Johnson-Lindenstrauss for Approximate Optical Flow

Lahav Lipson, Alexander Raistrick, Darby Haller

December 2021

1 Introduction

Optical Flow is the task of estimating per-pixel correspondences between two images. That is, given a pair of input images, we want to match each pixel in the left image with one on the right, such that both pixels correspond to the same point in the scene.

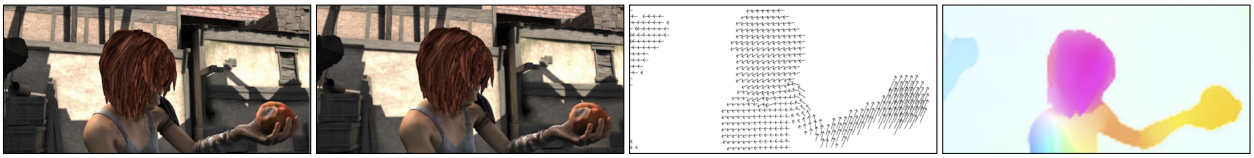
The ability to estimate correspondences between pairs of images is critical for applications in which the main objective is to reconstruct scene geometry from only camera input. Specifically, if the scene is assumed to be static, accurate correspondences between photographs taken from known viewpoints can uniquely define a 3D point cloud for the scene. Equivalently, if the scene geometry is known but the camera trajectory is not, it can be uniquely defined using (a small number of) correspondences. This latter problem is equivalent to the problem of orienting objects subject to rigid motion while the camera locations are known. The usefulness of optical flow for 3D scene understating and/or reconstruction problems has motivated the study into better methods for estimating correspondences between images.

Having a motion vector per-pixel is useful for many applications as it is predictive of the scene state at future time steps. Fast moving objects, thin structures, and non-rigid motion all make the problem difficult to solve, however. In practice, many indoor surfaces are textureless or otherwise have very low-frequency details, causing ambiguity about which pixels correspond with one another. Classical methods solve such problems by incorporating priors about scene geometry, or interpolate between values with higher confidence. Furthermore, cameras often contain motion blur, which further complicate the problem.

Optical Flow is even grounded in psychology and human perception through its connection to "Looming", a measurable physiological reaction to an object's visual expansion towards the eye. Optical Flow is said to be a 'low level' task, meaning that it requires us to make estimates of fundamental properties of an unconstrained scene (i.e., it's motion) rather than structured, high-level object labels or other semantic information.

Traditional approaches tackle optical flow as an optimization optimization problem over hand-designed image features [1, 2]. Given some procedure to extract a feature descriptor vector for each pixel, one must compute a matching between pixels. Such methods typically define a matching loss penalizes pixels with dissimilar features, while simultaneously enforcing hand-chosen priors. For instance, some methods impose that the flow should be piece-wise smooth, should not contain too extreme motions, or should be cycle-consistent, meaning applying the flow model in both directions leads back to the source pixel.

While classical methods use hand-designed features descriptors such as SIFT [3] and ORB [4], modern approaches instead use learned features. RAFT [5], a state-of-the-art deep learning method for optical flow,



(a) Image 1 (b) Image 2 (c) Optical Flow Vectors (d) Optical Flow Visualization

Figure 1: Example Optical Flow Input/Output Given two input images (a, b), we must estimate pixel-wise motion vectors (c). We can compactly represent direction and magnitude of motion with color hue and saturation, as shown in (d)

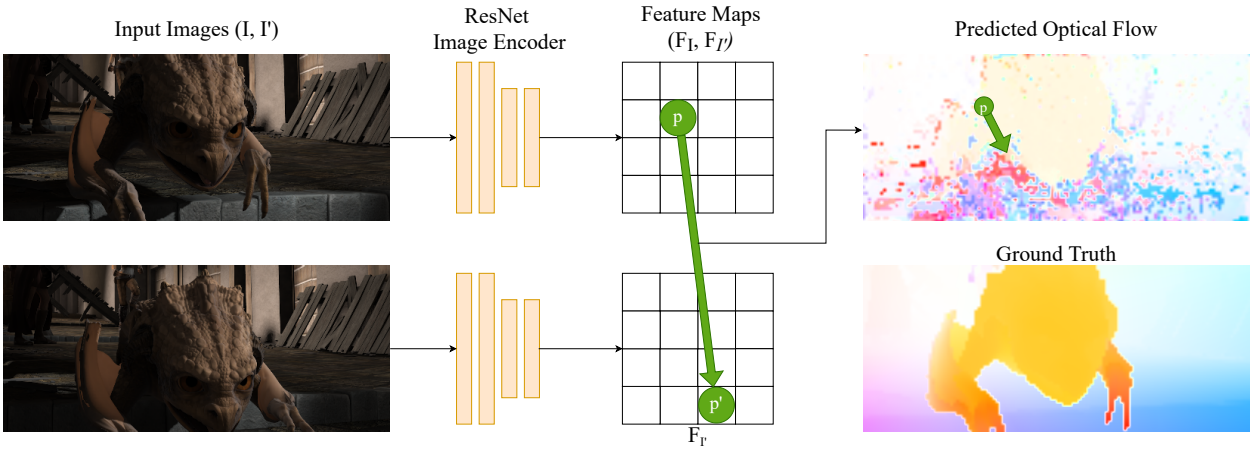


Figure 2: Simplified Model Architecture Given two input images, we produce ResNet [6] feature maps, and directly decode optical flow by matching each pixel p to the pixel p' with closest feature distance $\|F_I(p) - F_{I'}(p')\|$

produces $256 \times H \times W$ feature maps for each image, and uses these per-pixel feature vectors to inform iterative updates to a running estimate of the optical flow between the two input images. Specifically, an optical flow estimate is used to sample and compare pairs of feature vectors between the images, serving as a kind of self-consistency check.

For high resolution images, generating and storing these high dimensional feature maps becomes a significant memory bottleneck. Currently, RAFT works around this issue by operating at reduced resolution, as low as 1/8th of the original image size. This greatly reduces the memory bottleneck, but limits the detail that can be achieved by the model, and leads to incorrect predictions for thin structures.

In this paper, we investigate the application of Johnson-Lindenstrauss (JL) maps to reduce the memory cost of computing and storing these feature maps by reducing the feature dimension of the feature vectors. Under a simplified proxy model, we produce theoretical bounds regarding the effect of JL on evaluation metrics. We justify this result empirically by evaluating our method on a real optical flow benchmark for various JL-reduced feature dimensions. Finally, we discuss the implication of our results for a downstream application of optical flow, and show initial theoretical results on how JL reduction could be used in an online setting to process a video of unknown length under fixed memory budget.

2 Preliminaries

2.1 Simplified Model Architecture

Unfortunately, the operations applied to the correlation volume computed by RAFT are complex and not conducive to direct analysis using Johnson-Lindenstrauss. As a simplyifying assumption, we will instead analyze an architecture which derives optical flow by matching the features directly (as opposed to using more deep network layers). We assume that the network produces feature maps $F_I, F_{I'}$ of shape $C \times W \times H$, and computes optical flow by matching each pixel p in the first image to the pixel p' in the second image with the closest feature distance $\|F_I(p) - F_{I'}(p')\|$. This architecture is depicted in Figure 2.

Essentially, we assume that any complex architecture components take place *before* matching all pairs of feature vectors, and that the resultant features directly represent the network's estimate of optical flow.

2.2 Optical Flow Evaluation

To evaluate the performance of an optical flow system, we one must obtain videos with known ground truth optical flow, and decide a set of metrics to measure performance.

Obtaining ground truth optical flow for arbitrary real-world videos is difficult in-practice. By definition, we must track every pixel between frames. In theory we could cover surfaces densely with motion tracking

markers, but in reality this is impractical and would bias the dataset by introducing easily trackable visual features.

Instead, we use SINTEL [7], a widely adopted flow benchmark composed of computer generated images, for which we can obtain ground truth from graphics data. SINTEL is derived from an open source animated short-film, and contains suitably complex motion, textures and rendering effects.

Optical Flow accuracy is measured using two categories of error metrics. The first is End-Point-Error (EPE), which reports the mean euclidean distance between optical flow predictions and ground truths. Second, it is common to report N-pixel error (for N=1, 3, 5), which represents the percentage of flow vectors which are within N pixel-distance of the ground truth.

For our purposes, N-pixel error is a more useful metric. Achieving state of the art EPE numbers requires not just matching corresponding pixels, but predicting flow vectors accurate to distances less than a pixel across. In applying JL, we accept that our flow results will not be accurate to this degree, but hope that we still obtain approximately correct results. For this reason, we will focus on error metrics such as 5-pixel error, which better captures this goal.

2.2.1 Occlusion

Since the goal of optical flow is to match pixels between images, we can run into issues when a pixel's match is not in the other image. This can happen because it went out of frame due to the motion of the camera, or became occluded for one reason or another. To remedy this, benchmarks either compare only on pixels which are visible in both images, or require that methods make predictions even for unseen pixels. In order to do the latter, the method must extrapolate based on what is visible using smoothness assumptions or learned priors about the data (such as object rigidity).

3 Theoretical results

Optical flow estimators output offset vectors in the Cartesian image-space plane based on input features; our goal is to reduce the dimensionality of these input features without perturbing the output offset vectors too much. Johnson-Lindenstrauss maps approximately preserve input feature-space distances, but we need further distributional assumptions to translate that into guarantees on the output (image-space offset preservation).

To have any hope of doing well on dimension-reduced optical flow, we first need to be able to do well on unreduced optical flow. Our model takes in a source pixel and outputs the closest target pixel in feature space. So before dimension reduction, our source pixels need to be closer in feature space to their corresponding ground truth target pixels than to all other target pixels. It is also acceptable if a source pixel is instead closest in feature space to a target pixel that is very close to (but not exactly) the correct target pixel in image space; guessing one or two pixels off is still good.

But image-space approximate maximality is not quite enough for JL to work; we also need a score-space margin separating the matching scores of the good close by pixel from those of the bad far off pixels. To see this, imagine if the above condition held in the unreduced setting, but with nearly no margin. For example, what if the correct target pixel was in the top left corner of the image, and its features had distance 1 from the source pixel's features, but there were a bunch of (independently) incorrect target pixels in the bottom right corner of the image that were only .00000000000001 farther away from the source pixel than the correct target pixel was? Then, with high probability, applying a Johnson-Lindenstrauss map would make our model incorrectly predict a flow to the bottom right of the image rather than to the top left.

In summary, not only do we need the best match to be within a certain image-space radius of the correct target pixel, we also need it to be a significant margin better than all incorrect pixels outside of that image-space radius. Definition 1 formalizes this notion, parameterized by an image space tolerance radius r and feature space distance margin ϵ. Note that it applies to ground truth pixel pairs, not entire image pairs - optimal bounds are sometimes achieved by choosing different ϵ and r's for different ground truth pixel pairs in the same image pair.

Let $||\cdot||_i$ denote the euclidean norm in Cartesian image space, and $||\cdot||_f$ denote the euclidean norm in feature space. For any image-space coordinates p, let $F_I(p)$ equal the features present at pixel p in image I.

Definition 1 *In the ground truth optical flow, every pixel p in the source image I flows to a corresponding pixel p' in the target image I'. For any image-space radius r, let $p^{*\prime} := argmin_{y \in N_r(p')} ||F_I(p) - F_{I'}(y)||_f^2$. Then, let ϵ be (arbitrarily close to) the largest positive margin such that $||F_I(p) - F_{I'}(p^{*\prime})||_f^2 < \frac{1-\epsilon}{1+\epsilon} ||F_I(p) - F_{I'}(y)||_f^2 \forall y \in I' \setminus N_r(p')$.*

ϵ is a function of p, p', I, I', and r, and for certain combinations of those inputs, it may not even exist (if the closest pixel in feature space is not within an r radius of p'). The smaller the radius, or the larger the score margin, the better accuracy guarantees we can make for dimension reduced optical flow, but in general, increasing the margin requires us to increase the radius. So each ground truth pixel pair has an associated $r - \epsilon$ curve, which, when m is fixed, trades off between how tight of a neighborhood we are trying to ensure we map into, and how likely it is that we map into that neighborhood (though this tradeoff only exists theoretically - in reality, all of these neighborhood likelihoods exist simultaneously, it's just that we can only use one of them at a time in the evaluation metric bounds we produce). Different evaluation metrics will be optimally bounded by using different points on any given $r - \epsilon$ curve.

Now, we will show upper bounds on the probability that Johnson-Lindenstrauss maps perturb flow predictions for source pixels too far away from their correct corresponding target pixels. Because we only need to get perturbation probability bounds on $O(n)$ vector distances at any one time for n vectors, we get a better probability of success than naively applying the Johnson-Lindenstrauss likelihood of not changing the $O(n^2)$ pairwise distances between n vectors.

Theorem 2 *Fix a source pixel $p \in I$ and its corresponding target pixel $p' \in I'$. The chance that p gets mapped more than r away from p' after Johnson-Lindenstrauss dimension reduction is no more than $(H \times W - \pi r^2 + 1) \times 2e^{-m\epsilon^2/8}$.*

Proof. p doesn't get mapped more than r away from p' as long as $||\Pi(F_I(p) - F_{I'}(p^{*\prime}))||_f^2 < ||\Pi(F_I(p) - F_{I'}(y))||_f^2 \forall y \in I' \setminus N_r(p')$. By the definition of ϵ, this is ensured as long as none of the $H \times W - \pi r^2 + 1$ squared vector norms of interest are perturbed by more than a factor of ϵ (though better results could be achieved by using one-sided Johnson-Lindenstrauss tail bounds, and also by using a different ϵ' for $p*'$ and ϵ_y for all $y \in I' \setminus N_r(p')$, and optimizing over their feasible combinations). Then, for all $y \in I' \setminus N_r(p')$,

$$||\Pi(F_I(p) - F_{I'}(p^{*\prime}))||_f^2 \leq \tag{1}$$

$$(1+\epsilon)||F_I(p) - F_{I'}(p^{*\prime})||_f^2 < \tag{2}$$

$$(1+\epsilon)\frac{1-\epsilon}{1+\epsilon}||\Pi(F_I(p) - F_{I'}(y))||_f^2 = \tag{3}$$

$$(1-\epsilon)||\Pi(F_I(p) - F_{I'}(y))||_f^2 \leq \tag{4}$$

$$||\Pi(F_I(p) - F_{I'}(y))||_f^2 \tag{5}$$

From the lecture notes, for any feature vector v, $Pr[||\Pi v||_f^2] \notin (1 \pm \epsilon)||v||_f^2 \leq 2e^{-m\epsilon^2/8}$, so by the union bound, the probability that any of the $H \times W - \pi r^2 + 1$ squared vector norms of interest are are perturbed by more than ϵ is $\leq (H \times W - \pi r^2 + 1) \times 2e^{-m\epsilon^2/8}$. ■

Now, depending on what distributions of per-pixel ϵ as functions of r our data provide, we can use the above bound to provide theoretical guarantees on the expected performance of Johnson-Lindenstrauss reduced optical flow for EPE (the average distance between the ground truth target and the target our model predicts) and Acc_r (the fraction of times our model guesses within r pixels of the ground truth target).

Theorem 3 $E_\Pi[EPE_{JL}] \leq E_{p,p' \in D}[max_r[max(0, (1 - (H \times W - \pi r^2 + 1) \times 2e^{-m\epsilon^2/8})) \times r + min(1, (H \times W - \pi r^2 + 1) \times 2e^{-m\epsilon^2/8})||(1 - |p_x|, 1 - |p_y|)||_i]]$, *where ϵ varies based on p, p', I, I', and r, and D is the distribution over all ground truth pixel pairs in all image pairs.*

Proof. For any pixel pair p, p' and radius r such that ϵ exists, if p is mapped into an r neighborhood of p', then its error is no more than r. By the above theorem, the chance of this happening is at least $1 - (H \times W - \pi r^2 + 1) \times 2e^{-m\epsilon^2/8}$ (and also at least 0 by the rules of probability). The only other possibility is that it gets mapped outside of the r neighborhood of p', in which case its error can't be any worse than its distance to the furthest corner, $||(1 - |p_x|, 1 - |p_y|)||_i$ (assuming the image size is $[0, 1] \times [0, 1]$). The

probability of this happening is no more than $(H \times W - \pi r^2 + 1) \times 2e^{-m\epsilon^2/8}$ (and also no more than 1 by the rules of probability). As the maximizing r will never be more than the distance to the farthest corner, the expectation of the error is no more than $max(0, (1 - (H \times W - \pi r^2 + 1) \times 2e^{-m\epsilon^2/8})) \times r + min(1, (H \times W - \pi r^2 + 1) \times 2e^{-m\epsilon^2/8}) ||(1 - |p_x|, 1 - |p_y|)||_i$. Because this is true for all r such that ϵ exists, it is true in particular for the r that maximizes this term. By linearity of expectation, the expectation over the distribution of pixel pairs follows. ■

Theorem 4 *For any pixel-space margin r, $E_\Pi[Acc_{r,JL}] \geq E_{p,p' \in D}[max(0, 1 - (H \times W - \pi r^2 + 1) \times 2e^{-m\epsilon_r^2/8})]$, where ϵ varies based on p, p', I, I' (and is taken to be $-\infty$ if it does not exist for r), and D is the distribution over all ground truth pixel pairs in all image pairs.*

Proof. Theorem 2 states that the chance that a given source pixel is mapped within r of its ground truth target pixel is no less than $1 - (H \times W - \pi r^2 + 1) \times 2e^{-m\epsilon_r^2/8}$, and due to the rules of probability, that chance also can't be any less than 0. $Acc_{r,JL}$ is defined as $E_{p,p' \in D}[\text{p mapped within } r \text{ of } p']$, so by linearity of expectations,

$$E_\Pi[Acc_{r,JL}] = \tag{6}$$

$$E_\Pi[E_{p,p' \in D}[\text{p mapped within r of p']}] = \tag{7}$$

$$E_{p,p' \in D}[E_\Pi[\text{p mapped within r of p']}] \geq \tag{8}$$

$$E_{p,p' \in D}[max(0, 1 - (H \times W - \pi r^2 + 1) \times 2e^{-m\epsilon_r^2/8})] \tag{9}$$

■

4 Online SLAM

Recently, DROID-SLAM [8] made use of optical flow as an intermediate step for Deep Visual SLAM. Simultaneous Localization and Mapping is a core problem in robotics. Critically, it requires us to process images online, without advance knowledge of the total length of a video. This requires that we retain sufficient information from each image to be able to compute optical flow between every pair of images seen so far. In practice, we simply store each feature map F_I, which eliminates repeated work of re-running the image encoder.

However, since all the features must be stored in memory, this places a limit on the length of a video that can be processed with a fixed memory budget. In an online setting, where we don't know how many frames we will have to process ahead of time, it is hard to choose the embedding dimension m of the Johnson-Lindenstrauss map. If we choose m to be very small, and it turns out that we didn't have to process many video frames, we would have sacrificed a lot of accuracy unnecessarily. On the other hand, if we choose m to be too big, but it turns out that we have to process many video frames, our system will run out of memory. Ideally, m would start high, and decrease as we received more and more frames, always using as much memory as was available. But we can't go back and apply a new, smaller Johnson-Lindenstrauss map to the original feature vectors - those have already been cleared out of memory. However, in the following section, we will show that simply dropping $m - m'$ indices across all dimension-reduced stored features works just as well as recomputing new dimension-reduced feature vectors using a new, smaller Johnson-Lindenstrauss map in $\mathbb{R}^{m' \times d}$, while only requiring access to the latest set of dimension reduced vectors.

Specifically, we will show that taking any m' elements of the output of a Johnson-Lindenstrauss map (and rescaling appropriately) is still a Johnson-Lindenstrauss map, but to a lower dimensional space. Then, getting the result of this new, smaller Johnson-Lindenstrauss map applied to the original features is as simple as dropping some of the indices from the latest set of dimension reduced vectors and rescaling.

Theorem 5 *Let $i_1 ... i_{m'}$ be any m' indices in $[m]$. If $\Pi \in \mathbb{R}^{m \times d}$ is a Johnson-Lindenstrauss map, then $f(x) = \frac{\sqrt{m}}{\sqrt{m'}}(\Pi x)[i_1 ... i_{m'}]$ is still a Johnson-Lindenstrauss map, though now in $\mathbb{R}^{m' \times d}$.*

Proof. A Johnson-Lindenstrauss map $\Pi \in \mathbb{R}^{m \times d}$ is a Gaussian random matrix where each variable is sampled from $N(0, \frac{1}{\sqrt{m}})$. If Π is a Gaussian random matrix where each variable is sampled from $N(0, \frac{1}{\sqrt{m}})$,

and $\Pi' \in \mathbb{R}^{m' \times d}$ is any m' rows from $\frac{\sqrt{m}}{\sqrt{m'}}\Pi$, then $\Pi' \in \mathbb{R}^{m' \times d}$ is a Gaussian random matrix where each variable is sampled from $N(0, \frac{1}{\sqrt{m'}})$, and is therefore a Johnson-Lindenstrauss map. ■

So, given this fact, and our established results for the performance of JL-reduced optical flow, we can devise a strategy that slowly degrades quality once memory is exceeded, rather than crashing altogether. Specifically, whenever adding a new image's features to memory would exceed our capacity, simply drop features dimensions from all existing images to make space for it, and rescale according to Theorem 5 to preserve feature distances according to JL.

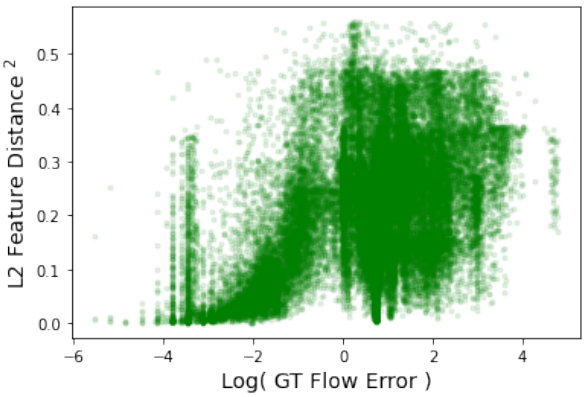


Figure 3: The algorithm works because corresponding pixels have small distances in feature space. The measurements were obtained by randomly sampling 10% of the top 10k closest matches in each image pair, across 40 image pairs chosen i.i.d. "GT-Flow-Error" is measured in pixels.

5 Experiments

We investigate the effectiveness of using a JL-Map to reduce the memory requirements of storing feature maps for predicting correspondences (i.e. optical flow) between two images. Specifically, we want to be able to take two $C \times H \times W$ feature maps and shrink them to $k \times H \times W$ where $k < C$, such that corresponding pixels still have similar features.

We evaluate on SINTEL [7] using the 5-px error metric as discussed in Section 2.2. We extract feature maps using a standard ResNet [6] convolutional neural network. This produces feature maps of dimension 55×128 for each image. In order to avoid training the model entirely from scratch, we initialize the network using weights released publicly by the authors of RAFT [5], which uses the same ResNet architecture for it's feature extraction stage.

Our matching algorithm assigns each pixel in the left image to the pixel in the right image whose feature has the highest inner-product with its own. This is a standard similarity metric used in optical flow literature. In order to keep our method simple and easy to analyze theoretically, we do not incorporate any additional priors such as cycle consistency or smoothness. As shown in Figure 4, this leads to large changes in optical flow between adjacent pixels.

5.1 Empirical Results

We compare the effects of three methods for dimensionality reduction in Figure 5. We find that average pooling of adjacent feature dimensions or randomly selecting a subset of feature dimensions preserves accuracy slightly better than using a JL-Map. This is likely due to the JL-Map arbitrarily scaling the relative importance of different feature dimensions. In all cases, we find that one can trade-off about 10% of accuracy in exchange for a 4x reduction in feature dimensions, i.e. $4\times$ less memory usage.

In general, we also observe that, across all pairs of pixels, corresponding pixels have similar feature vectors. Similarly, non-corresponding pixels have dissimilar feature vectors. See Figure 3. This implies that the ResNet backbone of RAFT learned to approximate visual similarity in the feature space.

6 Discussion

Overall, we have shown that under significant simplifying assumptions, it is possible to use Johnson-Lindenstrauss maps to significantly reduce the memory and compute overhead of computing and matching features for optical flow. We obtain a theoretical lower bound, then achieve an even stronger empirical result on a popular optical flow benchmark [7] showing that one may reduce the dimensionality of our feature vectors by up to 80% with only incremental drops in performance.

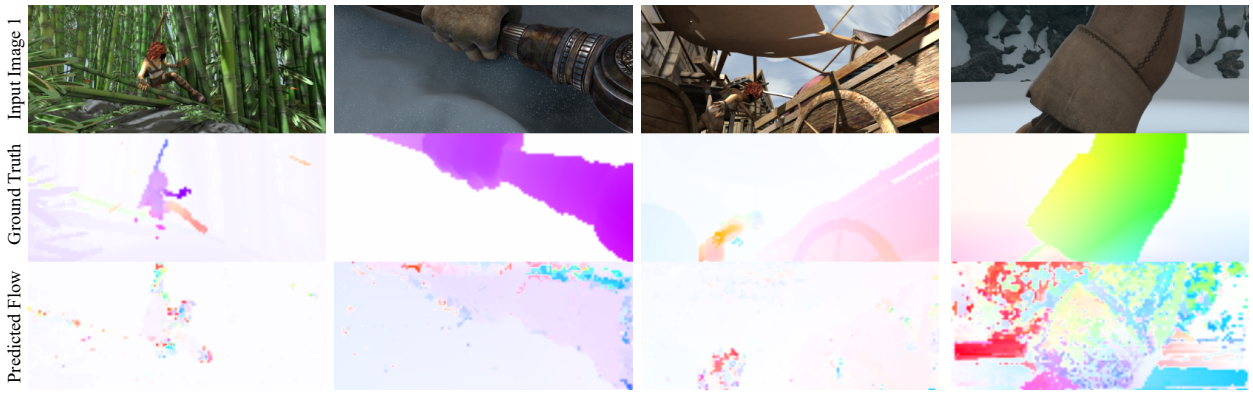


Figure 4: Example Optical Flow Predictions. For each example SINTEL test frame, we show one of the two RGB input frames (top row), the ground truth optical flow (middle row) and our prediction using feature matching.

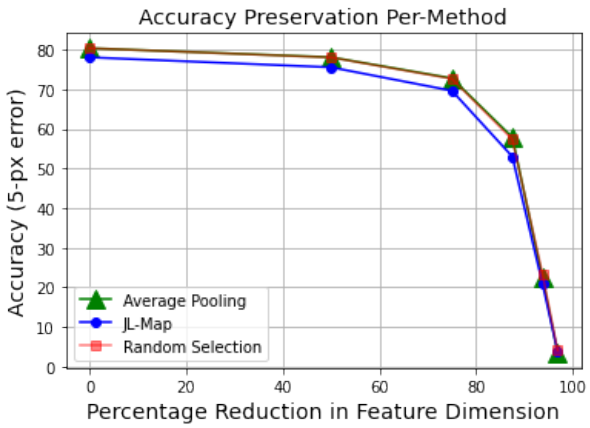


Figure 5: We compare the effects three methods of dimensionality reduction on the accuracy of the matchings. Randomly selecting the feature dimensions to preserve (red) performs as well as averaging adjacent feature dimensions (green). Using a JL-Map (blue) performs slightly worse, potentially because because it arbitrarily scales the relative importance of different feature dimensions. In all cases, we see approximately a 10% drop in accuracy in exchange for a 4x drop in feature size.

We also find however that naive solutions such as dropping random dimensions, or typical deep learning operations such as 'average pooling' bins of dimensions together work just as well. We anticipate with a revised training loss (e.g. a Triplet Loss) designed specifically to promote a margin in feature distance we could improve results, however we leave this to future work.

We see the most significant potential for this method as an intermediate step in dense visual slam. We show that we can apply JL once to the initial features of each image, then drop feature dimensions online as new images are received in order to produce approximate flow estimates for a video of unknown length.

More broadly, our results are interesting as an initial analysis of whether Johnson-Lindenstrauss can be applied to a broader class of vision architectures. Our simplified optical flow model is a specific instance of associative embedding [9]. Associative embedding is a task-agnostic architecture used to represent any association between pixels by matching nearest feature vectors. For example, one can represent semantic segmentation (grouping all pixels of the same object class), or multi-human pose prediction (grouping detected joints into coherent human skeletons) in this setting. While many instances of associative embedding are less dense than optical flow (i.e, they do not require a prediction for every pixel), our results provide some evidence that JL could be applied for special cases where large numbers of features must be retained in memory or matched simultaneously.

References

[1] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *ARTIFICAL INTELLIGENCE*, 17:185–203, 1981.

[2] Christian Banz, Sebastian Hesselbarth, Holger Flatt, Holger Blume, and Peter Pirsch. Real-time stereo vision system using semi-global matching disparity estimation: Architecture and fpga-implementation. In

2010 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, pages 93–101. IEEE, 2010.

[3] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[4] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[5] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. *CoRR*, abs/2003.12039, 2020. URL `https://arxiv.org/abs/2003.12039`.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[8] Zachary Teed and Jia Deng. DROID-SLAM: deep visual SLAM for monocular, stereo, and RGB-D cameras. *CoRR*, abs/2108.10869, 2021. URL `https://arxiv.org/abs/2108.10869`.

[9] Alejandro Newell and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. *CoRR*, abs/1611.05424, 2016. URL `http://arxiv.org/abs/1611.05424`.