on the push-relabel [66] algorithm, an algorithm that gains its speed from only updating those nodes for which you have excess flow to push and a path towards the drain, this using a distance to drain value for each node. However, this also makes it particularly hard to extend for parallel processing.

Vineet and Narayanan [68] used two different kernels in their GPU implementation of push-relabel, one for pushing of excess flow and one for relabeling of distance values. These kernels are interleaved in multiple iterations and to improve speed, thread blocks were deactivated, if no excess flow could be pushed in the last iteration. If Fast-PD were to be implemented on a GPU using such an approach, it should take about 100-150 ms¹, assuming the same speed-up as the case of the binary labels. This is, however, overly optimistic given the harder problem of parallelization. The reason why GPU optimization is not more beneficial is likely to be because of the substantial additional cost of either using atomic functions or larger global memory buffers for propagation of excess flow between thread blocks.

A factor that complicates graph cut based labeling on GPUs is that a labeling is not available until convergence is reached and a minimum cut is found. Typically, small islands of activation exist around the boundaries of segments, islands that might remain for hundreds of iterations. Even if thread blocks are deactivated when no excess flow remains, multiple kernel calls are still required with only a handful SPs being used in each call. In contrast, even if deactivation of blocks is harder to achieve with belief propagation, it is possible to interrupt message passing after a given number of iterations and still get satisfactory results. Thus we also considered the LBP-M method for GPU implementation.

Most GPU implementations of belief propagation applies a coarse-to-fine structure [61] for faster convergence, some using extensions to OpenGL [69, 70] and others using more recent general purpose APIs [71, 72]. Yang et al. [70] uses deactivation of converged pixels to improve speed. This is, however, hard to extend to large SIMD-like architectures, where deactivation is preferably done per block. Overlapping tiles have also been proposed [71, 73], with most computations done locally within each tile. However, due to the local memory required for storing of messages, one becomes limited to relatively small tiles, which makes it harder to keep all SPs fully active. One further needs to propagate messages between tiles,

which means that multiple kernel calls are still required, with propagated messages stored to global memory. Since a hierarchical framework reduces the number of iterations per level anyway, its hard to simultaneously exploit the joint benefits of both local tile processing and hierarchies [73].

For our GPU implementation of LBP-M we apply the max-product (min-sum) method as proposed by Felzenszwalb and Huttenlocher [61], but without reusing memory for incoming and outgoing messages, due to the parallelism. To keep the majority of operations as SIMD-like as possible and exploit the total bandwidth of the GPU, we ignored the potentials of deactivation and local processing in tiles, and kept all processing in one compact kernel that is called iteratively, 8 times per hierarchy level. Assuming that Potts model is used, the operations executed in this kernel can be summarized in two steps:

$$\begin{aligned} \text{(S1) foreach } l_i \text{ do } & h_{ij}(l_i) \leftarrow \psi_i(l_i) + \sum_{k \in N_i \setminus j} m_{k \rightarrow i}(l_i) \\ \text{(S2) foreach } l_j \text{ do } & m_{i \rightarrow j}(l_j) \leftarrow \min(h_{ij}(l_j), \min_{l_i} h_{ij}(l_i) + d_{ij}) \end{aligned}$$

Here d_{ij} represents the pair-wise cost of assigning nodes i and j to two different labels. The first step S1 involves four global memory reads and one local store, whereas S2 requires one local read and one global write, per label and message passing direction. If we assume 4-connectedness, it is possible to interleave message computations from the same node, but in different directions, and reduce the total number of global memory operations. By doing so, S1 will be limited to five global reads in total per label, and S2 to four global writes, including d_{ij} that requires one global read per pair. Fewer global memory operations than that is not possible to achieve, unless the structure of the graph is changed.

Operations are scheduled to MPs in terms of thread blocks. In our implementation we use one thread per node in the graph, since do not assume to know the number of labels beforehand, just that its limited. Unfortunately, with the above mentioned scheme the optimal number of threads per block is only 128 threads for 7-label problems. The reason for this is that the $h_{ij}(l_i)$ values have to be computed and stored, and cannot be used in S2 until $\min_{l_i} h_{ij}(l_i)$ has been determined. This requires local memory for temporary storage of $h_{ij}(l_i)$, which will sets a limit on the number of threads that can be kept in each block. A solution to this problem is to change the order of S1 and S2, and store $h_{ij}(l_i)$, instead of $m_{i \rightarrow j}(l_j)$ in each iteration. First the kernel runs S2 completing the operations from the previous iterations and then S1 for the following one. Regardless of the number of labels, 768 threads can then be used at all time, since the local memory requirement does not depend on the number of labels.

¹This conclusion is drawn from extrapolating results of binary to multi-label problems using code available by the authors of [30, 68].

	M1: original	M2: $m_{i \rightarrow j}(l_j)$	M3: $h_{ij}(l_i)$
3 labels	13.1	6.3	6.0
5 labels	18.3	8.9	8.0
7 labels	22.6	11.8	10.0

Figure 22: Computational cost (in milliseconds) of the three GPU versions of MRF inference in LBP-M, using a 5-level hierarchy with 8 iterations per level. M1 interleaves messages from the same node, but in different directions. M2 interleaves and stores messages. M3 interleaves messages, but stores $h_{ij}(l_i)$. To make the costs comparable to those in Fig. 8, an additional cost of 4.9 ms has to be included for 7 labels, to set up the graph structure, allocate memory and transfer data between CPU and GPU memory.

As soon as $h_{ij}(l_i)$ is computed it is stored, and it is not read back until needed. Since the order of S1 and S2 is changed, the procedure has to start with a S1-only iteration and end with a S2-only one. Furthermore, one additional global load and store per direction is necessary for $\min_l h_{ij}(l_i)$. This extra memory cost is low compared to the per label cost of storing $h_{ij}(l_i)$.

A summary of the computational cost for the alternative GPU kernels can be seen in Fig. 22. The first (M1) performs all the operations one message passing direction at the time, while the second (M2) interleaves the message computations in different directions. The reduction from 16 (four per direction) global reads to only five, clearly has a large influence on the performance. The third alternative (M3) also interleaves messages, but stores $h_{ij}(l_i)$ instead of $m_{i \rightarrow j}(l_j)$ to global memory. Since the kernel is already bandwidth limited, this only slightly improves the performance, despite the increased SP utilization. However, with 13 or more labels M2 run out of local memory and M3 becomes the only alternative. Note that even if an additional cost of 4.9 ms for setting up the graph structure, allocating memory and transferring data between CPU and GPU memory is included, M3 is still about 50 times faster than the single-threaded CPU version or 23 times faster than the multi-threaded one.

8.1.2. SIFT feature extraction

Another time-consuming part of the complete system is the Predict step that relies on SIFT features that are tracked over time. The SIFT extractor [65] has two components; a detector based on maximization of Differences of Gaussians in image space and scale, and a descriptor based on local gradient histograms. While the descriptor works per detected feature and has a random access like memory access pattern, the detector is computed for every pixel in a Gaussian pyramid, which makes it more easily parallelized. Thus even if the detector performs considerably more operations, the computational cost of the two components are similar for an heavily optimized CPU version, as can be seen in Fig. 23. The numbers are given for a 30 scale Gaussian pyramid, where the resolution has been reduced to half every 6 scales (octave). The cost of the detector is also proportional to the number of pixels, while the descriptor instead depends on the number of extracted features.

This dependency can not be seen for a naive GPU implementation that is similar to the CPU version in that the detector and descriptor are run once per scale. Since images quickly become

very small when subsampling is done for every octave, there is not enough work for all SPs to stay active, when the number of pixels and extracted features become too few. In an optimized GPU version, features are instead extracted simultaneously for all scales within an octave. Thus instead of creating a Gaussian pyramid one scale at the time through gradual low-pass filtering, all scale are created in parallel by directly blurring the finest scale with filter kernels of different sizes. Once features have been detected, descriptors for all these features are similarly created in parallel regardless of scale. As shown in the table, this leads to a radical reduction in computational cost, for the descriptor in particular.

Other costs that have to be considered in GPU implementations are those of memory allocation and transfer of image data to and SIFT features from the GPU. The memory allocation is particularly costly on the GPU when it is done naively, that is when it is done just before it is needed. Some cost is associated to the amount of memory allocated, but dominating is a fixed cost per memory allocation call. Even if temporary buffers were reused as much as possible in the naive implementation, ten different allocations were needed per octave. To reduce the memory allocation cost, these were instead replaced by one large allocation at the beginning of the process and left the software divide this memory upon need.

With these two seemingly trivial optimizations, the SIFT extractor becomes almost three times faster for a QuadVGA (1280x960 pixel) image and even faster for VGA (640x480 pixel) images. Compared to the optimized CPU version, it is between 12 and 19 times faster depending on resolution. The total cost of 5.4 ms required for the optimized GPU version in VGA can also be compared to about 440 ms for the same image and settings on the same CPU using VLFeat [74], one of the most frequently used implementation of SIFT. In fact, for our particular application it is even faster, since we can disregard the cost of transferring SIFT feature data back from the GPU. Features are instead matched over time, as explained in Section 6.2, with the features kept on the GPU. The cost of brute force matching of all matching between two frame is about 1.2 ms in VGA on the GPU, about six times faster than optimized on a CPU.

8.2. Summary

With the above mentioned GPU optimizations the full system running LPB-M, with images and disparities provided by a Kinect unit, runs in 13.0 Hz with 3 scene part models, 11.1 Hz with 5 models and 9.8 Hz with 7 models, when the system runs in tracking mode with 3 segmentation iterations per update. The Modeling step, with one CPU thread running per model, takes about 16 ms and the Labeling step 45 ms in total with 7 models. The Predict step, including SIFT extraction, matching and tracking, takes 6 ms and 4 ms is associated to overhead, such as image extraction and color conversions. Despite the additional cost of about 30 ms for Kinect disparity extraction and disparity to image alignment, as well as visualization, the system is fast enough for real-time tracking as exemplified by the sequences of Fig. 2 and Fig. 3.

	CPU (opt)		GPU (naive)		GPU (opt)	
	QuadVGA	VGA	QuadVGA	VGA	QuadVGA	VGA
Download image to GPU			0.5	0.2	0.5	0.2
Memory allocation			5.2	4.5	1.4	0.6
DoG feature detection	140	34	11.7	6.2	6.9	2.3
SIFT feature description	102	35	16.8	11.7	2.6	1.6
Readback SIFTs from GPU			3.4	1.8	1.3	0.7
Total cost (ms)	242	69	37.6	24.5	12.7	5.4

Figure 23: The computational cost of SIFT feature extraction and the respective stages for an optimization CPU version, a naive GPU implementation and an optimized GPU one. For a QuadVGA (1280x960 pixel) image 1311 features were found, whereas 453 features were extracted in VGA (640x480 pixels). For the CPU implementation the cost of memory allocation is negligible and is thus excluded, and the cost of memory transfer between CPU and GPU is only relevant to the GPU implementations. The GPU implementations are done using the CUDA API from NVidia.

9. Conclusions

We have studied the problem of multi-object segmentation and tracking, and done so without letting the real-time constraint prevent us from using state-of-the-art energy minimization techniques for labeling. From experiments it has been concluded that the modeling of the problem is more important than the choice of MRF inference method. We have proposed a heterogeneous scene representation that consists of a background model of planar surfaces and uniform clutter, with foreground objects hypotheses modeled as 3D ellipsoids. It has further been concluded for grouping of pixels, distance measures in either 2D or 3D are just as important as colors, and that planar background models help to disambiguate objects from the surfaces that are placed on.

In experiments it has been shown that modern maximization based methods, such as max-product belief propagation (LBP-M), tree reweighted message passing (TRW-S) or primal-dual graph cuts (GC), lead to very similar end results, when run in an EM-like iterative framework with labeling and modeling interleaved. Unlike labeling problems with fixed energy terms, the dynamics prevent methods, LBP-M in particular, from being stuck in local minima. What can not be directly observed from the quantitative experiments, however, is that the paths to the final segmentation can be quite different. Foreground object seeds tend to evolve faster, but sometimes more erratic, with TRW-S and GC, which is reasonable given their better convergence properties for energy minimization. However, the minimization problem solved in each iteration is based only on statistics from the previous iteration, which might still be incomplete, if the full extent of an object has not yet been captured. Thus better convergence in the energy minimization does not necessarily translate into faster convergence overall.

Given that LBP-S was the only marginalization based approach in our study, it has properties rather different from those of the others. By considering all possible labeling in each iteration, not just the current MAP labeling, it shows the best ability to grow from very small initial foreground seeds to full object segmentation. Unfortunately, this also means that it is more likely to connect objects located close to each others, especially if these have overlaps in their color distributions. However, if competing seeds can be simultaneously initiated on each object, LBP-S do just as well as the top contenders, and should

thus not be disregarded as a candidate method. The same is true if the end goal is object modeling, not segmentation per se, since with marginalization the full space of labeling is taken into consideration for modeling.

As soon as you apply segmentation in a real-time settings, computational speed becomes an issue. Due to the costs associated with multi-label MRF inference methods, in terms of both speed and memory, such methods are rarely used for real-time purposes. We have in this study shown that MRF methods are in fact feasible, even for the purpose of object tracking, using GPUs for accelerations and given recommendations on how to use the hardware to its full potential. It has been concluded that, given the pattern of memory accesses, message passing methods are more suitable for high levels of parallelism, even if graph cut based methods are faster on single threaded CPUs. Given the similarities in segmentation results, message passing methods are thus recommended for real-time purposes.

A choice made in designing the real-time system was to let the segmentation in one frame only depend on the scene part models of the previous frame, but not on the previous segmentation, despite the temporal consistency that can often be expected. From one frame to the next, only the model parameters remain. The motivation for this modeling centric approach, was to allow for initialization of previously seen objects in new scenes, using as little prior information as possible. In later studies we intend to study the problem of re-initialization of such objects, when the scene has undergone different degrees of changes, from gradual changes in view points, to larger changes where objects temporarily leave the visual field, and to complete scene changes. For tracking we have already made an exception from the general rule and do in fact exploit segmentation from previous frames. The question is what information to use when more radical changes occur and no previous segmentations can be applied. A possible avenue is to combine SIFT features, extracted during earlier tracking, and the intrinsic scene part parameters, the shape and color information, for detection and re-initialization.

A segmentation system can never perform better than what its models and prior assumptions allow. As soon as you introduce abstractions in its representations, you may run into problems, if the object to be represented diverges from what can typically be assumed. What frequently occurs in multi-object

segmentation is the difficulty to disambiguate objects in direct contact. Are there two objects placed next to each others or is there just one of an unusual shape or texture? We believe that such questions can not always be resolved from single images alone and are exploring methods for active object hypothesis testing and validation [75]. We also use the presented system to learn more about objects through tactile exploration of object shape [76] and study objects in terms of object-related sensorimotor contingencies [77].

This leads to the more general question of active scene understanding, the theme of our current research, which involves the problem of representing scenes with objects and their relations, updating representations over time, as well as planning and executing hypothesis testing through actions to understand the scene well enough to solve whatever task is placed on the system. In the presented experiments, foreground models were automatically inserted on start-up, given the number of models and expected size. This does not necessarily have to be the case. In a fixating system, e.g., new models can sequentially be inserted each time a part of the scene that has not yet been visited is fixated on. If a previously fixated, and currently tracked, object is to be revisited, the corresponding model keeps information on where it is located in the scene and how to make a saccade towards it. Thus even if the presented multi-object segmentation and tracking system can be used for many different applications, applications vary in the way scene part models should be inserted, exploited and eventually discarded.

References

- [1] L. Itti, C. Koch, E. Niebur, A model of saliency-based visual attention for rapid scene analysis, *IEEE Trans. Pattern Anal. Mach. Intell.* 20 (11) (1998) 1254–1259.
- [2] Y. Boykov, M.-P. Jolly, Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images, in: *Proc. IEEE Int. Conf. Comput. Vis.*, Vol. 1, 2001, pp. 105–112.
- [3] C. Rother, V. Kolmogorov, A. Blake, Grabcut: Interactive foreground extraction using iterated graph cuts, *ACM Trans. Graph.* 23 (2004) 309–314.
- [4] A. Mishra, Y. Aloimonos, Visual segmentation of simple objects for robots, in: *Proc. Robotics: Sci. Syst.*, 2011, pp. 217–224.
- [5] P. Fitzpatrick, G. Metta, Grounding vision through experimental manipulation, *Philosophical Trans. Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 361 (1811) (2003) 2165–2185.
- [6] L. Montesano, M. Lopes, A. Bernardino, J. Santos-Victor, Learning object affordances: from sensory-motor coordination to imitation, *IEEE Trans. Robotics* 24 (1) (2008) 15–26.
- [7] C. Bibby, I. Reid, Robust real-time visual tracking using pixel-wise posteriors, in: *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 831–844.
- [8] P. Chockalingam, N. Pradeep, S. Birchfield, Adaptive fragments-based tracking of non-rigid objects using level sets, in: *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 1530–1537.
- [9] Y. Shi, W. Karl, Real-time tracking using level sets, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Vol. 2, 2005, pp. 34–41.
- [10] L. Grady, C. Alvino, The piecewise smooth Mumford-Shah functional on an arbitrary graph, *IEEE Trans. Image Process.* 18 (11) (2009) 2547–2561.
- [11] H. Sidenbladh, M. J. Black, D. J. Fleet, Stochastic tracking of 3d human figures using 2d image motion, in: *Proc. Eur. Conf. Comput. Vis.*, 2000, pp. 702–718.
- [12] M. de La Gorce, N. Paragios, D. J. Fleet, Model-based hand tracking with texture, shading and self-occlusions, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.
- [13] D. Koller, K. Daniilidis, H. Nagel, Model-based object tracking in monocular image sequences of road traffic scenes, *Int. J. Comput. Vis.* 10 (3) (1993) 257–281.
- [14] B. Leibe, K. Schindler, N. Cornelis, L. Van Gool, Coupled object detection and tracking from static cameras and moving vehicles, *IEEE Trans. Pattern Anal. Mach. Intell.* 30 (10) (2008) 1683–1698.
- [15] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models, *Int. J. Comput. Vis.* 1 (4) (1988) 321–331.
- [16] V. Caselles, F. Catté, T. Coll, F. Dibos, A geometric model for active contours in image processing, *Numer. Math.* 66 (1993) 1–31.
- [17] D. Mumford, J. Shah, Optimal approximations by piecewise smooth functions and associated variational problems, *Commun. Pure Appl. Math.* 42 (5) (1989) 577–685.
- [18] T. Chan, L. Vese, Active contours without edges, *IEEE Trans. Image Process.* 10 (2) (2001) 266–277.
- [19] T. Chan, W. Zhu, Level set based shape prior segmentation, in: *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Vol. 2, 2005, pp. 1164–1170.
- [20] D. Cremers, F. Tischhäuser, J. Weickert, C. Schnörr, Diffusion snakes: Introducing statistical shape knowledge into the Mumford-Shah functional, *Int. J. Comput. Vis.* 50 (2002) 295–313.
- [21] D. Cremers, G. Funka-Lea, Dynamical statistical shape priors for level set based tracking, *IEEE Trans. Pattern Anal. Mach. Intell.* 28 (2006) 1262–1273.
- [22] D. Freedman, T. Zhang, Active contours for tracking distributions, *IEEE Trans. Image Process.* 13 (4) (2004) 518–526.
- [23] J. Besag, On the statistical analysis of dirty pictures, *J. Royal Stat. Soc. B-48* (1986) 259–302.
- [24] L. Ford, D. Fulkerson, Maximal flow through a network, *Can. J. Math.* 8 (1956) 399–404.
- [25] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, *IEEE Trans. Pattern Anal. Mach. Intell.* 23 (11) (2001) 1222–1239.
- [26] M. Wainwright, T. Jaakkola, A. Willsky, Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches, *IEEE Trans. Inf. Theory* 51 (2002) 3697–3717.
- [27] V. Kolmogorov, Convergent tree-reweighted message passing for energy minimization, *IEEE Trans. Pattern Anal. Mach. Intell.* (2006) 1568–1583.
- [28] J. Pearl, Reverend bayes on inference engines: A distributed hierarchical approach, in: *Proc. Natl. Conf. Artif. Intell.*, 1982, pp. 133–136.
- [29] Y. Weiss, Correctness of local probability propagation in graphical models with loops, *Neural Comput.* 12 (1) (2000) 1–41.
- [30] N. Komodakis, G. Tziritas, N. Paragios, Fast, approximately optimal solutions for single and dynamic MRFs, in: *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.
- [31] M. F. Tappen, W. T. Freeman, Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters, in: *Proc. IEEE Int. Conf. Comput. Vis.*, 2003, pp. 900–907.
- [32] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, C. Rother, A comparative study of energy minimization methods for Markov random fields with smoothness-based priors, *IEEE Trans. Pattern Anal. Mach. Intell.* 30 (6) (2008) 1068–1080.
- [33] Y. Boykov, V. Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, *IEEE Trans. Pattern Anal. Mach. Intell.* 26 (9) (2004) 1124–1137.
- [34] D. Batra, A. Kowdle, D. Parikh, J. Luo, T. Chen, iCoseg: Interactive co-segmentation with intelligent scribble guidance., in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 3169–3176.
- [35] S. Vicente, V. Kolmogorov, C. Rother, Joint optimization of segmentation and appearance models, in: *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 755–762.
- [36] A. Mishra, Y. Aloimonos, Active segmentation with fixation, in: *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 468–475.
- [37] S. Bagon, O. Boiman, M. Irani, What is a good image segment? A unified approach to segment extraction, in: *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 30–44.
- [38] A. Blake, C. Rother, M. Brown, P. Perez, P. Torr, Interactive image segmentation using an adaptive GMMRF model, in: *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 428–441.
- [39] Y. Li, J. Sun, C.-K. Tang, H.-Y. Shum, Lazy snapping, *ACM Trans. Graph.* 23 (2004) 303–308.

- [40] A. M. Treisman, G. Gelade, A feature-integration theory of attention, *Cogn. Psychol.* 12 (1) (1980) 97–136.
- [41] J. M. Wolfe, K. R. Cave, S. L. Franzel, Guided search: an alternative to the feature integration model for visual search., *J. Experimental Psychology: Human perception and performance* 15 (3) (1989) 419.
- [42] C. Koch, S. Ullman, Shifts in selective visual attention: towards the underlying neural circuitry., *Human Neurobiol.* 4 (4) (1985) 219–27.
- [43] J. K. Tsotsos, S. M. Culhane, W. Y. Kei Wai, Y. Lai, N. Davis, F. Nuflo, Modeling visual attention via selective tuning, *Artif. Intell.* 78 (1) (1995) 507–545.
- [44] J. J. Clark, N. J. Ferrier, Modal control of an attentive vision system, in: *Proc. IEEE Int. Conf. Comput. Vis.*, 1988, pp. 514–523.
- [45] A. Oliva, A. Torralba, M. Castelhano, J. Henderson, Top-down control of visual attention in object detection, in: *Proc. IEEE Int. Conf. Image Process.*, Vol. 1, 2003, pp. 253–256.
- [46] G. Kootstra, N. Bergström, D. Kragic, Using symmetry to select fixation points for segmentation, in: *Proc. IEEE Int. Conf. Pattern Recognit.*, 2010, pp. 3894–3897.
- [47] D. Martin, C. Fowlkes, J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues, *IEEE Trans. Pattern Anal. Mach. Intell.* 26 (5) (2004) 530–549.
- [48] Y. Zhang, M. Brady, S. Smith, Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm, *IEEE Trans. Med. Imaging* 20 (1) (2001) 45–57.
- [49] R. Potts, Some generalized order-disorder transformation, *Proc. Camb. Philos. Soc.* 48 (1952) 106–109.
- [50] D. Geman, S. Geman, C. Graffigne, P. Dong, Boundary detection by constrained optimization, *IEEE Trans. Pattern Anal. Mach. Intell.* 12 (7) (1990) 609–628.
- [51] P. Kohli, L. Ladický, P. H. Torr, Robust higher order potentials for enforcing label consistency, *Int. J. Comput. Vis.* 82 (3) (2009) 302–324.
- [52] N. Vasconcelos, A. Lippman, Empirical bayesian em-based motion segmentation, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Los Alamitos, CA, USA, 1997, pp. 527–532.
- [53] W. T. Freeman, E. C. Pasztor, Learning low-level vision, in: *Proc. IEEE Int. Conf. Comput. Vis.*, 1999, pp. 1182–1189.
- [54] J. Besag, Spatial interaction and the statistical analysis of lattice systems (with discussion), *J. Royal Stat. Soc. B* 36 (2) (1974) 192–326.
- [55] D. Greig, B. Porteous, A. Seheult, Exact maximum a posteriori estimation for binary images, *J. Royal Stat. Soc. B* 51 (2) (1989) 271–279.
- [56] S. Roy, I. J. Cox, A maximum-flow formulation of the n-camera stereo correspondence problem, in: *Proc. IEEE Int. Conf. Comput. Vis.*, 1998, pp. 492–499.
- [57] A. V. Goldberg, R. E. Tarjan, A new approach to the maximum flow problem, *J. ACM* 35 (1988) 921–940.
- [58] Y. Weiss, W. T. Freeman, On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs, *IEEE Trans. Inf. Theory* 47 (2) (2001) 723–735.
- [59] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, D. J. Spiegelhalter, *Probabilistic Networks and Expert Systems*, Springer Verlag, 1999.
- [60] S. Benedetto, G. Montorsi, D. Divsalar, F. Pollara, Soft-output decoding algorithms in iterative decoding of turbo codes, in: *JPL TDA Prog. Rep.*, Vol. 42–124, 1995, pp. 63–87.
- [61] P. F. Felzenszwalb, D. P. Huttenlocher, Efficient belief propagation for early vis., *Int. J. Comput. Vis.* 70 (1) (2006) 41–54.
- [62] M. Björkman, D. Kragic, Active 3d segmentation through fixation of previously unseen objects, in: *Proc. British Mach. Vis. Conf.*, 2010, pp. 119.1–119.11.
- [63] M. Björkman, J.-O. Eklundh, Vision in the real world: Finding, attending and recognizing objects, *Int. J. Imaging Syst. Technol.* 16 (5) (2006) 189–209.
- [64] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis, *IEEE Trans. Pattern Anal. Mach. Intell.* 24 (5) (2002) 603–619.
- [65] D. Lowe, Object recognition from local scale-invariant features, in: *Proc. IEEE Int. Conf. Comput. Vis.*, Vol. 2, 1999, pp. 1150–1157.
- [66] A. V. Goldberg, R. E. Tarjan, A new approach to the maximum flow problem, in: *Proc. ACM Symp. Theory Comput.*, 1986, pp. 136–146.
- [67] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanski, S. Chennupati, P. Hammarlund, R. Singhal, P. Dubey, Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU, *SIGARCH Comput. Archit. News* 38 (3) (2010) 451–460.
- [68] V. Vineet, P. Narayanan, CUDA cuts: Fast graph cuts on the GPU, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2008, pp. 1–8.
- [69] A. Brunton, C. Shu, G. Roth, Belief propagation on the GPU for stereo vision, in: *Proc. Can. Conf. Comput. Robot Vis.*, 2006, pp. 76–81.
- [70] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, D. Nister, Real-time global stereo matching using hierarchical belief propagation, in: *Proc. Br. Mach. Vis. Conf.*, 2006, pp. 989–998.
- [71] Y. Xu, H. Chen, R. Klette, J. Liu, T. Vaudrey, Belief propagation implementation using CUDA on an NVidia GTX 280, *Adv. Artif. Intell.*, *Lect. Notes Comput. Sci.* 5866 (2009) 180–189.
- [72] Q. Yang, L. Wang, N. Ahuja, A constant-space belief propagation algorithm for stereo matching, in: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1458–1465.
- [73] C.-K. Liang, C.-C. Cheng, Y.-C. Lai, L.-G. Chen, H. H. Chen, Hardware-efficient belief propagation, *IEEE Trans. Circuits Syst. for Video Technol.* 21 (5) (2011) 525–537.
- [74] A. Vedaldi, B. Fulkerson, VLFeat: an open and portable library of computer vision algorithms, in: *Proc. Int. Conf. Multimed.*, 2010, pp. 1469–1472.
- [75] N. Bergström, M. Björkman, D. Kragic, Generating object hypotheses in natural scenes through human-robot interaction, in: *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 827–833.
- [76] M. Björkman, Y. Bekiroglu, V. Högman, D. Kragic, Enhancing visual perception of shape through tactile glances, in: *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013 (in press).
- [77] V. Högman, M. Björkman, D. Kragic, Interactive object classification using sensorimotor contingencies, in: *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013 (in press).