DRAFT COPY ONLY

Segmenting and Tracking Visual Entities

Dissertation

zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
"Doctor rerum naturalium"

der Georg-August-Universität Göttingen

im Promotionsprogramm
der Georg-August University School of Science (GAUSS)

vorgelegt von
Jérémie Papon aus Summit, NJ, USA

Göttingen, 2014

DRAFT COPY ONLY

Segmenting and Tracking Visual Entities

A DISSERTATION PRESENTED BY

JÉRÉMIE PAPON

TO THE FACULTY OF NATURAL SCIENCES AND MATHEMATICS

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
COMPUTER SCIENCE

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
GÖTTINGEN, GERMANY
MARCH 2014

Referentin/Referent: Prof. Dr. Florentin Wörgötter
Koreferentin/Koreferent: Prof. Dr. Dieter Hogrefe
Tag der mündlichen Prüfung: TBD

DRAFT COPY ONLY

Segmenting and Tracking Visual Entities

Abstract

Abstract text.

Contents

The work described in this thesis has appeared in the following publications:

Papon, J.; Kulvicius, T.; Aksoy, E.; Wörgötter, F., "Point Cloud Video Object Segmentation using a Persistent Supervoxel World-Model," *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov. 2013.

Papon, J.; Abramov, A.; Schoeler, M.; Wörgötter, F., "Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds," *Computer Vision and Pattern Recognition (CVPR) 2013*, June 2013.

Papon, J.; Abramov, A.; Wörgötter, F., "Occlusion Handling in Video Segmentation via Predictive Feedback," *European Conference on Computer Vision (ECCV) 2012*. Workshops and Demonstrations, Lecture Notes in Computer Science Volume 7585, 2012, pp 233-242.

Papon, J.; Abramov, A.; Aksoy, E.; Wörgötter, F., "A modular system architecture for online parallel vision pipelines," *Applications of Computer Vision (WACV) 2012*, pp.361-368, Jan. 2012.

The research leading to this thesis was supported with funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience and grant agreement no. 269959, Intellact.

List of Figures

Acknowledgments

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetuer. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.

Some Quote.

Quoteauthor Lastname

1 Introduction

There's something to be said for having a good opening line.

1.1 Problem Definition and Motivation

1.1.1 The Image Segmentation Problem

1.1.2 The Tracking Problem

Multi-target visual tracking (MTVT) and 6DoF pose estimation are crucial challenges for many applications such as visual surveillance, action recognition, and robotic imitation learning. In many such functions, visual tracking serves as the precursor to all further high-level inference, making robust tracking fundamental to the success of a large variety of intelligent systems. Related to the problem of visual tracking is segmentation, the task of grouping observations according to the entities which they contain. Video object segmentation (VOS) attempts to cluster pixels of video frames into segments which are both spatially and temporally coherent. While generally similar to MTVT, VOS goes a step beyond localizing tracked objects, in that it makes an association decision for each observed pixel; in addition to estimating overall state, it must re-estimate spatial extent every frame. In both VOS and MTVT there are two chief challenges that must be addressed: first, the data association problem, whereby noisy observations must be associated with the proper targets, and secondly, the occlusion problem, in which targets may become partially or fully obscured for a number of observations.

1.1.3 Segmentation In Sequential Frames

Video segmentation has the potential to be more accurate than single image segmentation, as it can take advantage of the temporal coherence of objects in space to infer information about the objects in a scene. Unfortunately, the addition of the temporal domain brings along new challenges as well; for instance that pixels which should be grouped across time may not be continuously visible, as in the case of partial or full occlusions. Additionally, the added dimension increases the computational complexity of the problem, making accurate segmentation a costly procedure. Temporal information also increases the exposure of the algorithm to noise, as each image frame is a separate noisy measurement. This adds a large amount of uncertainty to the problem, since measured values (i.e., of color) for an object can show significant variation over time.

While MTVT and VOS are clearly related, they traditionally have been considered separate areas of research. In this work, we unify them by taking a mature tracking approach, particle filtering, and apply it to tracking supervoxels (3d segments) from a recent 3d segmentation technique [**?**], Voxel Cloud Connectivity Segmentation (VCCS). To make this possible, we extend the concept of VCCS to dynamic scenes by maintaining a world-octree supervoxel model which lets objects persist indefinitely through occlusions. Additionally, we use a novel global energy function to associate observations to predictions, and thereby extract accurate object segmentations (even for fully occluded objects) from tracker predictions.

1.1.4 Perception as Accumulation of a 3D World Model

1.2 State of the Art

1.2.1 Segmentation and Superpixels

Segmentation of scenes into objects remains one of the most challenging topics of computer vision despite decades of research. To address this, recent methods often use hierarchies which create a rank order that build bottom-up from small localized superpixels to large-scale regions [**? ? ?**]. As an alternative, researchers have also pursued strictly top-down approaches. These began with coarse segmentations using multiscale sliding window detectors [**?**], later progressing to finer grained segmentations and detections based on object parts [**? ?**]. These two avenues of research led naturally to methods which *combine* bottom-up hierarchy building with top-down object- and part-detectors [**? ? ?**]. While these approaches have yielded quite good results even on complex, varied data sets, they have lost much of the generality of learning-free approaches. In general the most powerful methods to-date use trained classifiers for segmentation [**? ?**]. This means they cannot be applied to arbitrary unknown scenes without being retrained, requiring the acquisition of a new data-set tailored to each test environment.

In this work we investigate model- and learning-free bottom-up segmentation of 3D point clouds captured with RGB-D cameras. In particular, we focus on the partitioning of cluttered scenes into basic *object parts* without the need for training data. As inspiration for a general rule for breaking scenes into elemental parts, we look to psychophysical studies, mostly performed on 2D images, which suggest that

the transition between convex and concave image parts might be indicative of the separation between objects and/or their parts [? ? ? ? ? ?]. While this feature has been used in machine vision to some degree [? ? ? ? ?] success has remained limited and more recent studies were forced to combine this feature with additional, often very complex feature constellations to achieve good scene partitioning [? ? ?]. It, thus, appears that direct transfer from 2D to 3D of the conventional, geometrically-defined convex-concave transition criterion shown in Fig. ?? A is not possible for achieving good 3D-segmentation. This puzzling observation can best be understood by ways of an example.

1.2.2 Multi-Target Tracking

Bayesian predictive filtering is a broad, well-established field in target tracking applications [? ? ?]. While effective for tracking, these methods generally depend on fixed models with a small dimensional state-space, and are unable to deal with the high-dimensionality of VOS. Nevertheless, there exist a few methods which attempt to apply tracking methodologies to the VOS problem. A recent method [?] uses graph cuts to extract segmentations, and a dynamical model to form predictions which guide successive segmentations. Unfortunately, this method formally models visible and occluded parts of the tracked objects, and so does not scale well with an increasing number of objects, and thus is better suited to extracting the silhouettes of a few objects than performing a full segmentation. Other methods, such as [?], are severely limited in that they require pre- computed models which are calibrated to a ground plane in order to resolve occlusions. The previously discussed work of Brendel and Todorovic [?] also combines tracking and segmentation, but as mentioned earlier, it is an off-line method (performing smoothing rather than filtering) and thus cannot be used in many applications.

While MTVT remains an unsolved problem, single target visual tracking (STVT) is a fairly well-studied problem, with many mature approaches [? ?]. Additionally, recent work has progressed in estimating pose (in addition to tracks) for single targets, for example [?] uses a particle filter to track 6-DoF pose of arbitrary objects in point clouds. Recent work in MTVT [?] successfully tracks multiple objects using a segmentation and association approach and adaptive 3D appearance models, but is limited by the need to align model point clouds to the observed data every frame. This precludes it from handling occlusions, as once a target is no longer observed, its track must be terminated.

1.2.3 Video Object Segmentation

There are many existing video object segmentation (VOS) methods, which can be classified based on three parameters; whether they are on- or off-line, whether they are dense or sparse, and whether or not they are supervised. We can reduce the comparison-space of related work by comparing only with algorithms which have the same three parameters as this work - on-line processing (the algorithm may only use past data), dense segmentation (every pixel is assigned to a spatio-temporal cluster), and unsupervised operation. Four state-of-the-art segmentation algorithms meet these requirements: Mean-shift video segmentation (MSVS) [?], Multiple hypothesis video segmentation (MHVS) from superpixel flows [?], Propagation, validation, and aggregation (PVA) of a preceding graph [?], and Matching

images under unstable segmentations [**?**]. Of these methods, none are able to handle full occlusions; in fact only MHVS considers occlusions, and it is only able to handle partial occlusions for a few frames, and does not consider full occlusions. Even state of the art off-line methods such as that of Brendel and Todorovic [**?**] only handle partial occlusions, claiming that "complete occlusions ... require higher-level reasoning".

Multiple hypothesis video segmentation (MHVS) from superpixel flows [**?**] provides dense online unsupervised video segmentations, but is only able to handle partial occlusions for a few frames, and does not consider full occlusions. There also has been much recent work in VOS specifically addressing the problem of segmenting foreground from background [**? ?**]. While these works have been to shown to perform very well in their task, they only solve the single target case, as they do not need to resolve the multiple association problem.

In [**?**] Papadakis and Bugeau use a dynamical model to guide successive segmentations, along with an energy function minimized using graph cuts to solve the label association problem. They formally model visible and occluded regions of tracked objects, tracking them as distinct parts. While they do consider occlusions, they do not maintain a world model, and as such their methodology must fail under complete occlusions.

1.3 Outline and Contributions

Some Quote.

Quoteauthor Lastname

2

Video Segmentation by Relaxation of Tracked Masks

There's something to be said for having a good opening line.

Before proceeding to discuss the principal elements of the proposed algorithm in detail, we shall first give a brief overview of the algorithm (depicted in Figure **??**). The algorithm begins by performing an initial segmentation on the first frame $\mathbf{F}_{t_\circ}$ to generate an initial set of labels $\mathbf{S}_{t_\circ}$. This segmentation result is used to generate initial sets of particles, each of which contains a map of the object. Color histogram features are then generated for each particle (as in [**?**]) and particles are initialized with randomly distributed initial velocities. Thus each object k at initial time $t_\circ$ from the segmentation is specified by a set of N_k particles $\mathbf{X}_{t_\circ}^{k,1:N_k}$, each of which contains a representation of the object, specified by a pixel existence map $\mathbf{M}$, a reference color histogram $\hat{q}$ calculated from $\mathbf{F}_{t_\circ} \cap \mathbf{M}_{t_\circ}^{k,n}$, a position shift vector $mathbfp_{t_\circ}$, and a velocity vector $\mathbf{v}_{\mathbf{t}_\circ}$.

The particles are then propagated in time independently, shifting their existence maps to new regions of the image. These shifted maps are used to generate new measured color histograms $q_{t_\circ+1}$ from frame $\mathbf{F}_{t_\circ+1}$, which are evaluated to determine particle weights. The set of particles for object k, $\mathbf{X}_{t_\circ+1}^{k,1:N_k}$, is then combined to create an overall object pixel likelihood map $\hat{\mathbf{M}}_{t_\circ+1}^{k}$. The pixel likelihood maps for all objects are then used to generate a label association likelihood map $\hat{\mathbf{L}}_{t_\circ+1}$, for frame $\mathbf{F}_{t_\circ+1}$, where each pixel in the map is a PDF specifying the probability of the pixel belonging to each object k.

The label association likelihood map is then sampled using a per-pixel selection procedure (as described in Section **??**) to generate a candidate label image, $\tilde{\mathbf{S}}_{t_\circ+1}$. This is used as the initialization for the Metropolis-Hastings algorithm with annealing of Abramov et al. [**?**], which updates the labels iteratively until an equilibrium segmented state is reached. The segmentation result, $\mathbf{S}_{t_\circ+1}$ is subsequently

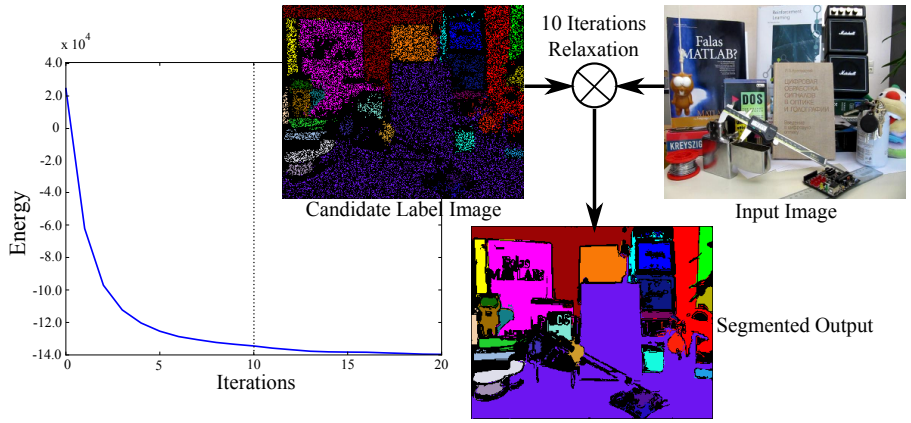

Figure 2.1.1: The relaxation process causes the energy of the label image to converge after few iterations (outcome after 10 iterations shown here). This results in efficient calculation of an accurate and temporally coherent segmentation.

used to update the set of particles via three mechanisms; birth, decay, and repopulation. Birth is used for new labels k in the segmentation output, and consists of initializing a set of particles $\mathbf{X}^k$ for the new label. Decay occurs when a label is not found in the segmentation output, and consists of killing a number of the particles of the missing label, determined by the decay rate λ_d. The most commonly occurring mechanism, repopulation, is a key component of the algorithm, and occurs for all previously existing object labels which are found in $\mathbf{S}_{t_o+1}$. Repopulation rejuvenates the set of particles for an object by replacing a number (determined by the repopulation rate λ_r) of particles in the set with particles generated from $\mathbf{S}_{t_o+1}$ and $\mathbf{F}_{t_o+1}$. That is, the map $\mathbf{M}$ and color histogram $\hat{q}_k$ of the repopulating particles are initialized using the new segmentation output and image frame.

2.1 Segmentation using Superparamagnetic Clustering

To adjust the candidate label image $\tilde{\mathbf{S}}_t$ to the current frame $\mathbf{F}_t$, we use a real-time image segmentation algorithm based on superparamagnetic clustering of data [**?**]. The method of superparamagnetic clustering represents an input image being segmented by a Potts model, with pixel color vectors arranged on the sites of a two-dimensional (2D) lattice, where each pixel is featured by an additional variable, called a spin. This allows the segmentation problem to be formulated as a minimization problem which seeks to find the equilibrium states of the energy function in the superparamagnetic phase. In this equilibrium state regions of aligned spins coexist and correspond to a natural partition of the image data [**?**]. Since every found segment carries a spin variable which is unique within the whole image, the terms *spin* and *label* are equivalent here. The equilibrium states are found by the use of the highly parallel Metropolis algorithm with a simulated annealing, called *relaxation process*, implemented on a Graphics Processing Unit (GPU) [**?**]. In this work, the relaxation process adjusts the predicted candidate label image to the current frame.

Superparamagnetic clustering of data was chosen due to its flexibility in allowing the use of any initialization state; there are no particular requirements to the initial states of spin variables. The closer the

initial states are to the equilibrium, the less time the Metropolis algorithm needs to converge. This property makes it possible to achieve temporal coherency in the segmentation of temporally adjacent frames by using the sparse label configuration taken from the candidate label image for the spin initialization of the current frame. A final (dense) segmentation result is obtained within a small number of Metropolis updates. Conventional segmentation methods do not generally have this property and cannot turn a sparse segmentation prediction into dense final segments which preserve temporal coherence. Moreover, since the method can directly use sparse predictions as the seed of the segmentation kernel, we can avoid the costly block-matching procedure required to find label correspondences in other work, such as in Brendel and Todorovic [**?**] or Hedau et al. [**?**]. Figure 2.1.1 illustrates the relaxation process, and the convergence of energy after a small number of iterations.

The primary goal of the proposed algorithm is to use predictions from Bayesian filtering to inform segmentation of higher-level temporal object correspondences. It is well known that sequential Bayesian estimation methods perform well in difficult tracking scenarios [**?**], and, under the Markov assumption, are computationally less demanding than video segmentation techniques such as MHVS [**?**], which consider many prior frames. Particle filtering is one such method which has been shown to approximate the optimal tracking solution well, even in complex multi-target scenarios with strong nonlinearities [**?** **?** **?**]. In this section we will briefly review the basic principles of sequential Bayesian estimation and particle filtering, and describe how they are used to predict pixel associations in order to seed the segmentation.

2.2 Tracking Object Masks

2.2.1 Sequential Bayesian Estimation

Sequential Bayesian estimation uses a state space representation, in which a state vector $\mathbf{x}_t$ describes the hidden state of a dynamic system. Bayesian estimation attempts to determine the posterior distribution of the state given all prior observations $\mathbf{z}$, i.e., $p(\mathbf{x}_t|\mathbf{z}_{1:t})$. This is accomplished using a two step recursion which first generates a hypothesis of the current state conditioned on the previous state and then performs a Bayes update using the new observation. These steps are known as the prediction and filtering steps, respectively.

The prediction step estimates the current distribution given all prior observations, or

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})d\mathbf{x}_{t-1}. \tag{2.1}$$

This prediction requires the specification of a stochastic *dynamic model*

$$\mathbf{x}_t = f_t(\mathbf{x}_{t-1}, \mathbf{v}_t), \tag{2.2}$$

where $\mathbf{v}_t$ is the process noise, which characterizes the state transition density $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. The dynamic model takes advantage of knowledge of the system to generate reliable predictions of how the state

evolves.

The filtering step uses Bayes rule to update the predicted density by conditioning it on the new observation $\mathbf{z}_t$:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})}. \tag{2.3}$$

This requires the specification of an observation, or measurement, model

$$\mathbf{z}_t = h_t(\mathbf{x}_t, \mathbf{w}_t), \tag{2.4}$$

where $\mathbf{w}_t$ is the measurement noise, which characterizes the observation density $p(\mathbf{z}_t|\mathbf{x}_t)$. Once the filtered, or posterior distribution is determined, an estimate of the state can be made using a variety of techniques (e.g., MAP, mean-shift).

Dynamic Model

In our method, the state of a particle consists of four elements; the pixel existence map $\mathbf{M}$, a reference color histogram $\hat{q}$, a position shift vector $\mathbf{p}$, and a velocity vector $\mathbf{v}_t$. Of these, only the position shift and velocity evolve over time, so we adopt the state vector

$$\mathbf{x}_t = [p_x v_x p_y v_y]^T, \tag{2.5}$$

where (p_x, p_y) denotes the accumulated shift of the pixel existence map in the image plane, and (v_x, v_y) the map velocity in the image plane. Motion is modeled using a constant velocity model in discrete time with uniform sampling period T, giving the dynamic model

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{v}_t, \tag{2.6}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

and noise $\mathbf{v}_t$ is assumed to be zero mean Gaussian with fixed covariance.

Measurement Model

In our method measurements are taken by calculating a color histogram, q_t for the region lying within the shifted pixel existence map $\mathbf{M}$. That is, for particle n of object k,

$$q_t^{k,n} = hist(\mathbf{F}_t \cap \mathbf{M}_t^{k,n}) \tag{2.8}$$

Color histograms are three dimensional, with 8 bins for each of the color components hue, saturation, and value. As in [**?**], a Gaussian density is used for the observation density $p(\mathbf{z}_t|\mathbf{x}_t)$, that is

$$p(\mathbf{z}_t|\mathbf{x}_t) = \frac{1}{\sqrt{2\pi}\sigma} \exp -\frac{\Delta(\hat{q}, q_t)^2}{2\sigma^2}, \tag{2.9}$$

where $\Delta(\hat{q}, q_t)$ is the Bhattacharyya distance (as proposed in [**?**]) between the reference histogram $\hat{q}$ for the particle and the measured histogram q_t for time t. The Bhattacharyya distance is a standard measure of similarity between discrete probability distributions, and is defined as

$$\Delta(\hat{q}, q_t) = \sqrt{1 - \sum \sqrt{\hat{q} q_t}}. \tag{2.10}$$

2.2.2 Parallel Particle Filters

Except in special cases (e.g., Kalman Filter), closed-form solutions to the integrals (2.1) and (2.3) are not available. The particle filter is a Monte-Carlo method designed to approximate the posterior distribution with a weighted set of random samples. There are many excellent descriptions of the mechanics of particle filtering available (such as [**?**]), so we shall avoid presenting them here, and proceed directly to presenting the details of our algorithm.

The predictive portion of the method uses multiple Sequential Importance Resampling (SIR) filters in parallel to track multiple targets (labels) simultaneously. At this stage in the algorithm targets are assumed independent and interaction between labels is therefore not considered (interaction is accounted for later, as described in Section 2.3). Particles are first propagated using the constant velocity dynamics model, and their predicted existence maps $\mathbf{M}^{k,n}$ are used to generate a measured histogram, q_t. Particles are weighted based on (2.9), and then normalized as a set for each label k. Systematic resampling is used to prevent particle degeneracy, due to its speed and good empirical performance [**?**].

The resulting distributions from the weighting procedure are used to generate object pixel likelihood maps for each label, $\hat{\mathbf{M}}^k_{t+1}$, which are then combined into the label association likelihood map $\hat{\mathbf{L}}_t$, as described in Section 2.3. $\hat{\mathbf{L}}_t$ can then be relaxed to produce a final segmented output, $\mathbf{S}_t$.

2.2.3 Particle Birth, Repopulation, & Decay.

One key improvement of the proposed algorithm over prior particle filtering methods is its use of the segmentation result $\mathbf{S}_t$ to update the particle sets. This allows the creation of new targets, adaptation to changing target appearance, and gradual elimination of targets which are no longer observed. This is accomplished via three mechanisms, which we term, respectively, birth, repopulation, and decay.

Birth occurs when a label which has not existed previously is found in the segmentation output $\mathbf{S}_t$, or more formally $\{k \notin \mathbf{S}_{1:t-1}, k \in \mathbf{S}_t\}$. It consists of generating a set of particles $\mathbf{X}^k$ for the new label using $\mathbf{S}_t$ to initialize an existence map $\mathbf{M}^k_t$ and $\{\mathbf{F}_t \cap \mathbf{M}^k_t\}$ to calculate a reference color histogram $\hat{q}^k_t$.

Repopulation is a key component of the algorithm, as it allows the pixel likelihood map for an object, $\hat{\mathbf{M}}^k$, to adapt over time to the changing appearance of the object. Every iteration, all previously

existing object labels which are found in $\mathbf{S}_t$ are repopulated by replacing some particles in the set with particles generated from $\mathbf{S}_t$ and $\mathbf{F}_t$. Particles are chosen for replacement using stratified sampling, at a rate specified by parameter λ_r. The repopulation mechanism gradually modifies the object "model" through the addition of particles which have an updated existence map and color histogram (coming from the segmentation result). We use the term model here loosely, since there is in actuality no explicit model for any of the objects - merely a pixel likelihood map generated at each time step from the objects constituent particles and the current image frame.

Stratified replacement and relatively low repopulation rates are used to help keep the influence of erroneous hypotheses to a minimum, but as with any adaptive method, they can occasionally lead the tracker astray. Replacement of particles, rather than updating of a central model, helps to reduce this problem, since a few erroneous particles will not completely derail the algorithm. Nevertheless, future work should investigate strategies that allow pruning of unlikely hypotheses without negatively affecting occlusion handling.

Decay occurs when a label is not found in the segmentation output, $k \notin \mathbf{S}_t$. Particles are selected from k using random sampling, at a rate determined by the decay rate λ_d, and are pruned; they are no longer considered when filtering k. This reduces the number of active particles for the label in the next iteration, N^k_{t+1}, by approximately $\lambda_d N^k_t$. If the number of active particles for a label falls below a certain threshold, N_{min}, then the set of particles for the label is deleted, and the object is no longer tracked. If a label which was being decayed is observed again, i.e., $\{k \notin \mathbf{S}_{t-1}, k \in \mathbf{S}_t\}$, then the label is revived by replacing particles which had been killed with new particles, which are initialized as in the repopulation step.

2.3 Extracting a Dense Image Labeling

The middle portion of Figure **??** depicts how the candidate label image, $\tilde{\mathbf{S}}_t$, is generated. The candidate label image is a summary of the accumulated knowledge of the particle filters; it is a prediction of what the segmented scene should look like. That is to say, it is a pixel-wise realization of the label association likelihood map $\hat{\mathbf{L}}_t$, which is constructed by combining the object pixel likelihood maps (which approximate the posteriors of the particle sets). $\tilde{\mathbf{S}}_t$ is the seed of the segmentation kernel, which uses pixel values from $\mathbf{F}_t$ to perform the relaxation process and generate a dense label image. In this section we will describe the process of generating the object pixel and label association likelihood maps, and then explain how the predictive loop allows occlusion handling without explicit object relationship or depth modeling.

2.3.1 Object Pixel Likelihood Maps.

The object pixel likelihood map for a particular object k is the weighted sum of the pixel existence maps of all of its labels,

$$\hat{\mathbf{M}}^k_t = \sum_{n=1}^{N_k} w^{k,n}_t \mathbf{M}^{k,n}. \tag{2.11}$$

Because the weights have been normalized, the pixel values in $\hat{\mathbf{M}}_t^k$ will be in the range $[0, 1]$. High pixel values will occur in regions which are present in the existence maps of highly weighted particles, or alternatively, are present in many particles with average weight.

2.3.2 Label Association Likelihood Map.

The label association likelihood map $\hat{\mathbf{L}}_t$ is a combination of all the object pixel likelihood maps, such that each pixel contains a discrete probability distribution giving the likelihood of the pixel belonging to a certain label. Additionally, a likelihood, $p_\circ$, for the pixel belonging to no label is inserted to allow pixels where no label has high likelihood to remain unlabeled in $\tilde{\mathbf{S}}_t$. More formally,

$$\hat{\mathbf{L}}_t = \bigcup_{n=1}^{K} \hat{\mathbf{M}}_t^n + p_\circ. \tag{2.12}$$

Each pixel of $\hat{\mathbf{L}}_t$ is then normalized, such that the sum of the discrete probabilities sums to one. The candidate label image can then be generated by taking a realization of $\hat{\mathbf{L}}_t$ to select pixel label values. Examples of the result of this process, $\tilde{\mathbf{S}}_t$, can be seen in Figures **??** and 2.5.1.

2.4 Occlusion Handling.

2.4.1 Color Distribution Interactions

Occlusion relationships are handled naturally, since foreground objects will tend to have a strong peak in their weight distribution, corresponding to those particles which align properly with $\mathbf{F}_t$. Objects they occlude will have a flat particle weight distribution, since there will exist no shifted existence map which contains a color distribution which matches the reference histogram. This is due to the fact that the occluding objects and objects surrounding the occluded object have color distributions which differ from the occluded object. Let us assume foreground object j is contained by occluded object k, that is

$$\mathbf{M}_t^{j,n} \subset \mathbf{M}_t^{k,n}. \tag{2.13}$$

We also assume that the number of particles is sufficiently large such that

$$\exists\, \mathbf{M}_t^{j,n} \in \mathbf{M}_t^j : hist(\mathbf{F}_t \cap \mathbf{M}_t^{j,n}) \approx \hat{q}^{j,n}. \tag{2.14}$$

If $hist(\mathbf{F}_t \cap \mathbf{M}^{k,n}) \neq hist(\mathbf{F}_t \cap \mathbf{M}^{j,n})$, that is, the objects have different color distributions, then from (2.13) and (2.14), it follows that [1]

$$\nexists\, \mathbf{M}_t^{k,n} \in \mathbf{M}_t^k : hist(\mathbf{F}_t \cap \mathbf{M}_t^{k,n}) \approx \hat{q}^{k,n} \tag{2.15}$$

[1] This also assumes that the areas surrounding the occluded object also have different color distributions.

and therefore that

$$min_{1:N_j}\{\Delta(\hat{q}^{j,n}, hist(\mathbf{F}_t \cap \mathbf{M}_t^{j,n}))\} < min_{1:N_k}\{\Delta(\hat{q}^{k,n}, hist(\mathbf{F}_t \cap \mathbf{M}_t^{k,n}))\} \quad (2.16)$$

and thus

$$max_{1:N_j}\{w_t^{j,n}\} > max_{1:N_k}\{w_t^{k,n}\}. \quad (2.17)$$

This means that in the label association likelihood map $\hat{\mathbf{L}}_t$, the occluding object will have a higher likelihood then the occluded. The candidate label image, $\tilde{\mathbf{S}}_t$ will therefore tend to favor occluding object labels, which will dominate the occluded object label during the segmentation relaxation process.

2.4.2 Depth Mask Reasoning

2.5 Experimental Results

In order to evaluate performance, we compare our method to the state of the art on several challenging video tracking benchmark sequences which are available online[2]. It should be noted that, as opposed to the other tracking algorithms, we do not pre-select a region to track, and track fully deforming object masks (rather than a rectangle). Additionally, we employ no learned or a-priori specified models, use 50 particles per label, and only have two parameters; the repopulation and decay rates λ_r and λ_d, which were both held constant at 0.05 throughout testing. Results are compared to the PROST [**?**], MilTrack [**?**], FragTrack [**?**], and ORF [**?**] tracking algorithms. Further details concerning the parameters used for the above algorithms in the benchmarking can be found in [**?**].

We shall not evaluate the visual quality of segmentation results here for a couple of reasons. First, detailed evaluation of the visual quality of super-paramagnetic clustering has been presented in [**?**] in greater detail than space would permit here. The segmentation results obtained from this work do not differ significantly from the above results, with the exception of labels having continuity through occlusions. Secondly, it is directly acknowledged in other VOS work that the methods fail under partial [**? ?**] or full [**? ?**] occlusions. As such, comparing performance to other VOS methods is somewhat unreasonable. Rather, the better comparison is to the state of the art in tracking methods, which attempt to handle full and partial occlusions.

In order to compare with the other methods, we needed to output a tracking rectangle for each frame. To do this, once the sequence was segmented, we found the segment which corresponded to the object to track in the first frame, and then took the bounding-box which contained it in each frame as the tracking rectangle. This bounding-box was then compared to ground-truth using two measures; Euclidean distance and the PASCAL-challenge based score proposed in [**?**]. The latter compares the area of intersection of the ground truth and tracked box with the union of the same. When this is greater than 0.5, the object is considered successfully tracked. Table 2.5.1 gives our results, as well as the results for

[2] http://www.GPU4Vision.org

Table 2.5.1: PROST dataset benchmark results. Top and bottom tables are average pixel error and PASCAL based scores, respectively

Sequence	PROST	MIL	Frag	ORF	HybridPF
Lemming	25.1	14.9	82.8	166.3	19.8
Box	13.0	104.6	57.4	145.4	114.1
Liquor	21.5	165.1	30.7	67.3	25.5
Board	39.0	51.2	90.1	154.5	30.9
Lemming	70.5	83.6	54.9	17.2	73.9
Box	90.6	24.5	61.4	28.3	7.5
Liquor	85.4	20.6	79.9	53.6	54.2
Board	75.0	67.9	67.9	10.0	71.4

the other methods.

Testing showed that, when certain assumptions hold, our algorithm performs on par with, and in some cases outperforms, state of the art tracking algorithms. This is the case for the *liquor*, *lemming*, and *board* sequences. In the *lemming* sequence, frames of which are shown in Figure 2.5.1, our algorithm outperforms the other methods in cases of occlusion, especially when the tracked object is fully occluded. While other methods offer false positives and erroneous tracks, our method decays the label for the object and avoids proposing incorrect tracking solutions. In the *liquor* sequence, our algorithm adapts to the changing appearance (size, shape, and color) of the tracked bottle, allowing it to maintain performance on par with the other algorithms, in spite of the difficulties of segmenting transparent objects. In the *board* sequence, our method successfully adapts to the rapidly changing appearance of the tracked board as it rotates, allowing it to maintain an accurate track and outperform the other methods.

In addition to showing the strengths of our method, a weakness was also highlighted by the benchmark sequences. The *box* sequence demonstrated the limitations of using unsupervised color-based segmentation to initialize the objects to track. In the sequence, the object to track contains strong color differences, which are segmented into different initial regions. As the object moves around, the particles for these regions are attracted to other objects it passes over which have similar color.

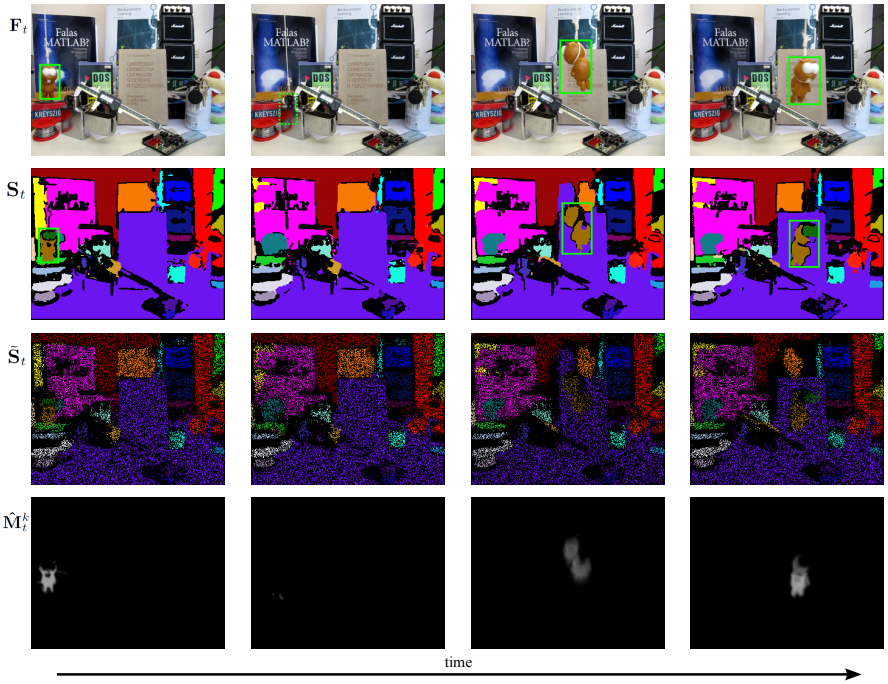

Figure 2.5.1: Output frames from the *lemming* sequence, in which a target is completely occluded (for $\sim$ 20 frames, second column) and changes significantly in appearance. The object which is tracked for comparison to other algorithms is highlighted with a green box. $\mathbf{F}_t$-Original frames from movie. $\mathbf{S}_t$-The output of the segmentation algorithm. $\tilde{\mathbf{S}}_t$-The candidate label image constructed by taking a random draw from $\hat{\mathbf{L}}_t$, the label association likelihood map. $\hat{\mathbf{M}}_t^k$-The overall object pixel likelihood map for the lemming label, created by combining the set of particles for the label. Intensity represents the sum of the normalized weights of the set of particles.

Some Quote.

Quoteauthor Lastname

3 Patch-based Perceptual World Model

There's something to be said for having a good opening line.

3.1 Octree Adjacency Graph

Adjacency is a key element of the proposed method, as it ensures that supervoxels do not flow across object boundaries which are disconnected in space. There are three definitions of adjacency in a voxelized 3D space; 6-,18-, or 26-adjacent. These share a face, faces or edges, and faces, edges, or vertices, respectively. In this work, whenever we refer to adjacent voxels, we are speaking of 26-adjacency.

As a preliminary step, we construct the adjacency graph for the voxel-cloud. This can be done efficiently by searching the voxel kd-tree, as for a given voxel, the centers of all 26-adjacent voxels are contained within $\sqrt{3} * R_{voxel}$. R_{voxel} specifies the voxel resolution which will be used for the segmentation (for clarity, we shall simply refer to discrete elements at this resolution as voxels). The adjacency graph thus constructed is used extensively throughout the rest of the algorithm.

3.2 Spatial Cluster Seeding

The algorithm begins by selecting a number of seed points which will be used to initialize the supervoxels. In order to do this, we first divide the space into a voxelized grid with a chosen resolution R_{seed}, which is significantly higher than R_{voxel}. The effect of increasing the seed resolution R_{seed} can be seen in Figure 3.2.2. Initial candidates for seeding are chosen by selecting the voxel in the cloud nearest to the

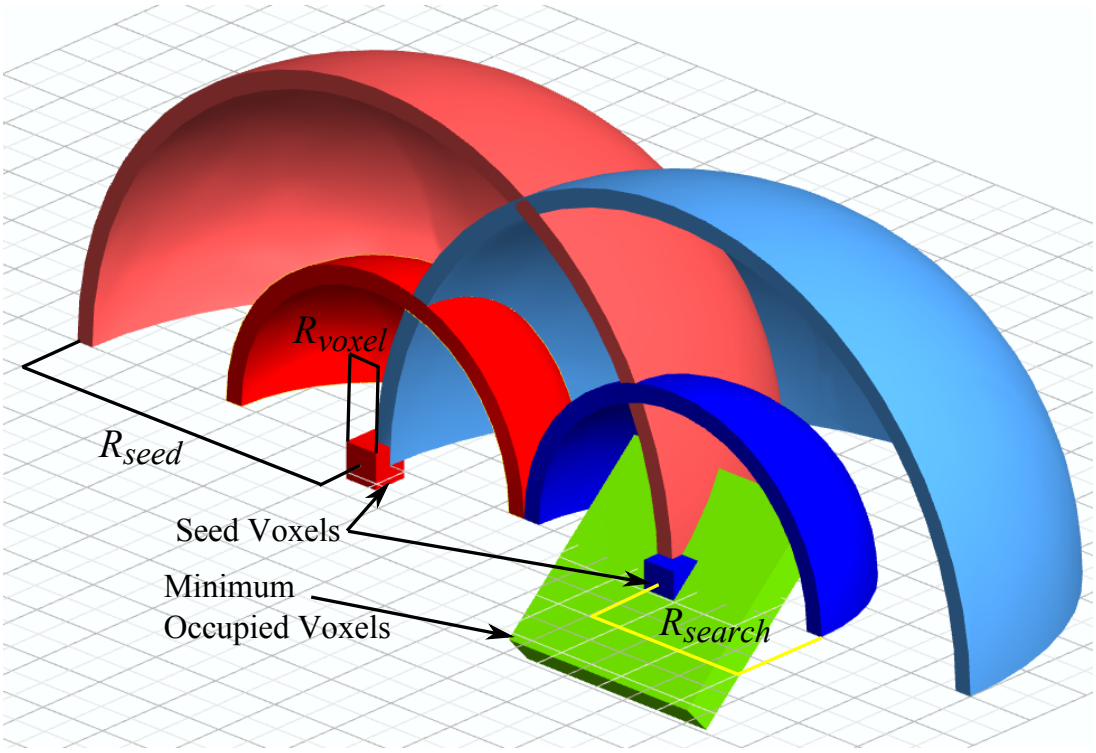

Figure 3.2.1: Seeding parameters and filtering criteria. R_{seed} determines the distance between supervoxels, while R_{voxel} determines the resolution to which the cloud is quantized. R_{search} is used to determine if there are a sufficient number of occupied voxels to necessitate a seed.

center of each occupied seeding voxel.

Once we have candidates for seeding, we must filter out seeds caused by noise in the depth image. This means that we must remove seeds which are points isolated in space (which are likely due to noise), while leaving those which exist on surfaces. To do this, we establish a small search radius R_{search} around each seed, and delete seeds which do not have at least as many voxels as would be occupied by a planar surface intersecting with half of the search volume (this is shown by the green plane in Figure 3.2.1). Once filtered, we shift the remaining seeds to the connected voxel within the search volume which has the smallest gradient in the search volume. Gradient is computed as

$$G(i) = \sum_{k \in V_{adj}} \frac{\| V(i) - V(k) \|_{CIELab}}{N_{adj}}; \tag{3.1}$$

we use sum of distances in CIELAB space from neighboring voxels, requiring us to normalize the gradient measure by number of connected adjacent voxels N_{adj}. Figure 3.2.1 gives an overview of the different distances and parameters involved in seeding.

Once the seed voxels have been selected, we initialize the supervoxel feature vector by finding the center (in feature space) of the seed voxel and connected neighbors within 2 voxels.

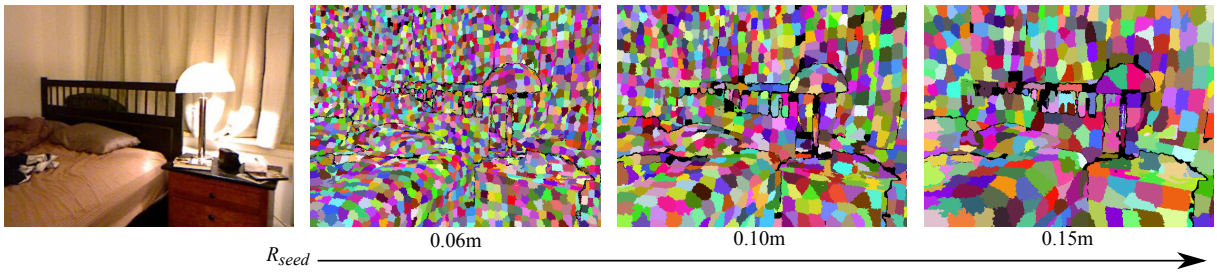

Figure 3.2.2: Image segmented using VCCS with seed resolutions of 0.1, 0.15 and 0.2 meters.

3.3 Cluster Features and Distance

VCCS supervoxels are clusters in a 39 dimensional space, given as

$$\mathbf{F} = [x, y, z, L, a, b, \text{FPFH}_{1..33}], \tag{3.2}$$

where x, y, z are spatial coordinates, L, a, b are color in CIELab space, and $\text{FPFH}_{1..33}$ are the 33 elements of Fast Point Feature Histograms (FPFH), a local geometrical feature proposed by Rusu *et al* [**?**]. *FPFH are pose-invariant features which describe the local surface model properties of points using combinations of their k nearest neighbors. They are an extension of the older Point Feature Histograms optimized for speed, and have a computational complexity of* $O(n \cdot k)$.

In order to calculate distances in this space, we must first normalize the spatial component, as distances, and thus their relative importance, will vary depending on the seed resolution R_{seed}. *Similar to the work of Achanta* et al, [**?**] we have limited the search space for each cluster so that it ends at the neighboring cluster centers. This means that we can normalize our spatial distance D_s using the maximally distant point considered for clustering, which will lie at a distance of $\sqrt{3}R_{seed}$. Color distance D_c, is the euclidean distance in CIELab space, normalized by a constant m. Distance in FPFH space, D_f, is calculated using the Histogram Intersection Kernel [**?**]. This leads us to a equation for normalized distance D:

$$D = \sqrt{\frac{\lambda D_c^2}{m^2} + \frac{\mu D_s^2}{3R_{seed}^2} + \varepsilon D_{HiK}^2}, \tag{3.3}$$

where λ, μ, and ε control the influence of color, spatial distance, and geometric similarity, respectively, in the clustering. In practice we keep the spatial distance constant relative to the other two so that supervoxels occupy a relatively spherical space, but this is not strictly necessary. For the experiments in this paper we have color weighted equally with geometric similarity.

3.4 Flow Constrained Region Growing

Assigning voxels to supervoxels is done iteratively, using a local k-means clustering related to [**?** **?**], with the significant difference that we consider connectivity and flow when assigning pixels to a cluster.

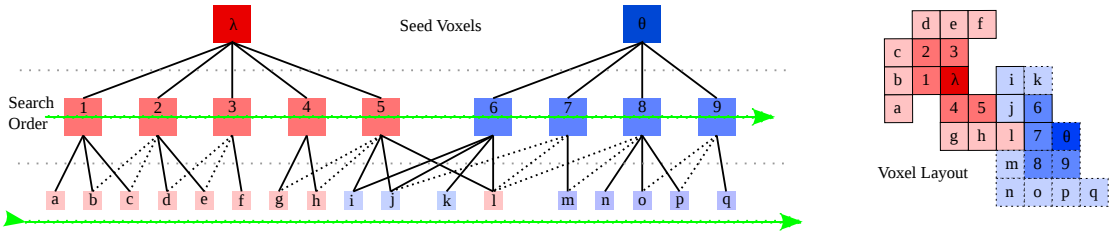

Figure 3.4.1: Search order for the flow constrained clustering algorithm (shown in 2D for clarity). Dotted edges in the adjacency graph are not searched, as the nodes have already been added to the search queue.

The general process is as follows: beginning at the voxel nearest the cluster center, we flow outward to adjacent voxels and compute the distance from each of these to the supervoxel center using Equation 3.3. If the distance is the smallest this voxel has seen, its label is set, and using the adjacency graph, we add its neighbors which are further from the center to our search queue for this label. We then proceed to the next supervoxel, so that each level outwards from the center is considered at the same time for all supervoxels. We proceed iteratively outwards until we have reached the edge of the search volume for each supervoxel (or have no more neighbors to check).

This amounts to a breadth-first search of the adjacency graph, where we check the same level for all supervoxels before we proceed down the graphs in depth. Importantly, we avoid edges to adjacent voxels which we have already checked this iteration. The search concludes for a supervoxel when we have reached all the leaf nodes of its adjacency graph or none of the nodes searched in the current level were set to its label. This search procedure, illustrated in Figure 3.4.1, has two important advantages over existing methods:

1. Supervoxel labels cannot cross over object boundaries that are not actually touching in 3D space, since we only consider adjacent voxels, and

2. Supervoxel labels will tend to be continuous in 3D space, since labels flow outward from the center of each supervoxel, expanding in space at the same rate.

Once the search of all supervoxel adjacency graphs has concluded, we update the centers of each supervoxel cluster by taking the mean of all its constituents. This is done iteratively; either until the cluster centers stabilize, or for a fixed number of iterations. For this work we found that the supervoxels were stable within a few iterations, and so have simply used five iterations for all presented results.

3.5 Depth Adaptive Grid

So far we have described the main algorithm for segmentation. Next we will introduce a generally applicable depth transform, which improves this specific analysis but can be used for all types of image analyses using algorithms with a fixed scale of observation on RGB-D data from a single RGB-D camera. In our case, we address shortcomings of the voxel grid VCCS is based on.

It is evident that observations from a single RGB-D camera have a significant drawback - the point density, and thus available detail of the scene geometry, falls rapidly with increasing distance from the

camera. In addition, the levels of both quantization and noise grow quadratically [**?** **?**], leading to a further degradation in the quality of geometric features. This change in point density with depth creates a tradeoff between capturing small-scale detail in the foreground (using small voxels) and avoiding noise in the background (using large voxels). This is a general problem which occurs in all algorithms working with a fixed scale (for example a radius search) on point clouds created from a single view.

We propose to compensate for the loss of point density and quantization with increasing depth z by transforming the points into a skewed space using the transformation $T : (x, y, z) \rightarrow (x', y', z')$ with

$$x' = x/z, \quad y' = y/z, \quad z' = \log(z) \tag{3.4}$$

The division of the x and y coordinates by z reverses the perspective transformation, equalizing the point density in the x-y-plane. Transforming the z coordinate helps to deal with the effects of depth quantization by compressing points as depth increases. It is easy to show that the transformation has the following property:

$$\frac{\partial x'}{\partial x} = \frac{\partial y'}{\partial y} = \frac{\partial z'}{\partial z} = \frac{1}{z} \tag{3.5}$$

Because the derivatives are equal, the local coordinate frame is stretched equally along all axes by the transformations. The important thing about this property is, that small cubic voxels are still cubic after the transformation. This leaves the geometry of space basically untouched in the foreground (if the voxel size is chosen sufficiently small), while voxels in the background are skewed and grow, to compensate for reduced amount of detail available in the data.

Rather than transforming the clouds back and forth, we instead transform the bins of the octree itself, creating an octree where bin volume (and thus, voxel size) effectively increases with distance from the camera viewpoint. Doing this directly within the octree allows us to determine adjacency as before (neighboring bins), even though distance between neighboring voxels increases with distance from the camera. Fig. 3.5.1 illustrates this advantageous effect of this transformation on the segmentation.

3.6 Experimental Results

In order to evaluate the quality of supervoxels generated by VCCS, we performed a quantitative comparison with three state-of-the-art superpixel methods using publicly available source code. We selected the two 2D techniques with the highest published performance from a recent review [**?**]: a graph based method, GCb10 [**?**][1], and a gradient ascent local clustering method, SLIC [**?**][2]. Additionally, we selected another method which uses depth images, DASP[**?**][3]. Examples of over-segmentations pro-

[1] http://www.csd.uwo.ca/~olga/Projects/superpixels.html

[2] http://ivrg.epfl.ch/supplementary_material/RK_SLICSuperpixels/index.html

[3] https://github.com/Danvil/dasp

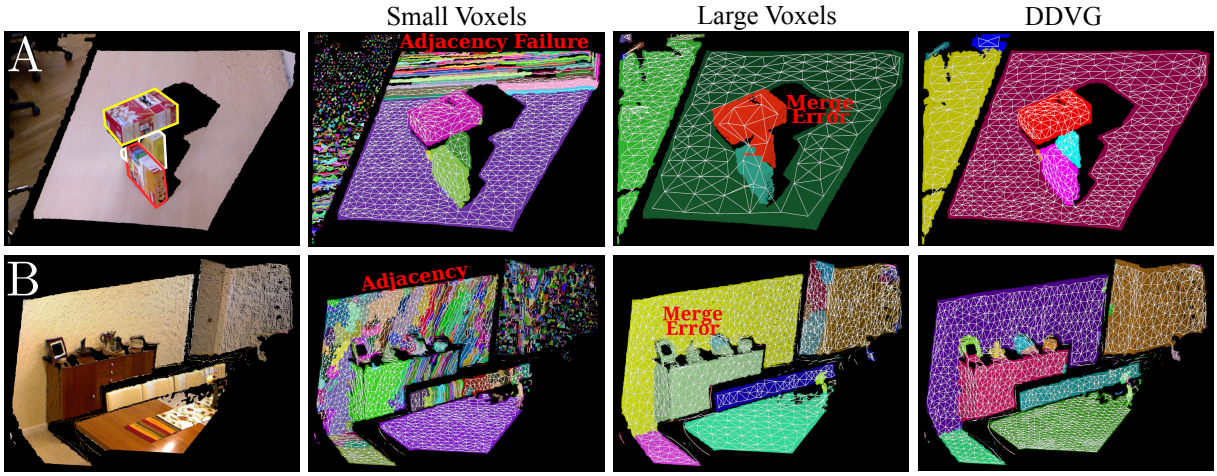

Figure 3.5.1: Two example point clouds **(A,B, left)** showing the need for the Depth Dependent Voxel Grid. For better visibility outlines have been drawn around the boxes in A. Using *Small Voxels* objects close to the camera can be segmented, but adjacency breaks down as the depth increases and the point density decreases. Using *Large Voxels* corrects the adjacency graph in the background, but leads to objects being merged in the foreground due to the coarse resolution. Using *DDVG*, the scale of the voxels gradually increases with distance from the camera – adapting to the increased noise level and lower point density – consequently adjacency is maintained and the segmentation of scenes with large depth variance is possible using fixed parameters. Note, the flat rug on the table in B does not differ enough from the table's surface and cannot be segmented by any purely depth dependent method.

duced by the methods are given in Figure 3.6.3.

3.6.1 Dataset

For testing, we used the recently created NYU Depth Dataset V2 semantic segmentation dataset of Silberman *et al [***?** *]*[4]*. This contains 1449 pairs of aligned RGB and depth images, with human annotated densely labeled ground truth. The images were captured in diverse cluttered indoor scenes, and present many difficulties for segmentation algorithms such as varied illumination and many small similarly colored objects. Examples of typical scenes are shown in Figure 3.6.3.*

3.6.2 Returning to the Projected Plane

RGB+D sensors produce what is known as an organized point cloud- a cloud where every point corresponds to a pixel in the original RGB and depth images. When such a cloud is voxelized, it necessarily loses this correspondence, and becomes an unstructured cloud which no longer has any direct relationship back to the 2D projected plane. As such, in order to compare results with existing 2D methods we were forced to devise a scheme to apply supervoxel labels to the original image.

To do this, we take every point in the original organized cloud and search for the nearest voxel in the voxelized

[4] http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html

Figure 3.6.1: Example of hole-filling for images after returning from voxel-cloud to the projected image plane. Depth data, shown in the top left, has holes in it, shown as dark blue areas (here, due to the lamp interfering with the Kinect). The resulting supervoxels do not cover these holes as shown in the bottom left, since the cloud has no points in them. To generate a complete 2D segmentation, we fill these holes in using the SLIC algorithm, resulting in a complete segmentation, seen in the top right. The bottom right shows human annotated ground truth for the scene.

representation. Unfortunately, since there are blank areas in the original depth image due to such factors as reflective surfaces, noise, and limited sensor range, this leaves us with some blank areas in the output labeled images. To overcome this, we fill in any large unlabeled areas using the SLIC algorithm. This is not a significant drawback, as the purpose of the algorithm is to form supervoxels in 3D space, not superpixels in the projected plane, and this hole-filling is only needed for comparison purposes. Additionally, the hole filling actually makes our results worse, since it does not consider depth, and therefore tends to bleed over some object boundaries that were correctly maintained in the supervoxel representation. An example of what the resulting segments look like before and after this procedure are shown in Figure 3.6.1.

3.6.3 Evaluation Metrics

*The most important property for superpixels is the ability to adhere to, and not cross, object boundaries. To measure this quantitatively, we have used two standard metrics for boundary adherence- boundary recall and under-segmentation error[**? ?**]. Boundary recall measures what fraction of the ground truth edges fall within at least two pixels of a superpixel boundary. High boundary recall indicates that the superpixels properly follow*

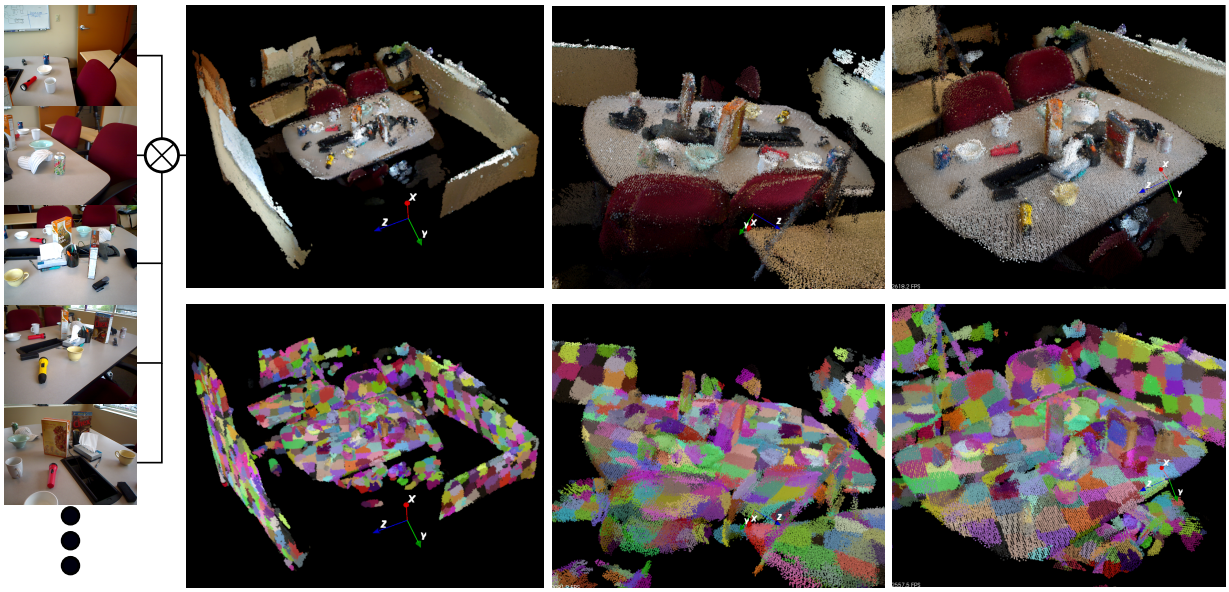

Figure 3.6.2: Over-segmentation of a cloud from the RGB-D scenes dataset[**?**]. The cloud is created by aligning 180 kinect frames, examples of which are seen on the left side. The resulting cloud has over 3 million points, which reduces to 450k points at $R_{voxel} = 0.01m$ and 100k points with $R_{voxel} = 0.02m$. Over-segmentation of these take 6 and 1.5 seconds, respectively (including voxelization).

Figure 3.6.3: Examples of under-segmentation output. From left to right- ground truth annotation, SLIC, GCb10, DASP, and VCCS. Each is shown with two different superpixel densities.

the edges of objects in the ground truth labeling. The results for boundary recall are given in Figure 3.6.4. As can be seen, VCCS and SLIC have the best boundary recall performance, giving similar results as the number of superpixels in the segmentation varies.

Under-segmentation error measures the amount of leakage across object boundaries. For a ground truth segmentation with regions $g_1, \ldots, g_M$, and the set of superpixels from an over-segmentation, $s_1, \ldots s_K$, under-segmentation error is defined as

$$E_{useg} = \frac{1}{N}\left[\sum_{i=1}^{M}\left(\sum_{s_j | s_j \cap g_i} |s_j|\right) - N\right], \tag{3.6}$$

where $s_j \mid s_j \cap g_i$ is the set of superpixels required to cover a ground truth label g_i, and N is the number of labeled ground truth pixels. A lower value means that less superpixels violated ground truth borders by crossing over them. Figure 3.6.4 compares the four algorithms, giving under-segmentation error for increasing superpixel counts. VCCS outperforms existing methods for all superpixel densities.

3.6.4 Time Performance

As superpixels are used as a preprocessing step to reduce the complexity of segmentation, they should be computationally efficient so that they do not negatively impact overall performance. To quantify segmentation speed, we measured the time required for the methods on images of increasing size (for the 2D methods) and increasing number of voxels (for VCCS). All measurements were recorded on an Intel Core i7 3.2Ghz processor, and are shown in Figure 3.6.5. VCCS shows performance competitive with SLIC and DASP (the two fastest superpixel methods in the literature) for voxel clouds of sizes which are typical for Kinect data at $R_{voxel} = 0.008m$ (20-40k voxels). It should be noted that only VCCS takes advantage of multi-threading (for octree, kd-tree, and FPFH computation), as there are no publicly available multi-threaded implementations of the other algorithms.

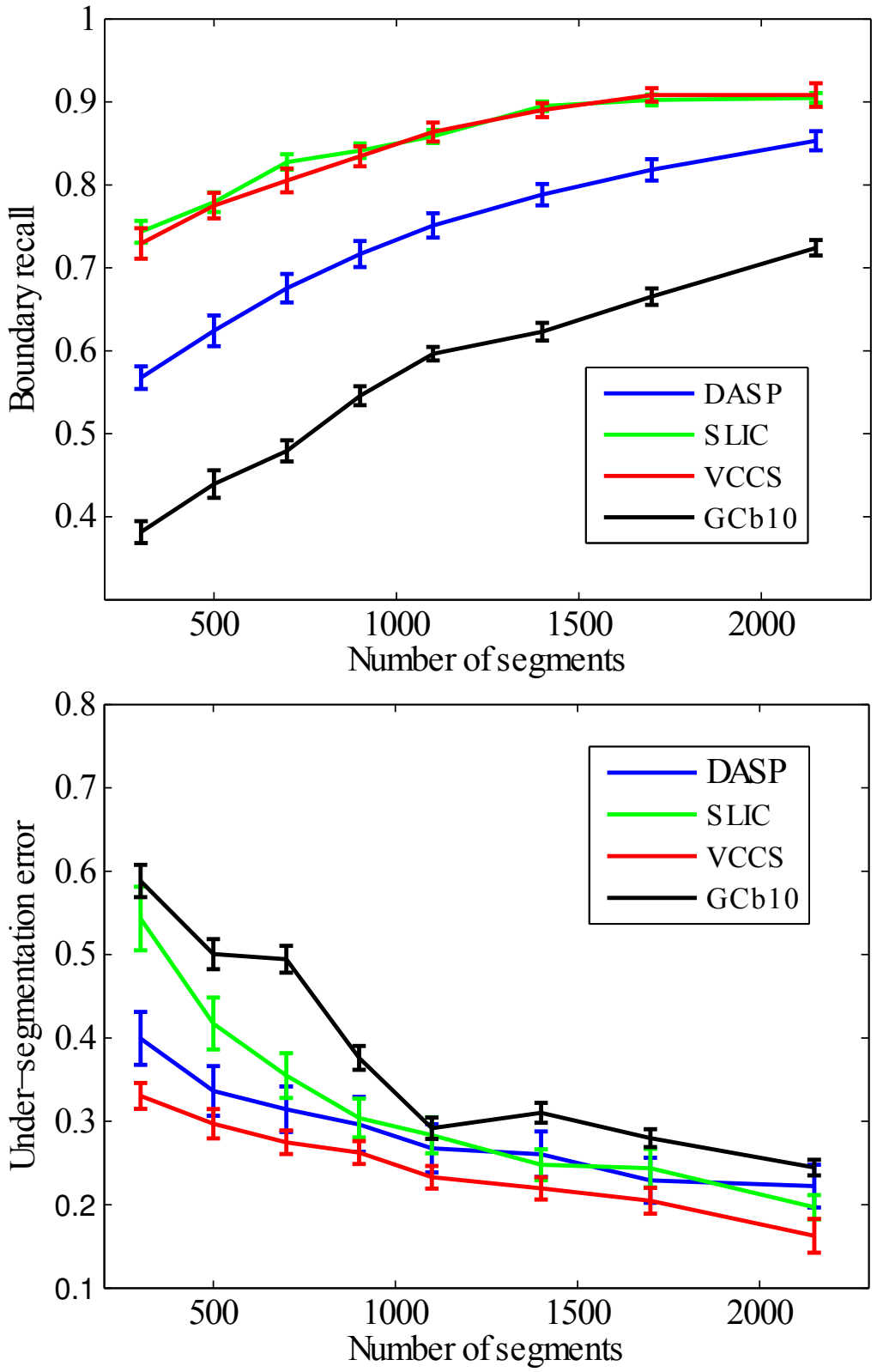

Figure 3.6.4: Boundary recall and under-segmentation error for SLIC, GCb10, DASP, and VCCS.

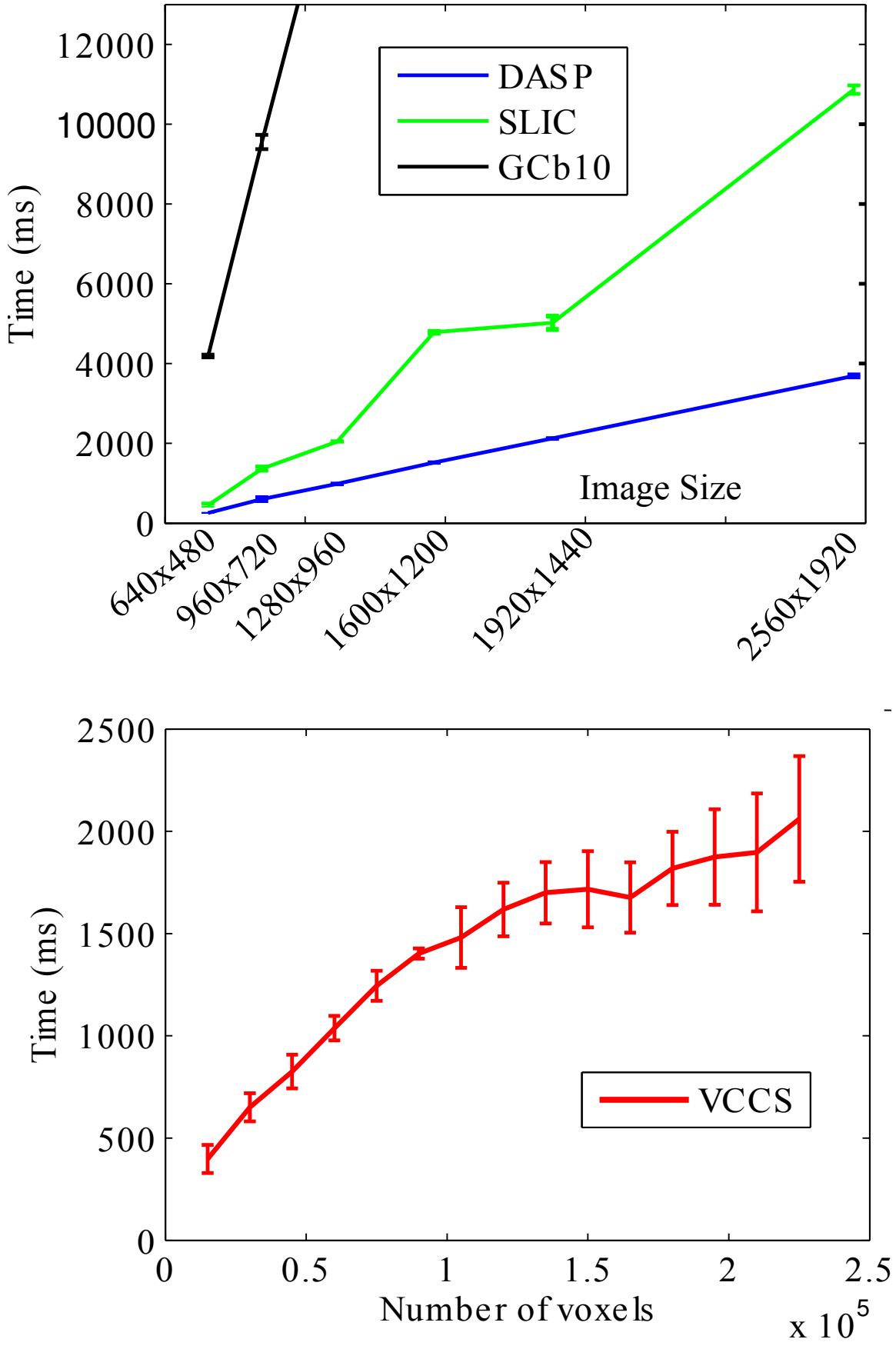

Figure 3.6.5: Speed of segmentation for increasing image size and number of voxels. Use of GCb10 rapidly becomes unfeasible for larger image sizes, and so we do not adjust the axes to show its run-time. The variation seen in VCCS run-time is due to dependence on other factors, such as R_{seed} and overall amount of connectivity in the adjacency graphs.

Some Quote.

Quoteauthor Lastname

4

Model-Based Point Cloud Video Segmentation

There's something to be said for having a good opening line.

4.1 Sequential Update of Perceptual Model

Adding newly observed points to an existing supervoxel octree is accomplished through a three stage process. First, we must insert the points into the octree, and initialize new leaves for them if they did not exist previously. This results in an octree where leaves fall into three possible categories (illustrated in Fig. 4.1.1; they are either new, observed, or unobserved in the most recent observation. Handling of new leaves is straightforward; we simply calculate adjacency relations to existing leaves and flag them as unlabeled.

To determine whether a leaf which existed previously has changed, we test the distance between the centroid of the points falling within its voxel (from the new frame) and its previous centroid. This is done in the same feature space used for growing the supervoxels, that is, we test whether the normal, color, and spatial location have varied more than a threshold value. This threshold is set to a relatively low constant value so that it favors false-positives (finding change when there was none), as they do not impact the tracking performance of the algorithm, but only have a slight effect on its run-time. If a leaf is found to have changed, we remove its previous labeling. We also perform a global check to see if more than half of a supervoxels support has changed; if so, we completely remove the supervoxels label from all of its constituent voxels.

Finally, we must consider how to handle leaves which were not observed in the inserted point cloud. Rather than simply prune them, we first check if it was possible to observe them from the viewpoint of the sensor which generated the input cloud. This occlusion check can be accomplished efficiently using the octree by determining

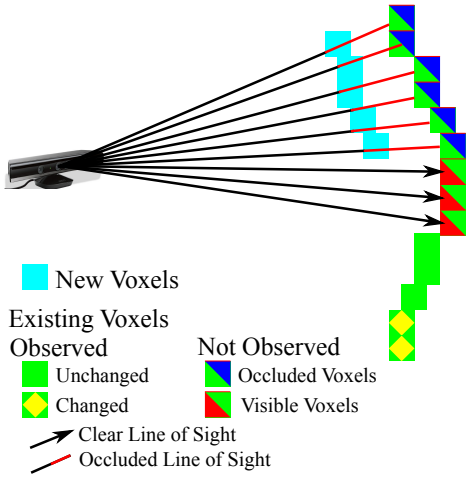

Figure 4.1.1: Categorization of voxels based on new frame of data. Voxels fall into three categories, they are either new, observed or not observed in the frame. Furthermore, observed voxels can either have changed or remained the same, while voxels not observed in the frame are either occluded or no longer exist (in which case they should be deleted).

if any voxels exist between unobserved leaves and the sensor viewpoint. If a clear line of sight exists from the leaf to the camera, it can safely be deleted. Conversely, if the path is obstructed, we "freeze" the leaf, meaning that it will remain constant until it is either observed or passes the line of sight test in a future frame (in which case, it can be safely deleted). This occlusion testing means that tracking of occluded objects is trivial, as occluded voxels remain in the observations which are used for tracking.

Once the octree voxels have been updated, we then proceed to update the supervoxels as before. That is, first we generate new seeds in regions of large unlabeled voxels, and then conduct the iterative region growing. This results in new supervoxels in regions which are new or changing, while leaving supervoxels in static and occluded regions unchanged. This reduces the tracking and segmentation problem to finding the best joint association of these new supervoxels with those from the prior time-step.

4.2 Tracked Model Representation

*While ideally one could directly track supervoxels themselves, this is generally not reliable due to the aperture problem seen in neural visual fields [**?**]; local motion can only be estimated perpendicular to a contour that extends beyond its field of view [**?**]. This means that in order to properly estimate motion of supervoxels, we must extend our considered field of view significantly beyond the size of the supervoxel itself; in fact, our aperture must contain the borders of the object, otherwise pairwise association of supervoxels is indeterminate.*

*As such, we first merge supervoxels into contiguous higher level object groupings. For this work, we use a plane fitting and removal algorithm to remove supporting surfaces, followed by a euclidean clustering of the remaining supervoxels as in [**?**]. It should be stressed that the overall tracking itself is independent of the segmentation used to initialize objects; one could easily use a model-based segmentation, or even a 2D classifier scheme on the original RGB image. Regardless of the segmentation used, the supervoxel clusters found are used to initialize the models which will be tracked.*

4.3 Supervoxel-Based Particle Filters

Tracking of the segmented models is accomplished using a bank of independent parallel particle filters. The models consist of clouds of supervoxels, and observations are the supervoxels produced using the persistent scheme discussed in Section **??**. *The observation model measures distance in a feature space of spatial distance, normals, color (in HSV space), and labels. Weights of predicted states (x, y, z, roll, pitch, yaw) are measured by associating observed supervoxels with nearest supervoxels from the transformed models, and then measuring total distance in the feature space as (4.1). That is, the weight w_i^k of particle i belonging to object k (of size N_k) with state x_i is the sum of the products of the coherences W,*

$$w_i^k \sum_{p,q \in x_i} W_d W_{HSV} W_n W_l \ . \tag{4.1}$$

Coherences are calculated for each correspondence pair between model supervoxel p and observed supervoxel q,

$$\begin{aligned} W_d &= \frac{1}{1+\frac{\|p-q\|^2}{3R_{seed}^2}} \\ W_{HSV} &= \frac{1}{1+\|p_{HS}-q_{HS}\|^2} \\ W_n &= \frac{1}{1+\left|1-n_p \cdot n_q\right|} \\ W_l &= \begin{cases} 1, & L_p = L_q \\ \frac{N_k-1}{N_k}, & L_p \neq L_q \end{cases} \end{aligned} \quad , \tag{4.2}$$

where supervoxel q has label L_q, normal n_q, and hue & saturation p_{HS}.

*KLD sampling [***?** *] is used to dynamically adapt the number of particles to the certainty of predictions. As matching supervoxel labels gives a high certainty of a correct prediction, objects which are not moving, and therefore have static supervoxel labels, need very few particles for accurate tracking. Details of the particle filters themselves are beyond the scope of this work, but we refer the reader to [***?** *] for an in-depth description of their operation. For this work, it is sufficient to understand that the particle filters yield independent predictions of 6DoF object state, allowing a transformation of the model to the current time-step - roughly aligning it with the currently observed supervoxels.*

4.4 Association by Joint Energy Minimization

The final step in the tracking process is to associate the observed supervoxels to the predictions coming from the particle filters, that is, we need to solve the multiple target data association problem. This is accomplished using an energy minimization which seeks to find an optimal global association of supervoxels to predictions. To do this, we first create a list of all observed supervoxels which lie within a radius R_{seed} of each predicted supervoxel coming from the particle filters (see Fig. 4.4.1). Then we determine all supervoxels which could only be associated with one possible object, associate them, and remove them from further consideration.

To associate the remaining observed supervoxels, we determine which objects are competing for them, and

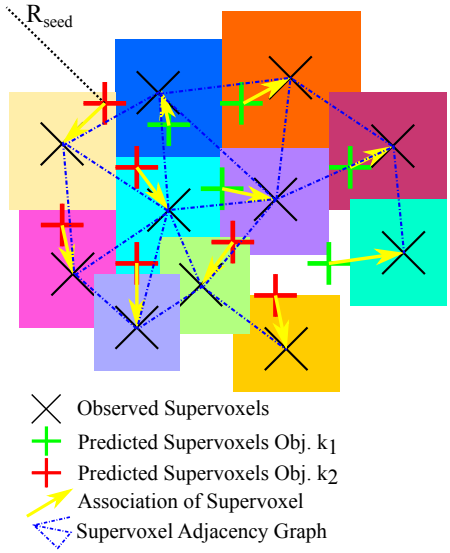

Figure 4.4.1: Association of observed supervoxels with predicted model supervoxels using global energy.

then find the predicted supervoxel from each object which lies closest to them in the feature space (using spatial location, normals, and color as in (3.3)). We adopt a RANSAC-like approach, similar to [?], to sample from the set of possible associations and determine a global association which best aligns the predictions to the observed supervoxels. Additionally, we use a weighted sampling strategy where the likelihood of assigning object k as the label L of supervoxel q falls off with increasing distance from the object centroid C_k

$$\mathcal{L}(L_q = k|C_k) = \frac{1}{C_k}. \tag{4.3}$$

To score a set of assignments, we compute a global energy, given in (4.4). Each global label association $\mathcal{A}$ *consists of local associations a which assign an object label k to each observed supervoxel q. The first summation term,* $\sum_p \|p_k - q\|$*, measures error in feature space between the observed supervoxel and the closest supervoxel in its associated predicted object* p_k*.*

$$E_{\mathcal{A}} = \prod_{a \in \mathcal{A}} \Delta_k \left(\sum_p \|p_k - q\| + \lambda \sum_{(q,q') \in \mathcal{N}} \delta(L_q \neq L_{q'}) \right) \tag{4.4}$$

The second summation is a smoothing prior which considers the adjacency graph of observed supervoxels. For every observed supervoxel, we compare its assigned label L_q *to the label of all supervoxels* q' *which lie within its adjacency neighborhood* $\mathcal{N}$*. We adopt the Potts model as in [?], where* $\delta(\dot{})$ *is 1 if the specified condition holds, and 0 otherwise, and* λ *is a weighting coefficient which controls the importance given to spatial continuity of labels.*

Finally, the multiplicative term $\prod_{a \in \mathcal{A}} \Delta_k$ *controls for the expansion or contraction of object volumes through the number of observed supervoxels associated with them.* Δ_k *penalizes for changes in volume by increasing the energy for deviations from unity in the ratio of observed supervoxels assigned to an object* $\hat{N}_k$ *with the number*

in the object model itself $\hat{N}_k$, that is

$$\Delta_k = \begin{cases} \hat{N}_k / N_k & \text{if } \hat{N}_k \geq N_k \\ 2 - \hat{N}_k / N_k & \text{if } \hat{N}_k < N_k \,. \end{cases} \tag{4.5}$$

Once the energy arrives at a stable minimum, we extract the resulting association of observed supervoxels to predicted results, and use them to update the tracked models.

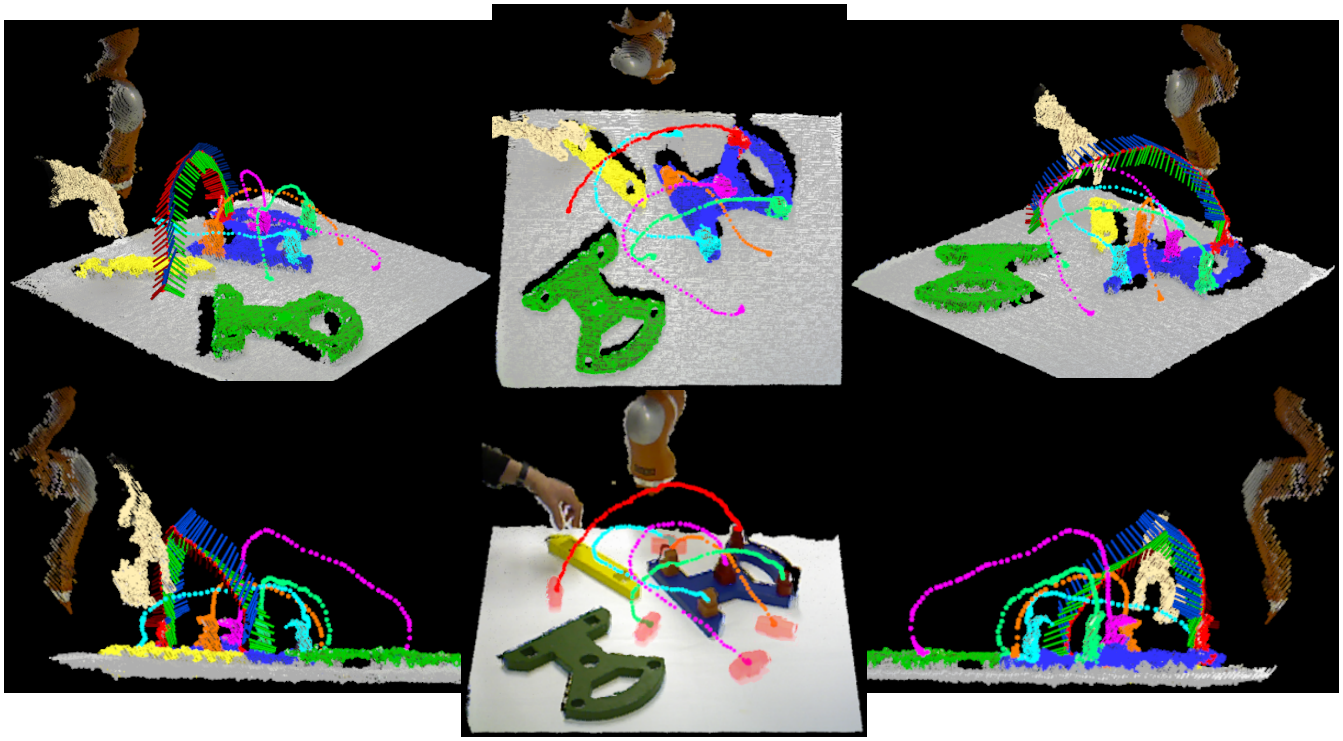

Figure 4.4.2: Result of tracking and segmentation on Cranfield scenario from different views. Here the tracks are shown as dots of the color of the tracked label for each timestep. Initial locations of the pegs are shown in the middle bottom frame as semi-transparent masks. Calculated orientation is shown for the red peg with a set of axes every second time-step; these axes show pose in a frame relative to the start.

4.5 Alignment and Update of Models

*The joint energy minimization results in a global association $\mathcal{A}$ which assigns observed supervoxels to tracked objects. In order to use this to update the object models, we determine a transform which aligns it to the internal representation stored by the particle filter. As an initial guess, we use the inverse of the predicted state, and then use an iterative closest point [**?**] procedure to refine the transform such that the set of observed supervoxels best aligns with the model prior. We then replace the model prior with the new observed supervoxels.*

As a final step, we use the refined transform to update the states of the particles. To do this, we shift each particle x_i towards the refined state $\hat{x}$, weighting the importance given to the refined state by a constant factor ε

$$x'_{i \in L} = (1 - \varepsilon) x_i + \varepsilon \hat{x} \,. \tag{4.6}$$

For this work, we found that an ε of 0.5 effectively removes noise (jitter) introduced by the replacement of the tracked model. Additionally, we correct the internal motion model of the particle filters to correspond to the new updated state.

4.6 Occlusion Handling

4.7 Experimental Results

In order to demonstrate the usefulness of the proposed method, in this Section we provide results from two successful applications. Both applications use the Cranfield scenario [?], a benchmark developed for assessing performance of assembly robot systems. Fig. 4.4.2 and the supplementary material[1] show the results of tracking and segmentation (only the pegs are shown in Fig. 4.4.2 to avoid clutter) using our Cranfield pieces. It can be seen that the algorithm is able to successfully track 6DoF states through the whole assembly task, even maintaining proper tracks for the pieces when they are fully occluded.

4.7.1 Imitation of Trajectories for Robot Manipulation

The standard way of teaching robots to perform human-like actions is imitation learning, also called programming by demonstration [? ?]. There are several ways to demonstrate movements: 1) recording movements in joint-space (joint angles) or target-space (Cartesian space) by ways of a motion capture device (requires putting markers on human body), 2) using kinaesthetic guidance (guiding a robot's movements by a human hand), or 3) via teleoperation (controlling a robot via joystick). The only way to obtain motion trajectories from human observation in a "non-invasive" procedure is by using stereo vision [?], however, usually it is model based. The tracking algorithm we have presented here can be used as an alternative method to obtain motion trajectories (in Cartesian space) in a model-free way.

To demonstrate this, we applied our tracking algorithm to obtain human motion trajectories in Cartesian space including orientation of manipulated object (in total six DoFs). We tested it using a recording of the Cranfield scenario where, first, we let a human demonstrate the action and then reproduced it using a KUKA Light Weight Robot (LWR) arm [?]. Specifically, here we imitate a human putting the separator block on the pegs. To generate trajectories for the robot from human demonstrations, we used a modified version of Dynamic Movement Primitives [? ?] (DMP) and learning method as described in [?]. We used Cartesian impedance control and, thus, generated six DMPs (three for motion of the end-effector in Cartesian space and three for orientation of the hand) based on trajectories obtained from the tracking algorithm. Here we used 100 equally spaced kernels with width $\sigma = 0.05$ for each dimension (for more details please refer to [?]). As demonstrated in Fig. 4.7.1 and the supplementary video, trajectories obtained by the proposed tracking algorithm are sufficiently accurate to allow reproduction of the human motion.

[1] See also http://www.youtube.com/watch?v=odVzWgW6Bs8 and https://www.youtube.com/watch?v=GjmUhm2JitU for longer versions.

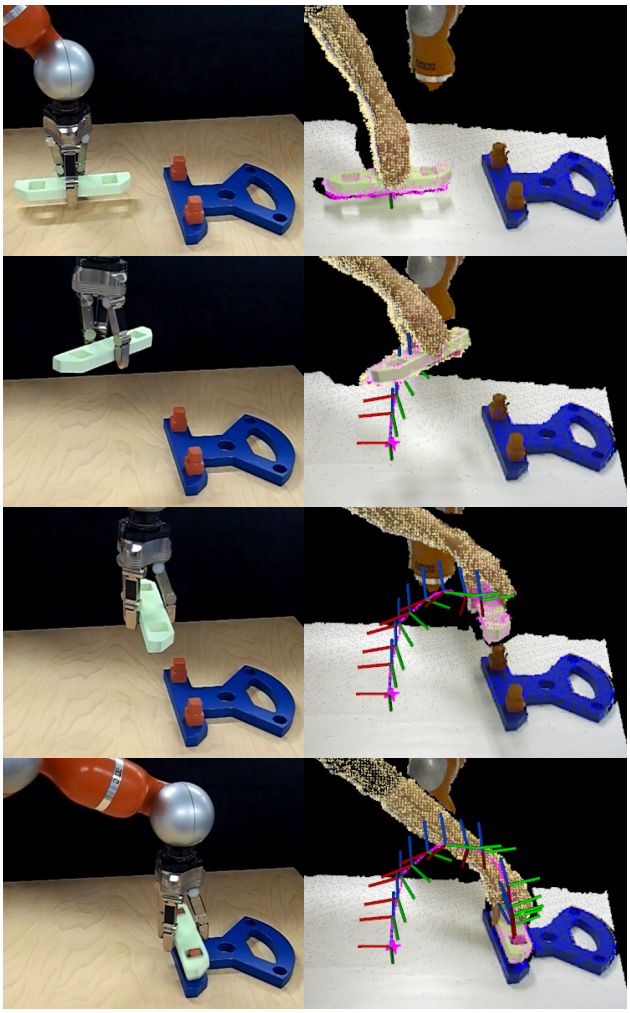

Figure 4.7.1: Kuka LWR arm imitating trajectory and pose learned from tracked human demonstration.

4.7.2 Semantic Summaries of Actions

*A fundamental task for intelligent autonomous robots is the problem of encoding long chain manipulations in a generic way, for use in tasks such as learning and recognition. As a demonstration of the usefulness of the proposed tracking framework, we use a recently introduced novel Semantic Event Chain (SEC) approach [**?**] which converts each segmented scene to a graph: nodes represent segment (i.e. object) centers and edges indicate whether two objects touch each other or not. By using an exact graph matching technique the SEC framework discretizes the entire graph sequence into decisive main graphs. A new main graph is identified whenever a new node or edge is formed or an existing edge or node is deleted. Thus, each main graph represents a "key frame" in the manipulation sequence. Figure 4.7.2 shows a few detected sample key frames from the long Cranfield action. While the complete action has in total 1453 frames, the SEC representation reduces it to just 35 key frames, each of which represents a topological change in the scene.*

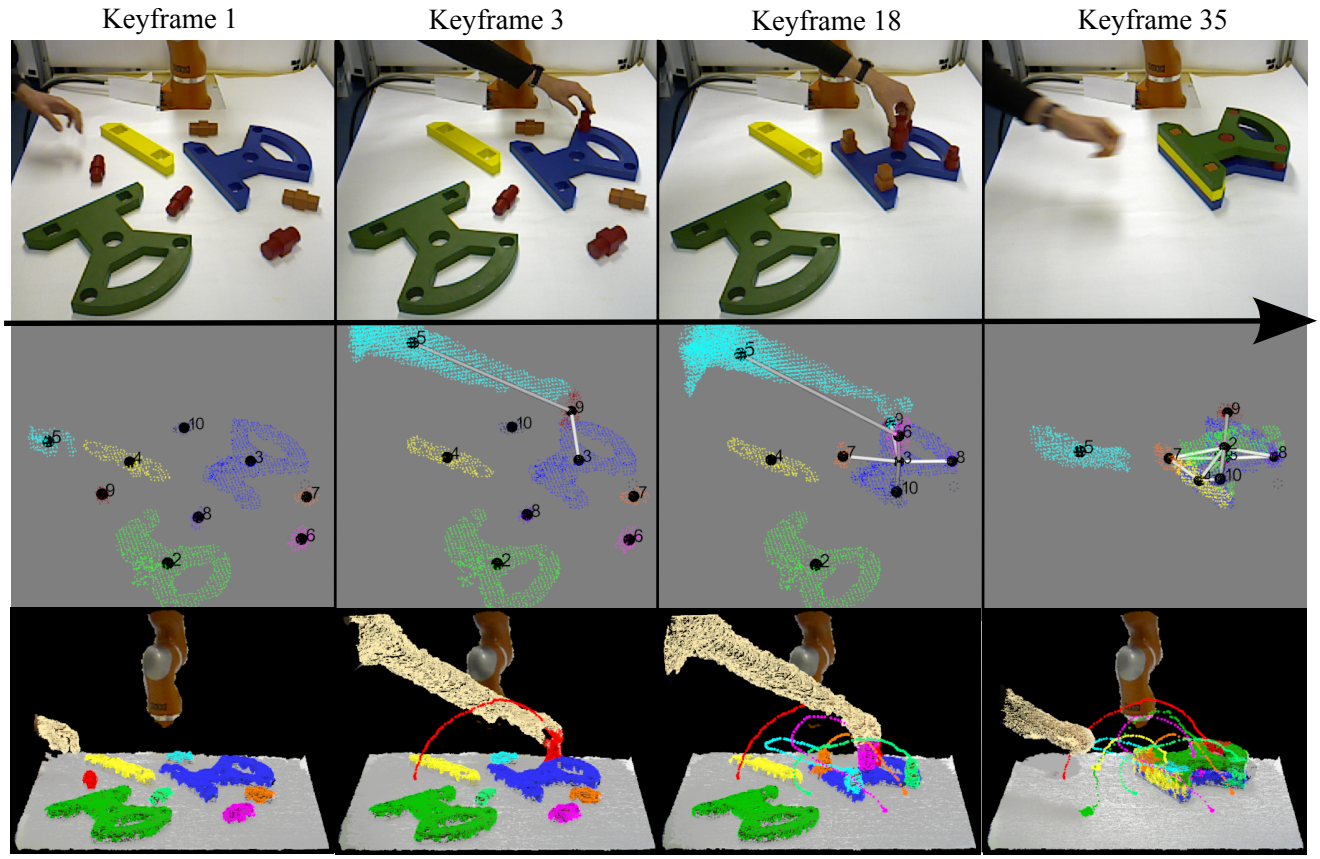

Figure 4.7.2: A few example key frames extracted from the long Cranfield action. Numbered nodes represent interacting objects, while edges show touching relations between objects. Each keyframe represents a topological change in the scene - here we show 4 of the 35 keyframes.

Some Quote.

Quoteauthor Lastname

5

Hierarchical Point Cloud Video Segmentation

There's something to be said for having a good opening line.

5.1 Local Convex Patches

5.2 Hierarchy of Temporal Hypotheses

5.3 Hypothesis Pruning

5.4 Extracting a Realization of Perceptual Model

5.5 Experimental Results

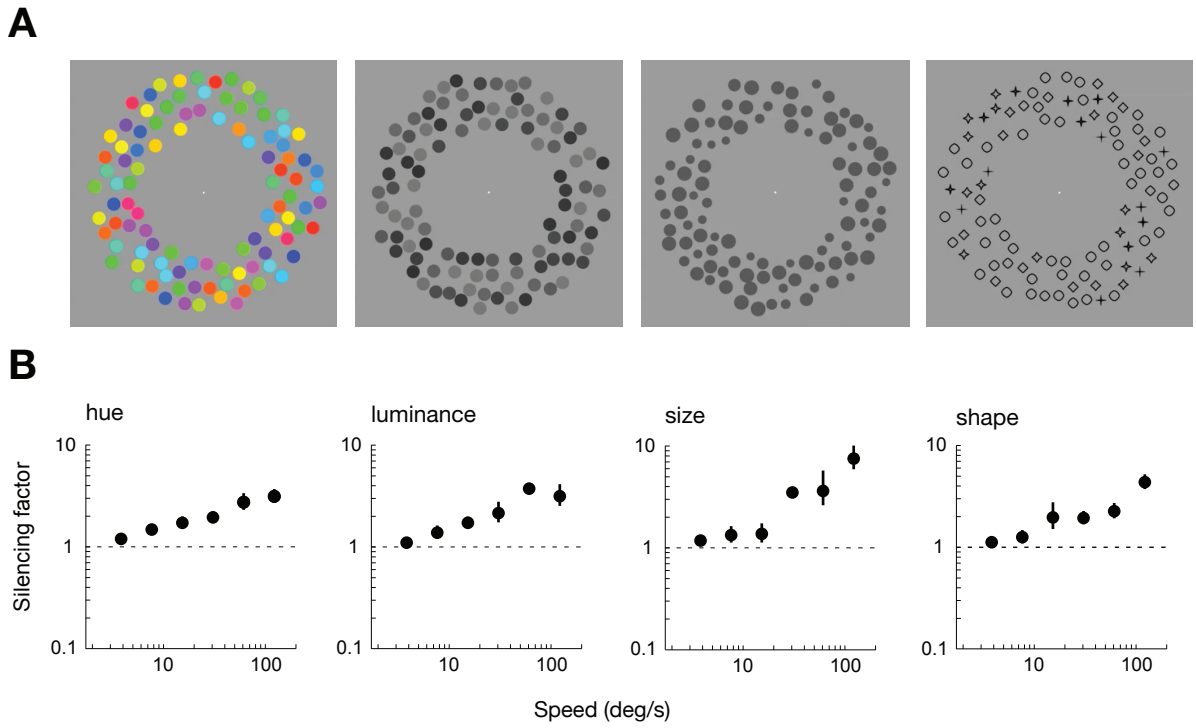

Figure 5.5.1: This is a figure that floats inline and here is its caption.

Some Quote.

Quoteauthor Lastname

6 Conclusions

There's something to be said for having a good opening line.

6.1 Summary and Contributions

6.2 Directions for Future Work

6.3 Discussion of Limitations of Benchmarks

Bibliography

Appendices

A

The Oculus Vision System

B

Particle Filters