IJMTST is UGC Approved Journal with Journal Id: 43137

International Journal for Modern Trends in Science and Technology, 8(S08): 01-09, 2022
Copyright © 2022 International Journal for Modern Trends in Science and Technology
ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST08S0801
Available online at: https://www.ijmtst.com/vol8si08.html

Eucledean Distance of Image Restoration using Mathematical Modeling and Simulation

K.Tejaswi | SK.RaziyaSulthana | B.Vishnupriya

Computer Science and Engineering , Chalapathi Institute of Engineering and Technology, Lam, Guntur, A.P, India

To Cite this Article
K.Tejaswi, SK.RaziyaSulthana and B.Vishnupriya. Eucledean Distance of Image Restoration using Mathematical Modeling and Simulation. International Journal for Modern Trends in Science and Technology 2022, 8(S08), pp. 01-09. https://doi.org/10.46501/IJMTST08S0801

Article Info
Received: 26 May 2022; Accepted: 24 June 2022; Published: 28 June 2022.

ABSTRACT

In this paper, we propose a novel algorithm that con-currently performs feature engineering and non-linear supervised hashing function learning. Our technical contributions in this paper are two-folds: 1) deep network optimization is often achieved by gradient propagation, which critically requires a smooth objective function. The discrete nature of hash codes makes them not amenable for gradient-based optimization. To address this issue, we pro-pose an exponentiated hashing loss function and its bilinear smooth approximation. Effective gradient calculation and propagation are thereby enabled; 2) pre-training is an important trick in supervised deep learning. The impact of pre-training on the hash code quality has never been discussed in current deep hashing literature. We propose a pre-training scheme inspired by recent advance in deep network based image classification, and experimentally demonstrate its effectiveness.The approach extracts temporally consistent object tubes based on an o -the-shelf detector. Besides the class label for each tube, this provides a location prior that is independent of motion. For the final video segmentation, we combine this information with motion cues. The method overcomes the typical problems of weakly supervised/unsupervised video segmentation, such as scenes with no motion, dominant camera motion, and objects that move as a unit. In contrast to most tracking methods, it provides an accurate, temporally consistent segmentation of each object. We report results on four video segmentation datasets: YouTube Objects, SegTrackv2, Ego Motion, and FBMS.

KEYWORDS: Video Segmentation, Motion Segmentation, Object Tracking.

1.INTRODUCTION

Various similarity metrics such as Hamming distance cosine similarity, l_p distance with $p \in (0,2]$, Jaccard index and Euclidean distance .

Video object segmentation plays a role in many high level computer vision tasks, such as action and event recognition. In contrast to single images, videos provide motion as a very strong bottom-up cue that can be exploited to support the high level tasks.

For this reason, video segmentation is often approached with unsupervised, purely bottom-upmethods. Especially motion segmentation can work quite well in a bottom-up fashion, if the objects of interest show some independent motion in the video. However, this is not the case in all videos. Very often, objects of interest are mostly static and almost all motion is due to camera motion. In such cases, motion segmentation fails. Also in cases where objects are moving jointly, such as a horse and its rider, a separation of the objects is often not

possible with just bottom-up cues. These limitations are avoided by adding user input that decides in these cases .However, this is not an option for a system that is supposed to automatically interpret video material. exponentiated objective function $\boldsymbol{Q}$ defined as the accumulation over all data pairs:

$(\theta^*, w_k^*) = argmin_\theta, w_k \boldsymbol{Q} \triangleq \sum_{i,j} \boldsymbol{l}(\boldsymbol{x_i}, \boldsymbol{x_j}),$ **(5)**

where represents the collection of parameters in the deep networks excluding the hashing loss layer. The atomic loss term is

$l(x_i, x_j) = e^{-Y_{i,j}(b_i \circ b_j)}$ (6)

This novel loss function enjoys some elegant traits de-sired by deep hashing compared with those in BRE [14], MLH [22] and KSH [20]. It establishes more direct connection to the hashing function parameters by maximizing the correlation of code product and pair wise labeling. In comparison, BRE and MLH optimize the parameters by aligning Hamming distance with original metric distances or enforcing the Hamming distance larger/smaller than pre-specified thresholds.

Video Object Segmentation

The key input to our video object segmentation are so-called tubes. Individual tubes are generated by tracking the detections of an o -the-shelf R-CNN detector [6]. The subsequent spatiotemporal segmentation is guided by this initial localization and the corresponding appearance cues.

Benjamin Drayer and Thomos Brox:

Tube extraction on three shots from the Youtube dataset. Neither strong changes in viewpoint, as in the first example, nor multiple instances of the same object class, as in the second example, are a problem for our approach. The third example shows a wrong classification of the motorcycle as a bicycle in the second image (green box). The method recovers later from this failure case.

Tube Extraction

The initial set of detections is generated by classifying the fast edge boxes from Zitnick and Dollar [31] with the R-CNN from Girshick et al. [6]. We denote the set of detected boxes with β and the i^{th} detection in frame t with B_t^i. Extracting a consistent tube over time translates to finding the longest path in a graph that connects all bounding boxes of a frame with all bounding boxes in successive frames:

$$p^* = \underset{p \le b}{argmax} \sum_{t_1 < t_2} S(B_{t1}^i, B_{t2}^j),$$

where $S(\cdot,\cdot)$ measures the similarity between two detection $B_{t_1}^i$ and $B_{t_2}^j$;

Pair wise potential

The pair wise term, enforcing spatial and temporal smoothness is weighted potts model:

$$E_p(u) = \sum_{(v_1,v_2)\in\varepsilon} \delta(u(v_1), u(v_2)).\lambda(v_1, v_2)$$

Where δ is the Kronecker delta and $\lambda(v_1, v_2)$ is the edge weight.

Gradient computation:

A prominent advantage of exponential loss is its easy conversion into multiplicative form, which elegantly simplifies the derivation of its gradient. For presentation clarity, we hereafter only focus on the calculation conducted over the topmost hashing loss layer. Namely, $h_k(X) = sign[w_k^\top z]$ for bit k, where $z = \phi(x)$ are the response values at the second top layer and w_k parameters to be learned for bit k(k=1……K).

Following the common practice in deep learning, two group of quantities $\partial Q/\partial w_k, k = 1$ and $\partial Q/\partial z_i$ (I ranges over the index set of current mini batch) need to be estimated on the hashing loss layer at each iteration.

The former group of quantities are used for updating w_k, k=1…..K. and the letter are propagated backwards to the bottom layers. The additive algebra of hash code product in eqn(3).Inspires us to estimate the gradients in a leave one out mode. For atomic loss in eqn,(6), it is easily verified

$$\ell(x_i, x_j) = \frac{e^{-Y_{i,j}(b_i \circ b_j)}}{e^{-Y_{i,j}(b_i(\backslash k) \circ b_{j(\backslash k)})}.e^{-\frac{1}{k}Y_{i,j}(b_i(k)b_{j(k)})}}$$

where only the latter factor is related to w_k. Since the product $b_i(k), b_j(k)$ can only be -1 or 1, we can linearize the latter factor through exhaustively enumerating all possible values, namely

$e^{-\frac{1}{k}Y_{i,j}(b_i(k)b_{j(k)})} = C_{i,j} + C'_{i,j}.(b_i(k)b_{j(k)}),$ (7)

Where $C_{i,j}, C'_{i,j}$ are two sample specific constants, calculated by

$C_{i,j} = \frac{1}{2}(e^{-\frac{1}{k}Y_{i,j}} + e^{\frac{1}{k}Y_{i,j}})$ and $C'_{i,j} = \frac{1}{2}(e^{-\frac{1}{k}Y_{i,j}} - e^{\frac{1}{k}Y_{i,j}})$. Since the hardness of calculating the gradient of Eqn. (7) lies in the bit product $b_i(k)\, b_j(k)$, we replace the signum function

using the sigmoid-shaped function $\sigma(x) = 1/(1 + \exp(-x))$, obtaining

$$\begin{aligned} \mathrm{b_i(k)b_{j(k)}} &= sign\left(w_k^\top z_i\right).sign\left(w_k^\top z_j\right) \\ &= sign\left(w_k^\top z_i z_j^\top w_k\right) \\ \approx 2.\sigma 9 w_k^\top z_i z_j^\top w_k) - 1 \qquad (8) \end{aligned}$$

Freezing the partial code product $\left(b_i(\backslash k) \circ b_j(\backslash k\right)$

We define an approximate atomic loss with only bits k active:

$\ell^{(k)}\left(x_i, x_j\right) \triangleq$

$e^{-Y_{i,j}\left(b_i(\backslash k) \circ b_j(\backslash k)\right)}.\left(c_{i,j} + c_{i,j}^{'}.\left(2.\sigma\left(w_k^\top z_i z_j^\top w_k\right) - 1\right)\right.$,

(9)Where the first factor $e^{-Y_{i,j}\left(b_i(\backslash k) \circ b_j(\backslash k)\right)}$ plays a role of reweighting specific data pair, conditioned on the rest K-1

Algorithm 1 Deep Hash Algorithm

1:Input: Training set $\mathcal{X}$, data labels, and step size $\eta > 0$;

2: Output: network parameters $\boldsymbol{w_k}$, **k=1…..K for the hashing –loss layer, and θ for the other layers;**

3: Concatenate all layers (excluding top hashing-loss layer) with a softmax layer that defines an image classification task;

4:Apply AlexNet [13] style supervised parameter learning algorithm, obtaining θ.

5: Calculate neuron response on second topmost layer through $\boldsymbol{z = \phi(x;\theta)}$**;**

Pre-training stage #2: initialize $\boldsymbol{w_k}$

6: Replicate all z's from previous stage;

7:While not converged do

9: for k=1 to K do

10: update $\boldsymbol{w_k}$ **by minimizing the image classification error;**

11: end for

12: end while

Simultaneous supervised fine-tuning

13: while not converged do

14: Forward computation starting from Z;

15: for k-1 to K do

16: Estimate $\frac{\partial Q}{\partial w_k} \alpha \sum_{i,j,k} \partial \ell^{(k)}(z_i, z_j)/\partial w_k$ **;**

17: Update $w_k \leftarrow w_k - \eta.\frac{\partial}{\partial w_k}$ **;**

18: end for

19: Estimate $\frac{\partial Q}{\partial z_i} \alpha \sum_{j,k} \partial \ell^{(k)}(z_i, z_j)/\partial z_i \ ,\forall_i$ **;**

20: propagate $\frac{\partial Q}{\partial z_i}$ **to bottom layers ,updating θ ;**

21: end while

Bits. Iterating over all k's the original lass function can now be approximated by

$\boldsymbol{\ell\left(x_i, x_j\right) \approx \frac{1}{K}\sum_{k=1}^{K} \ell^{(k)}(x_i, x_j)}$ **(10)**

compared with other sigmoid-based approximation in previous hashing algorithms (e.g.KSH[20]),ours only requires $\boldsymbol{|w_k^\top z_i z_j^\top w_k|}$ **is sufficiently large.**

Since the objective $\boldsymbol{Q}$ **in Eqn.(5) is composition of atomic losses on data pairs, we only need to instantiate the gradient computation on specific data pair** $\boldsymbol{\left(x_i, x_j\right)}$**. Applying basic calculus rules and discarding some scaling factors, we first obtain**

$\frac{\partial \ell^{(k)}(x_i, x_j)}{\partial w_k^\top z_i z_j^\top w_k} \propto$

$e^{-Y_{i,j}\left(b_i(\backslash k) \circ b_j(\backslash k)\right)}.c_{i,j}^{'}.\left(1 - \sigma\left(w_k^\top z_i z_j^\top w_k\right)\right).\sigma\left(w_k^\top z_i z_j^\top w_k\right)$**,**

And further using calculus chain rule brings

$$\frac{\partial \ell^{(k)}(x_i, x_j)}{\partial w_k} = \frac{\partial \ell^{(k)}(x_i, x_j)}{\partial w_k^\top z_i z_j^\top w_k}.\left(z_i z_j^\top + z_j z_i^\top\right) w_k \ ,$$

$$\frac{\partial \ell^{(k)}(x_i, x_j)}{\partial z_i} = \frac{\partial \ell^{(k)}(x_i, x_j)}{\partial w_k^\top z_i z_j^\top w_k}.\left(w_k w_k^\top z_j\right) .$$

Importantly , the formulas below obviously hold by the construction of $\boldsymbol{\ell^{(k)}\left(x_i, x_j\right)}$:

The gradient computations on other deep network layers simply follow the regular calculus rules. We thus omit the introduction.

Two-Stage Supervised Pre-Training

Deep hashing algorithms (including ours) mostly strive to optimize pair wise (or even triplet as in [16]) similarity in Hamming space. This raises an intrinsic distinction compared with conventional applications of deep networks (such as image classification via AlexNet [13]). The total count of data pairs quadratic ally increases with regard to the training sample number, and in conventional applications the number of atomic losses in the objective only linearly grows. This entails a much larger mini-batch size in order to combat numerical instability caused by under-sampling, which unfortunately often exceeds the maximal memory space on modern CPU/GPUs.

The network parameters are learned through optimizing the objective of a relevant semantics learning task (e.g., image classification). After stage one is complete, we extract the neuron outputs of all training samples from the second topmost layer (i.e., the variable z's), feed them into another two-layer shallow network as shown in Figure1and initialize the hashing parameters w_k, k = 1 K.

Finally, all layers are jointly optimized in a fine-tuning process, minimizing the hashing loss
objective Q. The entire procedure is illustrated in Figure2and detailed in Algorithm1.

Experiments

This section reports the quantitative evaluations between our proposed deep hashing algorithm and other competitors.

Description of Datasets: We conduct quantitative comparisons over four image benchmarks which represent different visual classification tasks. They include MNISTfor handwritten digits recognition, CIFAR10which is a subset of 80 million Tiny Images datasetand consists of images from ten animal or object categories, Kaggle-Face, which is a Kaggle-hosted facial expression classification dataset to stimulate the research on facial feature representation learning, and SUN397 [30] which is a large scale scene image dataset of 397 categories. Figure3 shows exemplar images. For all selected datasets, different classes are completely mutually exclusive such that the similarity/dissimilarity sets as in Eqn (1) can be calculated purely based on label consensus. Table1summarizes the critical information of these experimental data, wherein the column of feature dimension refers to the neuron numbers on the second topmost layers (i.e., dimensions of feature vector z).

Implementation and Model Specification: We have implemented a substantially-customized version of the open source Caffe. The proposed hashing loss layer is patched to the original package and we also largely enrich Caffe's model specification grammar. Moreover, To ensure that mini-batches more faithfully represent the real distribution of pair wise affinities, we re-shuffle the training set at each iteration.

Baselines and Evaluation Protocol: All the evaluations are conducted on a large-scale private cluster, equipped with 12 NVIDIA Tesla K20 GPUs and 8 K40 GPUs. We de-note the proposed algorithm as Deep Hash. On the chosen benchmarks, Deep Hash is compared against classic or state-of-the-art competing hashing schemes, including unsupervised methods such as random projection-based LSH [6], PCAH, SH [28], ITQ [10], and supervised methods like LDAH [5], MLH [22], BRE [14], and KSH [20]. LSH and PCAH are evaluated using our own implementations. For the rest aforementioned baselines, we thank the authors for publicly sharing their code and adopt the parameters as suggested in the original software packages. Moreover, to make the comparisons comprehensive, four previous deep hashing algorithms are also contrasted, denoted as DH-1 and DH -1 from [29], DH-2 [19], and DH-3 [16]. Since the authors do not share the source code or model specifications, we instead cite their reported accuracies under identical (or similar) experimental settings.

All methods share identical training and query sets. After the hashing functions are learned on the training set, all methods produce binary hash codes for the querying data respectively. There exist multiple search strategies using hash codes for image search, such as hash table lookup [1] and sparse coding style criterion [18]. Following recent hashing works, we only carry out Hamming ranking once the hashing functions are learned, which refers to the process of ranking the retrieved samples based on their Hamming distances to the query. Under Hamming ranking protocol, we measure each algorithm using both mean-average-precision (mAP) scores and precision-recall curves.

Investigation of Hamming Ranking Results: Table3and Figure3show the MAP scores for our proposed Deep Hash algorithms (with supervised pre-training and fine-tuning) and all baselines. To clearly depict the evolving accuracies with respect to the search radius, Figure 4displays the precision-recall curves for all algorithms with 32 hash bits. There are three key observations from these experimental results that we would highlight:

On all four datasets, our proposed Deep Hash algorithm significantly perform better than all baselines in terms of mAP. For all non-deep-network based algorithm, KSH achieves the best accuracies on MNIST, CIFAR10 and Kaggle-Face, and ITQ shows top performances on SUN397. Using 48 hash bits, the best mAP scores obtained by KSH or ITQ are 0.9817, 0.5482, 0.4132, and 0.0471 on MNIST / CIFAR10 / Kaggle-Face / SUN397 respectively. In comparison, our proposed Deep Hash performs nearly perfect on MNIST (0.9938), and defeat KSH and ITQ by very large margins, scoring 0.7410, 0.5615, and 0.1293 on other three datasets respectively.

We also include four deep hashing algorithms by referring to the accuracies reported in the original publications. Recall that the evaluations in [29,16] feed baseline algorithms with non-CNN features (e.g., GIST). Interestingly, our experiments reveal that, when conventional hashing algorithms take CNN features as the input, the relative performance gain of prior deep hashing algorithms becomes marginal. For example, under 48 hash bits, KSH's mAP score 0.5482 is comparable with regard to DH-3's 0.581. We attribute the striking superiority of our proposed deep hashing algorithm to the importance of jointly conducting feature engineering and hash function learning (i.e., the fine-tuning process in Algorithm1).

4 Elevating inter-bit mutual complementarity is overly crucial for the final performance. For those methods that generate hash bits independently (such as LSH) or by en-forcing performance-irrelevant inter-bit constraints (such as LDAH), the mAP scores only show slight gains or even drop when increasing hash code length. Among all algorithms, two code-product oriented algorithm, KSH and our pro-posed Deep Hash, show steady improvement by using more hash bits. Moreover, our results also validate some known insights exposed by previous works, such as the advantage of supervised hashing methods over the unsupervised alter-natives.

Effect of Supervised Pre-Training: We now further high light the effectiveness of the two-stage supervised pre-training process. To this end, in Table3we show the MAP scores achieved by three different strategies of learning the network parameters. The scheme "Deep Hash (random init.)" refers to initializing all parameters with random numbers without any pre-training. A typical supervised gradient back-propagation procedure as in AlexNet [13] is then used. The second scheme "Deep Hash (pre-training)" refers to initializing the network using two-stage pre-training in Algorithm1, without any subsequent fine-tuning. It serves as an appropriate baseline for assessing the benefit of the fine-tuning process as in the third scheme "Deep Hash (fine-tuning)". In all cases, the learning rate in gradient descent drops at a constant factor (0.1 in all of our experiments) until the training converges.

Experimental results in terms of mean-average-precision (MAP) under various hash bits. The MAP scores are calculated based on Hamming ranking. Best scores are highlighted in bold. Note that the MAP scores are in the numerical range of ***[0; 1]***. We directly cite the performance reported in [29, 19,16] since the source codes are not publicly shared. In the table, "–" indicates the corresponding scores are not available. Refer to text for more details.

Experimental results in terms of mean-average-precision (mAP) under varying hash code lengths for all algorithms. Best viewing in color mode.

There are two major observations from the results in Table3. First, simultaneous tuning all the layers (including the hashing loss layer) often significantly boosts the performance. As a key evidence, "Deep Hash (random init.)" demonstrates prominent superiority on MNIST and CIFAR10 compared with "Deep Hash (pre-training)". The joint parameter tuning of "Deep Hash (random init.)" is sup-posed to compensate the low-quality random parameter initialization. Secondly, positioning the initial solution near a "good" local optimum is crucial for learning on challenging data. For example, the dataset of SUN397 has as many as 397 unique scene categories. However, due to the limitation of GPU memory, even a K40 GPU with 12GB memory only support a mini-batch of 600 samples at maximum. State differently, each mini-batch only comprises 1.5 samples per category on average, which results in a heavily biased sampling towards the pair wise affinities. We attribute the relatively low accuracies of "Deep Hash (random init.)" to this issue. In contrast, training deep networks with both supervised pre-training and fine-tuning (i.e., the third scheme in Table3) exhibit robust performances over all datasets.

Comparisons of three strategies of parameter initialization and learning for the proposed DeepHash. See text for more details.

Comparison with Hand-Crafted Features: To complement a missing comparison in other deep hashing works [29, 19,16]), we also compare the hashing performance with conventional hand-crafted features and CNN features extracted from our second topmost layers. Following the choices in relevant literature, we extract 800-D GIST feature from CIFAR10 images, and 5000-D Dens eSIFT Bag-of-Words feature from SUN397 images. The comparisons under 16 and 32 hash bits are

found in Table 4, exhibiting huge performance gaps between these two kinds of features. It clearly reveals how the feature quality impacts the final performance of a hashing algorithm, and a fair setting shall be established when comparing conventional and deep hashing algorithms.

Precision-recall curves under 32 hash bits on all image benchmarks

	Hand-Crafted Feature		CNN Feature	
	16 bits	32 bits	16 bits	32 bits
LSH	0.1215	0.1385	0.1354	0.1752
ITQ	0.1528	0.1604	0.2757	0.2862
BRE	0.1308	0.1362	0.2634	0.2803
MLH	0.1373	0.1334	0.1810	0.1800
KSH	0.2191	0.2081	0.3958	0.5039
DeepHash	0.216	0.2304	0.5472	0.5674

	Hand-Crafted Feature		CNN Feature	
	16 bits	32 bits	16 bits	32 bits
LSH	0.0067	0.0072	0.0059	0.0063
ITQ	0.0159	0.0157	0.0309	0.0410
BRE	0.0070	0.0075	0.0252	0.0319
MLH	0.0148	0.0147	0.0083	0.0144
KSH	0.0105	0.0095	0.0216	0.0300
DeepHash	0.0166	0.0189	0.0387	0.0525

MAP scores using hand-crafted features and CNN features in hashing-based image search. The method "Deep Hash" refers to the variant without fine-tuning. The top and bottom tables correspond to the results on CIFAR10 and SUN397 respectively.

Implementation Details

Similarity measure:

With the category label, the appearance and center term, we favor a consistent appearance of the object. The side and volume constraints enforce the tube to change its shape smoothly. Temporal consistency is encoded in the matching term, and the score rewards con dent detections.

Illustration of the graph structure for a sample video. The left column shows all initial detections, the right column the two final high scoring tubes that have been extracted: the boat (magenta) and the person (cyan). Dashed boxes indicate parts of the tube that have been interpolated between frames. The graph structure is shown in the middle, where we show exampled all edges for B_{t1_1} . The corresponding longest path is shown in green.

The category label is a very powerful indicator and has to be consistent through time.

Therefore we set

$$S_{category} = \begin{cases} 1 \\ -\infty \\ else \end{cases} if category(B_{t1}) = category(B_{t2})$$

Due to movement of the camera and/or the object itself, the bounding box can change over time. This is supposed to be a rather slow process, therefore we favor small changes in both, the volume and the sides.

$$S_{vol} = \min(\frac{vol(B_{t1})}{vol(B_{t2})}, \frac{vol(B_{t2})}{vol(B_{t1})})$$

The cost for the side change is computed in the same way, where we take the minimum of the height and the width change. The matching term gives the ratio on how many points of B_{t2} are matched by the optical flow F originating from B_{t1}.

$$S_{matc\,h} = \frac{|matc\acute{h}es|}{vol(B_{t2})}$$

$$Matc\acute{h}es = \{P \in B_{t2} | \ni q \in B_{t1} : F(q) = P\}$$

Although the optical flow is an indicator on how similar the two boxes are, the volume of B_{t2} and the possible distinct motion of object and background weaken this term.

We compensate for that by additionally penalizing the deviation of the propagated center c_p of box B_{t1}

With the actual center C of B_{t2}.

$$\boldsymbol{S_{center} = \frac{1}{1 + 0.1.\|c_p - c\|}}$$

Correlating B_{t1} with frame t_2 gives us the propagated center c_p. On a finer level, this is less accurate than optical flow, but it is more robust.

The appearance term is the cosine-distance of color histograms H(.):

$$S_{app} = \frac{\langle H(B_{t1}), H(B_{t2}) \rangle}{\|H(B_{t1})\|.\|H(B_{t2})\|}$$

This cue is independent of the area and shape given by the corresponding bounding box. Boxes with $S_{app} \leq 0.8$ are considered as distinct and the term is set to $-\infty$. When objects of the same class interact (e.g. overtaking cars, Figure 5), the appearance is important to track them correctly.

Graph:

We build the graph by connecting the detections in a temporal order. Since we cannot assume that the detections are present in every frame we interconnect each detection with the detections of the subsequent 20 frames. Additionally, each node is connected to the source and sink, so that new objects can enter and leave the scene, while being correctly tracked without introducing additional knowledge about the object's presence in the video. See Figure 4 for a visualization of such a graph.

Post processing:

We interpolate the possible sparse tube into a dense one. The missing box in frame t is interpolated by correlating the box found in frame (t -1) with frame t. Gaps of more than one frame are interpolated from both sides. The set of tubes is cleaned by a volumetric non-maximum suppression, where overlapping tubes of the same class with an intersection over union > 0:5 are suppressed by the longer, respective higher scoring tube.

Tube parameter evolution:

We give a justification of the different terms in our proposed similarity measure in form of an ablation study in Table. For the evaluation, we selected a subset of the SegTrackv2 dataset (birds of paradise, bmx, drift, girl, monkey, penguin and soldier) that excludes the categories, for which we have either no detector or not sufficient detections. With no or little detections, tracking is almost independent of different similarity measures. Note that we favored the SegTrackv2 dataset as it provides annotation for every frame and allows for a more detailed analysis. Having only every 10th frame annotated (YouTube), flickering or swapping of labels between objects (as in the drift sequence, Figure 5) will be missed by the evaluation metric.

	score	Sside	Svol	Svol Sside	Smatch	Scenter	Smatch Scenter	Sapp	*all*
-	*58:75*	*59:7*	*59:5*	***59:0***	***57:8***	*59:13*	*57:85*	*55:73*	59:83
+	*53:47*	*55:07*	*54:33*	*53:99*	*54:75*	*54:81*	*54:21*	*57:39*	59:83

Table5: Ablation study of the tube extraction on a subset of SegTrackv2, reported as IoU. Removing components from the system makes results worse (-). Some components are more important than others. The performance of the individual components (+) confirm that the appearance term is the most significant.

Results:

With the evaluation on four video segmentation datasets (YouTube Objects [9], egoMotion [16], SegTrackv2 [14] and FBMS [17]), we prove the performance of the proposed method. The datasets impose different challenges and shortcomings.

Benjamin Drayer and Thomas Brox

Multiple tubes tracking the same object (a) are a result of quick changes in appearance, shape or position and lead to over segmentation (e). Using the location cues (b-d), we merge the tubes and consistently segment the bird (f). EgoMotion and FBMS are complementary. In egoMotion, there is always a single object that is largely static, whereas FBMS contains multiple moving objects. The downside of FBMS is that there is no ground-truth annotation for static objects, because it is a benchmark dataset designed for motion segmentation.

The evaluation metric is the average intersection over union

$$IOU = \frac{|S \cap GT|}{|S \cup GT|},$$

Where S is the segmentation and GT the ground truth.

The results in Table 2 show that the foreground features without further processing perform similar to pure motion based segmentation. The best result is achieved by the combination of motion and appearance cues, which beats current state of the art methods by at least 3%.

Results for the YouTube dataset, reported as IoU. The proposed method performs 3% better than the current state-of-the-art method [30]. The combination of motion and appearance features lead to the best performance.

The SegTrackv2 dataset [14] consists of 14 videos with frame-wise ground-truth annotation. Single and multiple objects, slow and fast motion, as well as occluding and interacting objects are present. Note that only in sequences, where known objects are present, our method can perform well. When processing sequences with unknown objects such as parachute or worm, we can only rely on motion features and we fall back to the

approach in [18]. Regarding cases, in which we can extract tubes, e.g. bmx or drift sequence, we clearly outperform the other methods. On average, we are 4:3% better than the other methods; see Table 3. Qualitative results and comparisons are shown in Figure

	[17]	[11]	[18]	[15]	OURS
bird of paradise	17:2	79:0	74:9	43:2	50:5
birdfall	0:5	0:5	4:5		4:5
bmx-person	4:8	70:4	47:8	0:9	90:7
bmx-bike	1:2	17:3	16:3	20:0	33:5
cheetah-deer	1:9	1:9	47:1		41:9
cheetah-cheetah	0:9	3:9	17:9		0
drift-car1	35:1	50:2	48:4	36:0	70:1
drift-car2	12:4	0:3	35:0	39:6	60:2
frog	41:5	43:1	57:3		68:4
girl	52:1	51:4	53:8	65:8	65:4
monkey	35:8	22:8	64:8		65:2
monkeydog-dog	1:4	6:8	0	0	0
monkeydog-monkey	54:9	54:9	77:7	0	77:7
hummingbird-bird1	3:9	11:0	10:1	39:8	10:4
hummingbird-bird2	55:4	32:4	51:6	30:2	9:4
parachute	90:3	89:8	68:7		68:7
penguin-penguin1	8:5	8:6	4:7	20:0	43:0
penguin-penguin2	3:8	4:1	1:9	0:7	0
penguin-penguin3	0	3:8	1:7	14:9	0
penguin-penguin4	0	0	2:2	0	0
penguin-penguin5	0	0	8:9	0	0
penguin-penguin6	0	5:3	18:3	0:61	73:6
soldier	63:0	50:1	36:9	0	64:0
worm	2:7	23:0	69:0		69:0
avg obj	27:9	34:5	43:7	24:8	48:0
avg vid	20:3	26:3	34:2	18:3	40:3

Table 7: Quantitative results for the SegTrackv2 dataset, reported as IoU. ( ) Results are averaged over the videos containing objects from the 20 Pascal classes.

	car	cat	Chair	dog	average
[17]	33:6	13:5	16:2	41:7	26:3
[11]	37:9	45:3	19:8	53:4	39:1
[18]	47:6	56:6	59:5	64:2	57:0
[15]	86:1	16:6	39:0	47:1	47:2
OURS	78:0	65:7	73:5	75:2	73:1

Results reported as IoU for the egoMotion dataset [16]. We have a 16% improvement compared to motion segmentation approaches.

Results for the SegTrackv2 dataset. From top to bottom: ground truth, Ochs et al. [17], Keuper et al.[11], Papazoglou and Ferrari [18] and ours. The example from the drift sequence shows that motion based methods fail when the objects are close to each other and move similarly. In the second frame the two pylons in the upper left corner are detected as traffic lights. For the penguin sequence, we detected only two tubes and thus did not segment the remaining penguins.

Concluding Remarks

In this paper a novel image hashing technique is presented. We accredit the success of the proposed deep hashing to the following aspects: 1) it jointly does the feature engineering and hash function learning, rather than feeding hand-crafted visual features to hashing algorithms, 2) the proposed exponential loss function excellently fits the paradigm of mini-batch based training and the treatment in Eqn. (10) naturally encourages inter-bit complementarities, and 3) to combat the under-sampling issue in the training phase, we introduce the idea of two-stage supervised pre-training and validate its effectiveness by comparisons.

Our evaluation on four video segmentation datasets showed that we achieve state-of-the-art performance except for the FBMS dataset due to the missing annotation of static objects.

Runtime for the video segmentation and its components in seconds per frame.

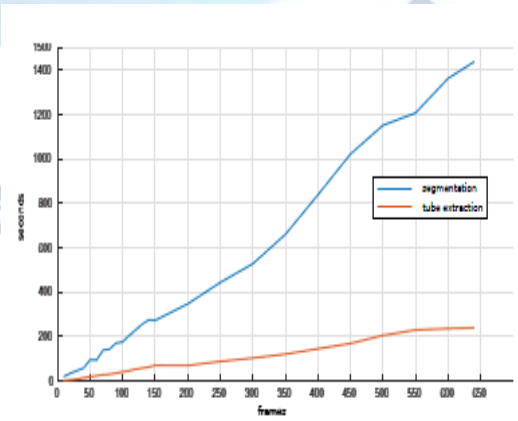

Fig. 9: The runtime of the segmentation (blue) and tube (red) scales linear with the number of frames. Detection,

optical flow and super pixel computation take constant time per frame.

Our comprehensive quantitative evaluations consistently demonstrate the power of deep hashing for the data hashing task. The proposed algorithm enjoys both scalability to large training data and millisecond-level testing time for processing a new image. We thus believe that deep hashing is promising for efficiently analyzing visual big data

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] Achanta, R. ; Shaji, A. ; Smith, K. ; Lucchi, A. ; Fua, P. ; Susstrunk, S. :SLIC Super pixels Compared to State-of-the-Art Super pixel Methods. In: IEEETrans. Pattern Anal. Mach. Intell. 34 (2012), Nov., Nr. 11

[2] Badrinarayanan, V. ; Budvytis, I. ; Cipolla, R. : Mixture of Trees ProbabilisticGraphical Model for Video Segmentation. In: International Journal of ComputerVision (2013).

[3] Endres, I. ; Hoiem, D. : Category Independent Object Proposals. In: European Conf. on Computer Vision (ECCV), 2010, S. 575-588.

[4] Girshick, R. ; Donahue, J. ; Darrell, T. ; Malik, J. : Rich feature hierarchiesfor accurate object detection and semantic segmentation. In: IEEE Conference onComputer Vision and Pattern Recognition (CVPR), 2014.

[5] Hartmann, G. ; Grundmann, M. ; Hoffman, J. ; Tsai, D. ; Kwatra, V. ; Madani, O. ; Vijayanarasimhan, S. ; Essa, I. ; Rehg, J. ; Sukthankar, R. :Weakly Supervised Learning of Object Segmentations from Web-scale Video. In:European Conf. on Computer Vision (ECCV), 2012

[6] Hua, Y. ; Alahari, K. ; Schmid, C. : Occlusion and motion reasoning for long-term tracking. In: European Conf. on Computer Vision (ECCV), 2014

[7] Jain, S. D. ; Grauman, K. : Supervoxel-Consistent Foreground Propagation inVideo. In: European Conf. on Computer Vision (ECCV), 2014, S. 656-671

[8] Keuper, M. ; Andres, B. ; Brox, T. : Motion Trajectory Segmentation viaMinimum Cost Multicuts. In: IEEE International Conference on Computer Vision(ICCV), 2015

[9] Li, F. ; Kim, T. ; Humayun, A. ; Tsai, D. ; Rehg, J. M.: Video Segmentation byTracking Many Figure-Ground Segments. In: IEEE International Conference onComputer Vision (ICCV), 2013

[10] Long, J. ; Shelhamer, E. ; Darrell, T. : Fully Convolutional Networks forSemantic Segmentation. In: CoRR abs/1411.4038 (2014)

[11] Nagaraja, N. ; Schmidt, F. ; Brox, T. : Video Segmentation with Just a FewStrokes. In: IEEE International Conference on Computer Vision (ICCV), 2015.

[12] Tang, K. ; Sukthankar, R. ; Yagnik, J. ; Fei-Fei, L. : Discriminative SegmentAnnotation in Weakly Labeled Video. In: IEEE Conference on Computer Visionand Pattern Recognition (CVPR), 2013

[13] Wang, H. ; Wang, T. : Primary object discovery and segmentation in videos viagraph-based transductive inference. In: Computer Vision and Image Understanding -143 (2016)

[14] Weinzaepfel, P. ; Harchaoui, Z. ; Schmid, C. : Learning to Track for Spatio Temporal Action Localization. In: IEEE International Conference on ComputerVision (ICCV), 2015

[15] Yang, J. ; Price, B. L. ; Shen, X. ; Lin, Z. L. ; Yuan, J. : Fast AppearanceModeling for Automatic Primary Video Object Segmentation. In: IEEE Trans. onImage Processing (2016), Feb

[16] Yang, Y. ; Sundaramoorthi, G. ; Soatto, S. : Self-Occlusions and Dis-occlusions in Causal Video Object Segmentation. In: IEEE International Conferenceon Computer Vision (ICCV), 2015

[17] Zhang, D. ; Javed, O. ; Shah, M. : Video Object Segmentation through SpatiallyAccurate and Temporally Dense Extraction of Primary Object Regions. In: IEEEConference on Computer Vision and Pattern Recognition (CVPR) 0 (2013), S.628{635. -ISSN 1063{6919

[18] A. Andoni, P. Indyk, H. L. Nguyen, and I. Razen-shteyn. Beyond locality-sensitive hashing. CoRR, abs/1306.1547, 2013.

[19] M. M. B. C. Strecha, A. M. Bronstein and P. Fua. LDAHash: improved matching with smaller descriptors. IEEE Transactions on Pattern Analysis and Ma-chine Intelligence, 34(1), 2012.

[20] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. IEEE Trans. Pattern Anal. Mach. Intell., 35(12):2916–2929, 2013.

[21] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagined classification with deep convolutional neural networks. In NIPS, 2012.

[23] 00H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural net-works. In CVPR, 2015.

[24] G. Lin, C. Shen, and A. van den Hengel. Supervised hashing using graph cuts and boosted decision trees. IEEE Trans. Pattern Anal. Mach. Intell., 37(11):2317– 2331, 2015.

[25] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In CVPR, 2015.

[26] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10, 000 classes. In CVPR, 2014.

[27] M. Norouzi, D. J. Fleet, and R. Salakhutdinov. Hamming distance metric learning. In NIPS, 2012.

[28] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In AAAI, 2014

[29] Y. Mu, J. Shen, and S. Yan. Weakly-supervised hashing in kernel space. In CVPR, 2010.

[30] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In CVPR, 2010.