CHAPTER 36

Regression Methods for Stereo and Optic Flow

A *correspondence algorithm* has two images, $\mathcal{I}_1$ and $\mathcal{I}_2$, and looks for $\mathbf{v}$ so that

$$\mathcal{L}_{\text{photo}}(\mathcal{I}_1(\mathbf{x}+\mathbf{v}), \mathcal{I}_2(\mathbf{x}))$$

where $\mathcal{L}_{\text{photo}}$ is a photometric loss that compares its two arguments in some way. If $\mathbf{v}$ is heavily constrained by geometry, the algorithm is solving a stereo problem, otherwise, it is solving an optic flow problem. This means that there are very strong similarities between learned algorithms for stereo and for optic flow. I am using the term "correspondence algorithm", so I don't have to write "stereo and optic flow" repeatedly.

Stereo and flow problems differ from straightforward image-to-image problems in one important way. You have to find which of $\mathcal{I}_1(\mathbf{x}+\mathbf{u}_1)$ and $\mathcal{I}_1(\mathbf{x}+\mathbf{u}_2)$ is more like $\mathcal{I}_2(\mathbf{x})$. Corresponding points might be quite far apart – equivalently, $\mathbf{u}_1$ or $\mathbf{u}_2$ or both might be quite big. This means you can't do flow or disparity prediction effectively by just stacking two images, passing the result into a unet, and training the unet to produce the right field. Try it – if you can make it competitive with recent stereo or flow algorithms, that would be news, because early attempts to do so ended badly.

Instead, you should expect the network architecture to build an internal representation of how well points in one image compare to points in the other. In both problems, iterative and coarse-to-fine reestimation can substantially improve estimates. You should expect to see analogs of these procedures in learned predictors as well.

36.1 COST VOLUMES AND MATCHING

Recall Procedure 32.1, where you compare a window to a reference window of N pixels. I was vague about how to compare. Here is a natural option. Write $\mathbf{v}_1$, $\mathbf{v}_2$ for vectors consisting of the pixel values in the two windows. Write $\mathbf{u}_1 = \mathbf{v}_1 - (1/N)\mathbf{v}_1^T\mathbf{1}$, etc. This is a zero mean version of the window. Then the correlation

$$\frac{\mathbf{u}_1^T\mathbf{u}_2}{\|\mathbf{u}_1\|\|\mathbf{u}_2\|}$$

is one comparison. It is large when the windows are very similar and small when they are different. This measure is often referred to in the vision literature as a normalized correlation. If you compute this number for every pair (location in the left image, location in the right image), you have an example of a cost volume.

Definition: 36.1 *Cost volume*

Given a query image and a reference image, a cost volume is a tensor that represents the similarity between each location in the query image and each of a set of locations in the reference image.

This example is not a particularly good cost volume. It is huge for large images. However, you may not need to use every pair of locations if you are careful, because similarity at a pixel is very like similarity at its neighbor. Normalized correlation isn't ideal as a matching score for correspondence algorithms for two reasons. If, say, $\mathbf{v}_2 = a\mathbf{v}_1 + b\mathbf{1}$, the normalized correlation won't be affected. But this means that making a patch brighter might not change the match score, violating photometric consistency. You can resolve this with two steps. First, instead of using the pixel values to describe the window, use some other feature of the window $\mathbf{f}$, trained to be well behaved. Second, use feature maps to measure similarity.

36.1.1 Building Cost Volumes

It is natural to compute a cost volume using unnormalized correlations of feature vectors. Pass two images $\mathcal{I}_1(\mathbf{x})$ and $\mathcal{I}_2(\mathbf{x})$ through the same encoder. Each yields a feature representation, so $\mathbf{f}_1$, etc. This feature representation may not be on the same pixel grid as the image, but is registered to the image. For example, if the images are $3 \times H \times W$ and the feature representation is $D \times H \times W$, there is a feature at each pixel. If the feature representation is $D \times (H/4) \times (W/4)$, it is still registered to the image. For any pixel, you can identify a feature vector that represents this pixel. Write $P(\mathbf{x})$ for the mapping that takes $\mathbf{x}$ in the image to the corresponding location in the tensor of features (for example, if the features are $D \times (H/2) \times (W/2)$, then $P : (i, j) \to (i/2, j/2)$, and so on **exercises**). You can build a cost volume out of these features in a straightforward way.

Procedure: 36.1 *Building a cost volume*

Choose k, so the correlation window has size $(2k+1) \times (2k+1)$ (k could be 0), and form

$$\mathcal{V}(i, j, u, v) = \sum_{\delta i=-k}^{\delta i=k} \sum_{\delta j=-k}^{\delta j=k} \mathbf{f}_1(i + \delta i, j + \delta j)^T \mathbf{f}_2(u + \delta i, v + \delta j)$$

where i, j and u, v range over the size of the feature tensor. Handle the edges of the image by padding. Now represent the similarity between $\mathcal{I}_1$ at $\mathbf{x}_1$ and $\mathcal{I}_2$ at $\mathbf{x}_2$ by $V(P(\mathbf{x}_1), P(\mathbf{x}_2))$.

It is usual to represent this object as an $(HW) \times H \times W$ tensor. The last two indices identify a location in the query image; the first spans all the locations

in the reference image. This captures the asymmetry between the query image and the reference image – you are matching from query to reference, which is not the same as matching from reference to query (see Section 32.1.4 if you're uncertain). Ideally, this value is large if the two locations match and small if they don't. This works particularly well when you can train the feature vectors to cooperate.

36.1.2 Practical Cost Volumes

A practical cost volume has (a) all relevant comparisons and (b) as few comparisons as reasonably possible. In the case of optic flow, you have to compare every point in $\mathcal{I}_1$ with every point in $\mathcal{I}_2$, so the tensor will have dimension $H \times W \times H \times W$. In the case of stereo, work with rectified images. You need to compare only points along an epipolar line, so the tensor will have dimension $H \times W \times W$ **exercises**. This tensor (in either version) is known as a *cost volume*.

Up until recently, a $(HW) \times H \times W$ tensor was unmanageably large. A reduced cost volume matches each location in the query image to a window of locations in the reference image. Choose some upper bound $d_{\max}$ for the displacement, then compare each location $\mathbf{x}_1$ in $\mathcal{I}_1$ with every location in a $2d_{\max} + 1 \times 2d_{\max} + 1$ window in $\mathcal{I}_2$. It is usually easier to work with an upper bound d_t for displacement in the feature tensor as opposed to $d_{\max}$. This doesn't change the underlying idea, but relations between d_t and $d_{\max}$ are a little obscure **exercises**.

Procedure: 36.2 *Building a reduced cost volume*

Choose k, so the correlation window has size $(2k+1) \times (2k+1)$ (k could be 0). Choose a range d_t and form

$$\mathcal{V}(i,j,u,v) = \sum_{\delta i=-k}^{\delta i=k} \sum_{\delta j=-k}^{\delta j=k} \mathbf{f}_1(i+\delta i, j+\delta j)^T \mathbf{f}_2(i+u+\delta i, j+v+\delta j)$$

where i, j range over the size of the feature tensor and u, v range over $[-d_t, d_t]$. Handle the edges of the image by padding. Now represent the similarity between $\mathcal{I}_1$ at $\mathbf{x}_1$ and $\mathcal{I}_2$ at $\mathbf{x}_2$ by $V(P(\mathbf{x}_1), P(\mathbf{x}_2))$. This is meaningful only if $\mathbf{x}_1$ and $\mathbf{x}_2$ are close enough.

You should think of the resulting object as a block of features, which can be passed through convolutional layers, etc. just like other features.

36.1.3 FlowNet and DispNet: Flow and Disparity with Cost Volumes

FlowNet [] computes an embedding for the two frames, then computes a reduced cost volume to compare the two image representations. Figure 36.1 shows the overall architecture. Each of the two frames is passed through a copy of an image encoder. The comparison layer – which has no learnable parameters – then computes a reduced cost volume for the pair of frames. This block is further encoded by a series of convolutional layers. The result, together with skip connections from

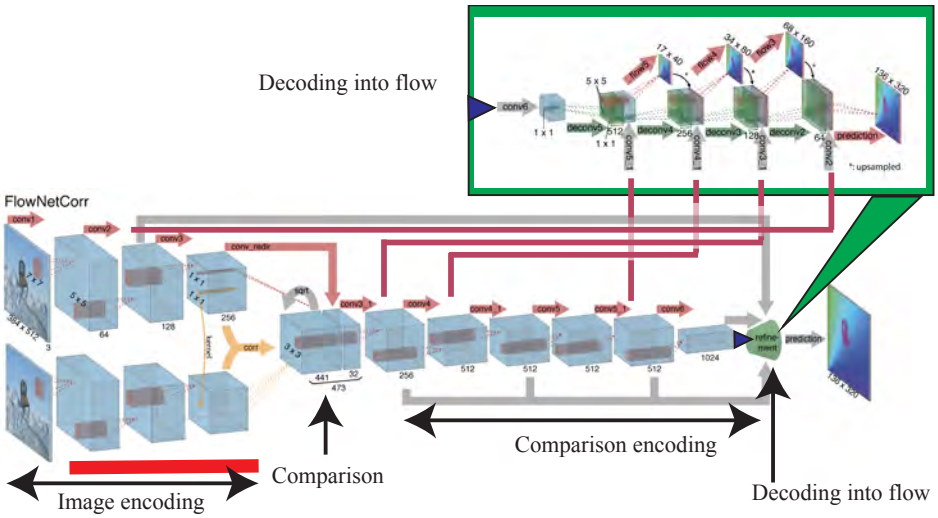


FIGURE 36.1: *The overall architecture of Flownet, a regression based optic flow predictor ([]). The frames pass through an image encoder; the results are compared (details in text), and the comparison further encoded. Finally, the result is decoded into a flow field, using skip connections from the comparison encoder and from the first frame encoder.* Image credit: *Figure 2 of* FlowNet: Learning Optical Flow with Convolutional Networks *by P. Fischer, A. Dosvitskiy, E.Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smag, D. Cremers, and T. Brox*

the comparison encoder and from the first frame encoder, is then decoded into a flow field. FlowNet is trained using tuples (image 1, image 2, flow) in the natural way. Section **??** describes how you get such tuples.

Estimating disparity isn't all that different from estimating flow. Work with rectified image pairs. The cost volume is then smaller because you need compute costs only along the (horizontal) epipolar lines. Decoding a cost volume into a set of disparities should largely follow the lines of decoding into flow. DispNet [] is a version of FlowNet that is adjusted to estimate disparities. Like FlowNet, DispNet is trained using tuples (image 1, image 2, disparity) in the natural way. Again, these are treated in more detail in Section **??**.

36.1.4 3D Convolutions and Soft Argmin for Disparity

Some patterns in images are more common than others (which is what made convolutional encoding work) and depth maps are highly structured (which certainly helps depth from a single image). These properties should mean that there are patterns in cost volumes as well. For example, flat frontal surfaces should produce highly stylized costs **exercises**. This suggests it would be helpful to detect patterns in a cost volume.

Definition: 36.2 *3D convolution*

A 3D convolution takes a tensor $\mathcal{V}$ with three dimensions and a three dimensional kernel $\mathcal{M}$ and produces a three dimensional result, following the pattern of Section 5.1.1. Write $\mathcal{N}$ for the result, and use:

$$\mathcal{N}_{ijk} = \sum_{uvw} \mathcal{I}_{i-u,j-v,k-w} \mathcal{M}_{uvw}.$$

All remarks for convolution apply to 3D convolution. It is a pattern detector; stride and padding are useful; there is an alternative (3D filtering); the difference between 3D convolution and 3D filtering is very seldom of interest; and, very often, people say one and do the other. The definition for multi-channel convolution extends to 3D convolution in the obvious way.

You could now decode a cost volume by applying a series of multi-channel 3D convolution layers, each followed by a ReLU, to it. This allows comparisons at different locations to interact with one another. For example, if planes were common, you should expect this to denoise the comparisons. Each comparison can be re-estimated by looking at its value and that of its neighbors; this is a kind of denoising. The kernels will be learned, as you should expect.

The result is another 3D tensor, and needs to be turned into a disparity map or a flow map. The 3D tensor is intended to represent all relevant comparisons at a location, but you want the system to report the best matching location. The soft argmin (from [**?**]) is a differentiable operation that reports a slightly smoothed estimate of the best location.

Definition: 36.3 *Soft ArgMin*

Given a vector of N values $[v_1, v_2, \ldots, v_N]^T$, write $\sigma_i(\mathbf{v})$ for

$$\frac{e^{-v_i}}{\sum_u e^{-v_u}}.$$

The soft argmin of these values is

$$\sum_i i\sigma_i(\mathbf{v}).$$

You could now build a decoder that consists of a series of multi-channel 3D convolution layers, each followed by a ReLU, with a soft argmin layer at the end. Such decoders are fairly successful at stereo prediction, but can be improved. Note that soft argmin has some issues **exercises**.

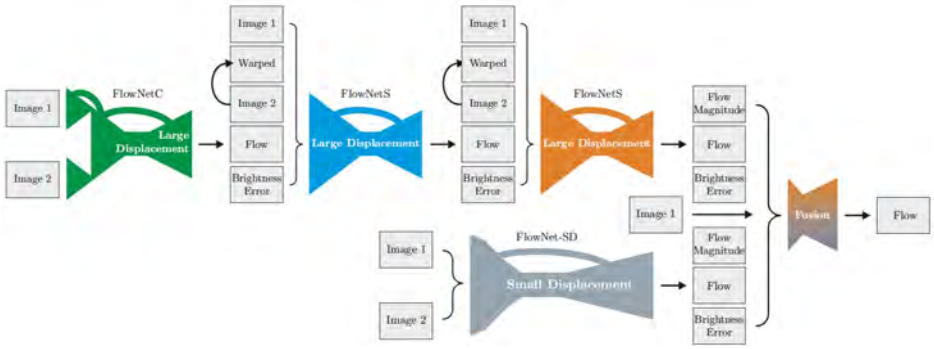


FIGURE 36.2: *Flownet2 obtains significant improvements by composing multiple Flownets. Compute a flow estimate with a flownet; now warp the second image to produce $\mathcal{W}$ (which would be frame 1 if you had the flow right). The magnitude of the residual between frame 1 and the warped version of frame 2 identifies errors in the flow. Provide this error, $\mathcal{I}_1$ and $\mathcal{W}$ to a version of flownet adjusted to accept the extra frames. This produces a revised flow. You can stack these revised versions as required (two seems to be enough). The resulting network is good at estimating large flows. Now build yet another flownet, allowing quite small displacements only. You now have two estimates of flow, so construct a fusion network that accepts the flow estimates and produces a final fused estimate. Show this network $\mathcal{I}_1$, the flow fields, the flow magnitudes, and the brightness errors (Figure* **??***).* Image credit: *Figure 2 of* FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks*, by E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy and T. Brox.*

36.1.5 Improving Cost Volume Algorithms using Residuals

Flownet was the first regression based flow predictor, but it wasn't particularly good. Significant improvements are obtained by using multiple flownets, by analogy with the iterative flow estimation of Section **??**. Start with an instance original flownet, which produces a flow estimate. Use this flow estimate to produce $\mathcal{W}$, a warped version of frame 2. If the flow is correct, the warped estimate will largely look like frame 1, so the magnitude of the residual between frame 1 and the warped version of frame 2 will identify errors in the flow. Compute this brightness error ($\mathcal{E}$), and provide $\mathcal{I}_1$, $\mathcal{I}_2$, $\mathcal{W}$ and $\mathcal{R}$ to a version of flownet adapted to accept the extra frames. This produces a revised flow. You can stack these revised versions as required (two seems to be enough). You should see this stack as a version of iterative reestimation, as in Section **??**.

The resulting network should be good at estimating large flows, but may make errors with small flows. For an example, think about images showing a periodic structure. The flow may be too big by a fixed number of periods because the network is capable of comparing quite widely separated locations. This is easily dealt with. Build yet another flownet, allowing quite small displacements only. You now have two estimates of flow, so construct a fusion network that accepts the flow estimates and produces a final fused estimate. Show this network $\mathcal{I}_1$, the flow fields, the flow

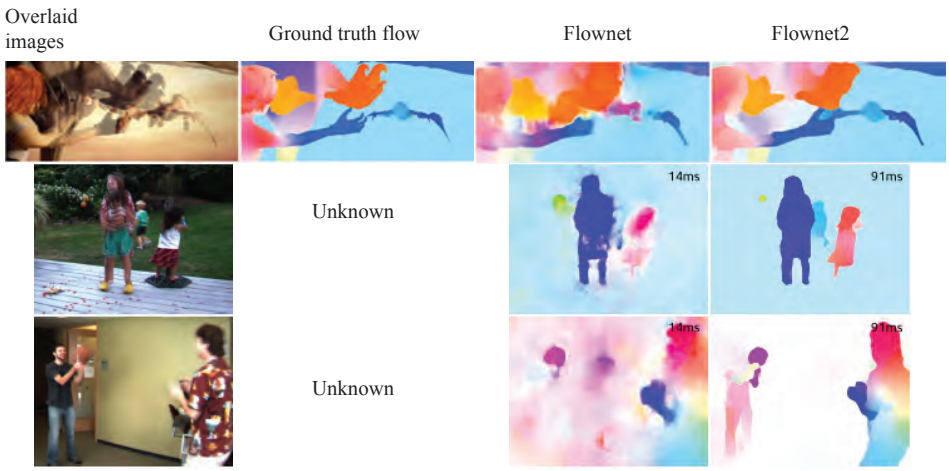


FIGURE 36.3: *Flownet2 predictions are a significant improvement over Flownet predictions. The* **top row** *shows qualitative results on an example image from the SINTEL dataset (Section 36.3). The overlaid image is an average of first and second frame, so you can identify flows by "ghosting" of edges. The direction of a flow vector is shown by hue (what changes from red to yellow to green to purple) and the magnitude by saturation (very small vectors are white, longer are pale colors, longest have intense colors). Notice how much more detail there is in Flownet2 estimates. SINTEL is rendered, so the ground truth flow is known, and you can compare. The* **bottom rows** *show flow predictions for real image pairs, where the ground truth is unknown. Again, Flownet2 produces much more detailed flow fields; they should look like highly plausible flow estimates to you.* Image credit: *Parts of Figure 6 and 7 of* FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks*, by E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy and T. Brox.*

magnitudes, and the brightness errors (Figure **??**). The fused estimate turns out to be a significant improvement over the initial estimate (Figure 36.3).

Refining using warping can be used very successfully to produce improved disparity estimates. You should expect this, because it is a form of coarse-to-fine estimation. Figure **??** illustrates the method of Liang *et al.*[]. You should see this network as having two components. The first produces a disparity estimate by: encoding left and right images, computing a cost-volume, then decoding that with 3D convolution and soft argmin (as in Section **??**). The second refines that estimate to produce a residual. The original estimate and the residual are added to yield the disparity estimate.

The disparity is refined by producing a feature representation of each image, using the same feature embedding for each image (the **orange** blocks on the left of Figure 36.4. The **orange** block on the right uses the disparity estimate to warp one images feature representation so it can be subtracted from the other images feature representation (the circle labelled "Re" in the figure). The result is a residual – the difference between left feature representation and warped right

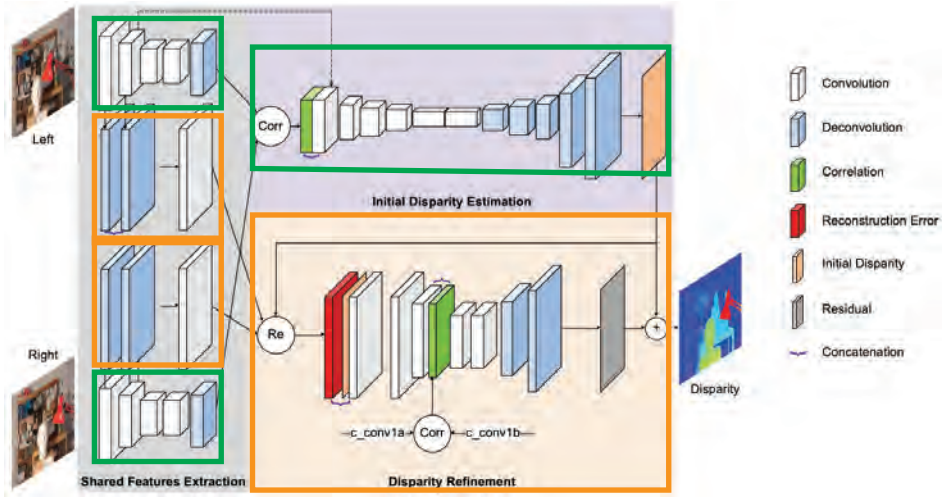


FIGURE 36.4: *Liang* et al.*[] estimate stereo disparity using two coupled networks. The first, outlined in* **green**, *produces a disparity estimate by: encoding left and right images, computing a cost-volume, then decoding that with 3D convolution and soft argmin (as in Section* **??***). This results in a disparity estimate (the* **orange** *block on the* **right***). A second network, outlined in* **orange**, *refines this estimate using procedures in the main text.* Image credit: *Figure 2 of Z. Liang et al., "Stereo Matching Using Multi-Level Cost Volume and Multi-Scale Feature Constancy," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 1, pp. 300-315, 1 Jan. 2021*

feature representation – which contains useful information. If the disparity estimate is correct, then the residual is zero in most locations. A representation of this residual is used to produce a further cost volume; this is refined to produce an estimate of the error in the disparity.

You could – and the authors do – estimate another residual, and yet another. Estimating residuals leads to very strong stereo matching. Figure 36.5 shows some qualitative examples, and the paper contains extensive quantitative information.

As of writing, to my knowledge, the best available regression based flow predictor is RAFT (for Recurrent All-pairs Flow Transforms). The paper is [], and a github page is at `https://github.com/princeton-vl/RAFT`. A similar procedure produces strong stereo disparities as well.

36.1.6 More than Flow or Displarity

A regression predictor that can predict flow well should be able to make a number of other possibly useful predictions. Natural candidates are motion boundaries (places where there is a very sharp change in flow) and occlusions (pixels that are visible in one frame, but not the other). Current datasets generally support these kinds of prediction.



FIGURE 36.5: *Refining a disparity estimate using residuals produces strong qualitative results. The figure shows left and right images and disparity maps obtained like this. Note the absence of the spatial noise in disparity (compare the top row with Figures 33.5; 33.6; and 33.7).* Image credit: *Parts of Figure 6 and 7 of Z. Liang et al., "Stereo Matching Using Multi-Level Cost Volume and Multi-Scale Feature Constancy," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 1, pp. 300-315, 1 Jan. 2021*

36.1.7 RAFT Flow

RAFT uses a network structure that is informed by the key tricks used in optimization-based flow methods (Section **??**): multi-scale representations and iterative flow estimation. Frame 1 and frame 2 are passed through an encoder to produce a feature representation. The correlation layer then computes cost volume comparing all pairs of locations (so if the images are $H \times W$, the correlation layer produces a $(HW) \times H \times W$ data block). Think of the resulting block as a set of fine-scale comparisons. The fine-scale comparisons are pooled to form comparisons at coarser and coarser scales, yielding a multi-scale representation. Comparisons are pooled across the last two dimensions only. This means you can determine where a specific pixel in frame 1 went to in frame 2 at various scales in the frame 2 representation. At each scale, the representation supports a flow arrow at every pixel, but different scales represent that flow arrow at different levels of precision. Knowing the location of the source pixel at the finest resolution yields better estimates of flow on small objects because they aren't subsampled away. Having the flow represented at a variety of precisions supports a coarse-to-fine search, giving a better chance of getting large movements right.

The comparisons are decoded into flow vectors by an incremental procedure. This starts with an estimate of zero flow. Each stage estimates an update to the previous stage's flow. The update is a function of the previous stage's flow, a context feature, and a readout from the multi-scale correlation representation. The context feature is taken from an encoding of the first frame. There are 10 or more (depending on the particular version) of these update stages.

There is a strong analogy between this procedure and traditional optimization procedures. Like this procedure, traditional optimization minimizes a cost function by taking a sequence of steps – equivalently, updating an estimate of the location

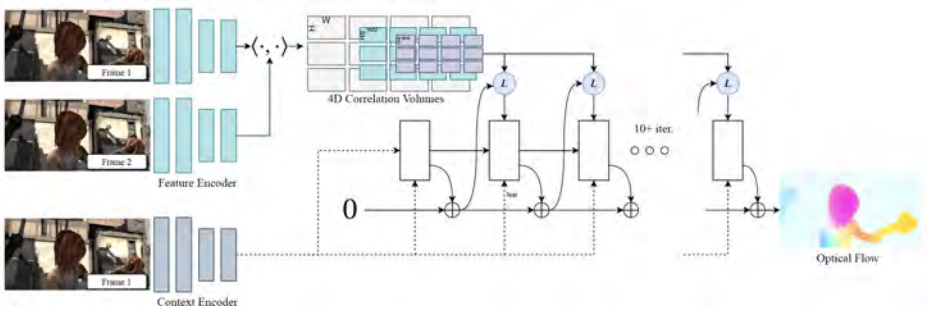


FIGURE 36.6: *RAFT encodes frame 1 and frame 2 to produce a feature representation, then computes an inner product between features for every pair of locations in frame 1 and frame 2. This volume of inner products is then pooled and subsampled to form representations at coarser and coarser scales. Pooling is done so you can tell where a specific pixel in frame 1, represented in the original resolution, went to in frame 2 at various resolutions. The result is a representation of flow at multiple levels of precision. An incremental procedure then starts with zero flow, and repeatedly updates the flow using the previous flow estimate, a context feature, and a readout from the multi-scale representation.* Image credit: *Figure 1 of* RAFT: Recurrent All-Pairs Field Transforms for Optical Flow *by Z. Teed and J. Deng.*



FIGURE 36.7: *Flow fields predicted by RAFT are very close to ground truth when it is available. Strong discontinuities in flow generate no noticeable difficulties. These flow fields are obtained from two frames from the SINTEL dataset (***left***), compared with ground truth (***right***). SINTEL is a dataset of rendered frames from a CGI movie, so the true flow can be known. The direction of a flow vector is shown by hue (what changes from red to yellow to green to purple) and the magnitude by saturation (very small vectors are white, longer are pale colors, longest have intense colors).* Image credit: *Figure 1 of* RAFT: Recurrent All-Pairs Field Transforms for Optical Flow *by Z. Teed and J. Deng.*



FIGURE 36.8: *For real images, there is no ground truth to compare to. This figure shows flow fields predicted by RAFT for two datasets. KITTI data has to do with autonomous vehicles. The DAVIS dataset is at* 1088×1920 *resolution. Flow fields should look plausible to you (for example, low flow at distant objects; strong discontinuities in flow around object boundaries).* Image credit: *Figure 1 of* RAFT: Recurrent All-Pairs Field Transforms for Optical Flow *by Z. Teed and J. Deng.*

of the minimum a number of times. However, there is no explicit cost function and so no reason to believe that the updates are descent directions. Flow estimates are very strong (Figure **??**), inference is fairly fast (0.1s for a pair of images at 1088×436 resolution on a relatively outdated consumer GPU), and high resolution is possible.

36.1.8 RAFT Stereo and IGEV Stereo

RAFT builds a cost-volume, then uses a process of iterative refinement to decode into a flow field. You should expect this architecture to apply to disparity estimation. It does, with minor adjustments, and the result – RAFT-Stereo [] – performs extremely well. A minor modification is required. RAFT-Stereo applies to rectified image pairs. For a $H \times W$ image, the cost volume is $W \times H \times W$ (you need to match only along epipolar lines).

IGEV stereo improves over RAFT stereo by computing an estimate of 3D geometry , then encoding that estimate and making the encoding available to the iterative reestimation procedure (Figure 36.11). Improvements on small detail are particularly noticeable (Figure 36.12). Narrow objects, like the rails in the third row, present serious matching problems. They are small, so there isn't much information to match. Worse, the disparity has a narrow spike, and smoothed disparity estimates will tend to suppress them unless there is a good match. But the geometry representation can preserve the fact that (a) there is something there and (b) it is at a different disparity from the background, and doing so appears to help.

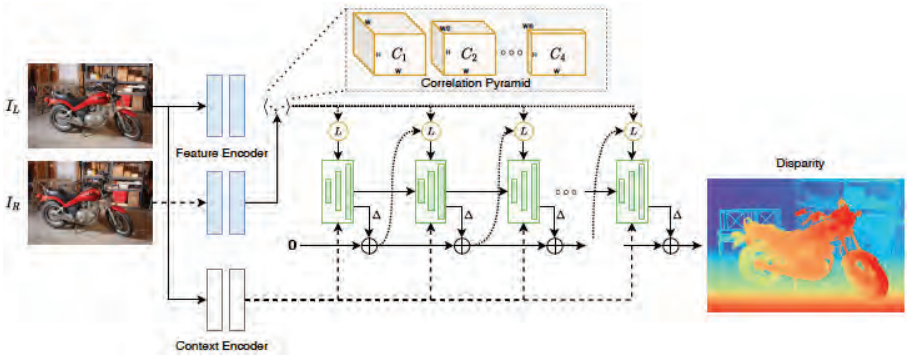


FIGURE 36.9: *RAFT-Stereo follows RAFT (compare this figure with Figure 36.6). Encode frame 1 and frame 2 to produce a feature representation, then computes a cost-volume. Pool and subsample this to form representations at coarser and coarser scales. An incremental procedure then starts with zero disparity, and repeatedly updates the disparity using the previous disparity estimate, a context feature, and a readout from the multi-scale representation.* Image credit: *Figure 1 of* RAFT-Stereo: Multilevel Recurrent Field Transforms for Stereo Matching*, by L. Lipson, Z. Teed and J. Deng.*

36.1.9 Training Generalities

You don't need disparity ground truth to train disparity estimation methods, likely because the range of disparities that must be searched is quite small. Instead, you can minimize a photometric loss between the left image and the warped right image (or the other way round). If the disparity field is wrong, the loss will be big, etc. However, optic flow methods are almost all trained on synthetic data. There are a number of different ways of constructing this data (some more details in Section 36.3). In some, there are incidental effects from aliasing, cropping or matting that mean you need to be careful.

Constructing a synthetic dataset for flow will naturally produce other, useful, information. For example, the flying chairs dataset (Figure 36.14 and below) not only has flow fields, but also has masks for pixels that are known in one image but not in the other. Similarly, the FlyingThings3D dataset has motion boundary information (Figure 36.15).

36.1.10 Boundaries and Scene Flow: Beyond Optic Flow and Disparity

You should expect that a predictor that can predict optic flow can predict disparity. It should also be able to predict motion boundaries – because they are associated with matching failures – and occlusions – again, matching failures. There is good evidence that flow prediction networks can successfully be trained to predict this information. A variant of FlowNet2 can predict flow or disparity, occlusions and motion and depth boundaries (Figure 36.16).

If a network can predict flow, disparity and boundaries, it might be able to

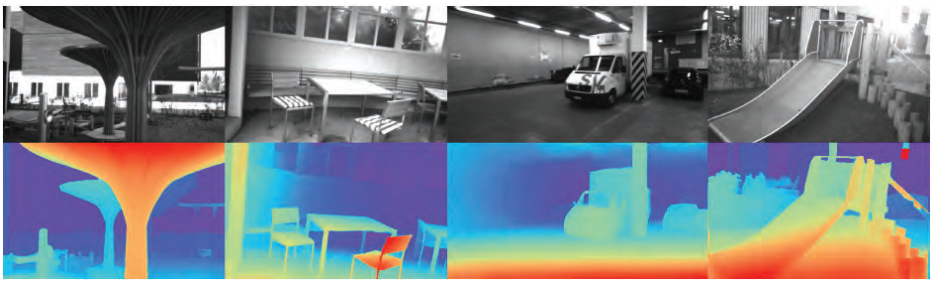


FIGURE 36.10: *RAFT-Stereo produces very strong disparity estimates.* **Top** *row shows frames from the ETH3D stereo dataset and* **bottom** *row shows recovered disparities. Notice that (a) there are precise boundaries in the right place; (b) there is very little spatial noise; and (c) there are sensible disparity estimates in regions that should be very hard to match. For example, matching must be very hard on near featureless surfaces, but the table in column 2, floor in column 3, and slide in column 5 are well reconstructed.* Image credit: *Figure 4 of* RAFT-Stereo: Multilevel Recurrent Field Transforms for Stereo Matching, *by L. Lipson, Z. Teed and J. Deng.*

predict more.

Definition: 36.4 *Scene flow*

Scene flow associates a 3D motion vector with 3D points corresponding to points in the image. Scene flow is usually dense, and often computed at every pixel.

Now assume you have a pair of images where only the camera moves and you know camera calibration. You should be able to get the scene flow, by the following argument. You could estimate the essential matrix, and use odometry to recover the camera movement. You could then rectify the images and use a stereo network to compute disparities. Now for each pixel in the image, you know a corresponding point in 3D in the first image's frame. The odometry information then yields the scene flow because nothing actually moved in the scene; the camera just moved.

In the harder case, some objects in the scene move independently and the camera moves as well. Assume you have access to a motion segmentation, so you know which objects are moving. You can treat each separate segment as a (rather odd) image where the camera moves, so you have the scene flow for each segment. The odometry should yield different movements for each segment. There are practical objections to this scheme (where did the motion segmentation come from? what happens if you can't estimate an essential matrix from a segment? and so on). But you don't need to build it. Instead, use this as an argument to suggest that something like FlowNet2 can predict scene flow from a pair of images. This argument is correct [].

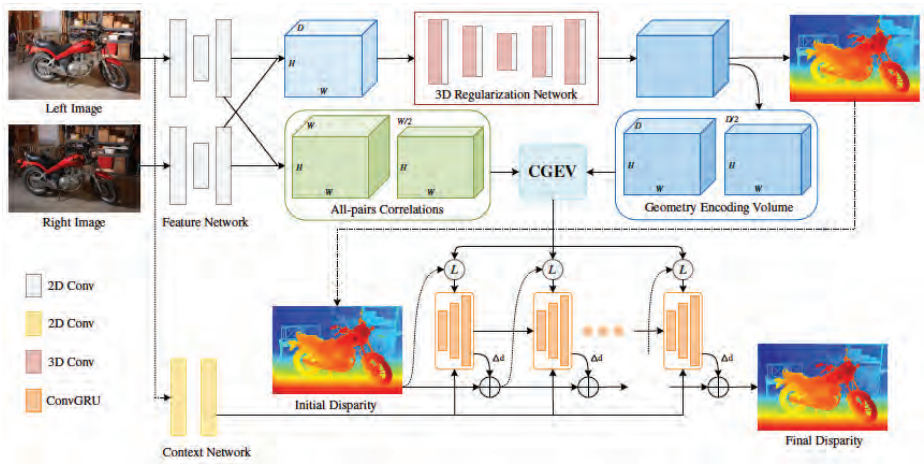


FIGURE 36.11: *The architecture of IGEV stereo (compare with Figure 36.10). IGEV improves over RAFT-Stereo by computing an estimate of 3D geometry (***top left** *– compare single image depth maps), encoding that, and using it during the iterative reestimation procedure.* Image credit: *Figure 3 of* Iterative Geometry Encoding Volume for Stereo Matching*, by G. Xu, X. Wang, X. Ding and X. Yang*

36.2 CROCOV2: FLOW AND STEREO WITHOUT CORRELATION VOLUMES

There are some nuisances associated with a cost volume. It is big, and gets bigger as the image resolution increases. It is just slightly inelegant – ideally, you would like to just feed images into an encoder, decode the results, and have flow or stereo without having to change the architecture of the encoder. There is good evidence that you can do so with transformer encoders, if you use the right unsupervised training loss.

36.2.1 CroCo

One successful loss follows from a *cross view completion* task (often referred to as *CroCo*). The discussion here focuses on the latter of two CroCo papers [] (CroCoV2!). Take two views of the same scene; tokenize both images; discard 90% of the tokens, chosen at random, for one image (the masked image); pass the remaining tokens for the masked image and the tokens for the second image (the reference image) through the same transformer encoder; and then pass all resulting tokens into a dense decoder (as in Section 22.2.1). Figure **??** illustrates the process. The decoded image should match the first view, and this loss is used to train the encoder.

Training encoders in this way requires considerable care. The positional embedding used by the decoder needs to be adjusted (below). Data needs to be quite carefully curated to have pairs where there is significant overlap, where there is a significant change of view, and where the pairs have high diversity. Details appear in [**?**]. The system in that paper used some 5.3 million real-world pairs of crops

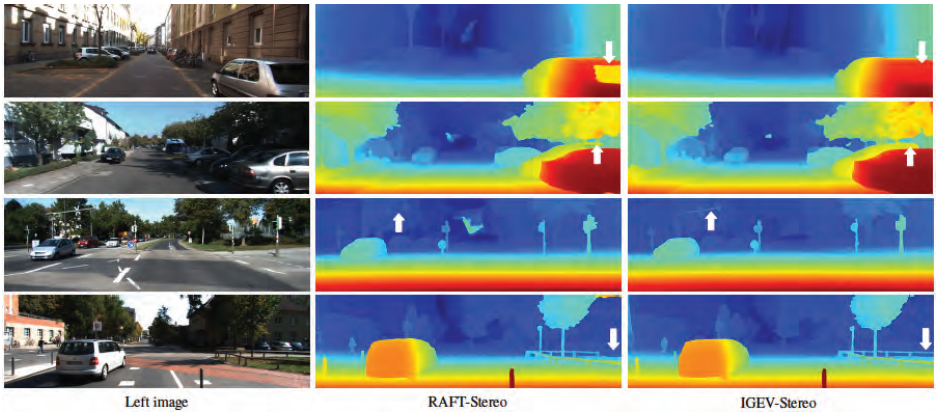


FIGURE 36.12: *IGEV disparities (***right***) compared with RAFT-Stereo disparities (***center***), showing important small improvements in detailed reconstruction. The white arrows point out details. Note the improved rear windscreen in the* **top** *row; the improvement in the traffic light gantry (***center** *row); and the improvement in the rails in the* **bottom** *row. Here* Image credit: *Figure 4 of* Iterative Geometry Encoding Volume for Stereo Matching*, by G. Xu, X. Wang, X. Ding and X. Yang*

and some 1.8 million synthetic pairs generated with a simulator.

36.2.2 ROPE - Rotary Positional Embedding

Both encoder and decoder are transformers, but the encoder has an important variation from the transformers of Chapter **??**. The positional embeddings of Section 22.1.4 encode the absolute position of a token (compare image patch) in a sequence. But the relative position of tokens is important. Imagine you build a masked autoencoder (as in Section 23.2) and apply it to say, pairs where one image is a zoomed version of another, or one is a somewhat shifted version of the other.



FIGURE 36.13: *Synthetic datasets need to be used with caution, because incidental effects may cause real problems. Here are two frames from the Flying chairs 2 dataset; notice the aliasing or cropping issues on the boundaries of the chair. Effects like these could mislead predictors, although the evidence suggests either they do not.* Image credit *Flying chairs dataset.*

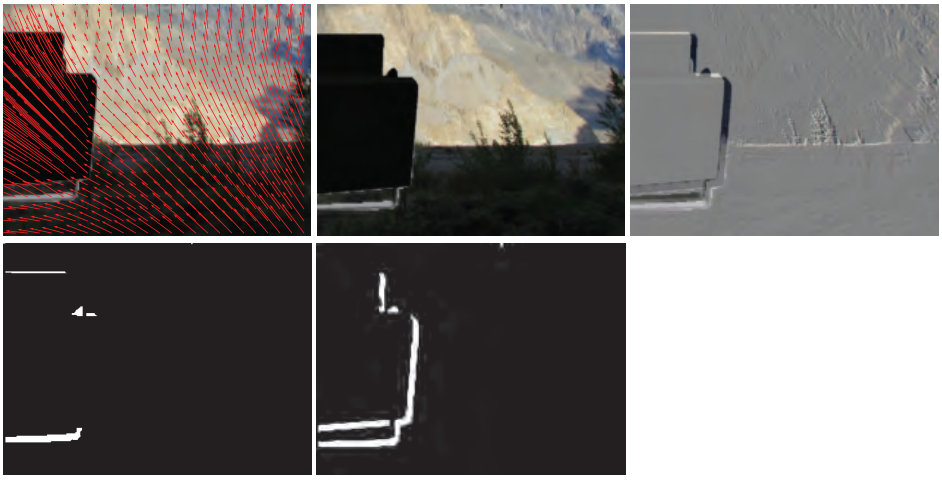


FIGURE 36.14: *A training example from the flying chairs dataset.* **Top** *shows* $\mathcal{I}_1$, $\mathcal{I}_2$ *and the flow field. The flow field is known because the dataset is synthetic – chairs are moved and placed on the background.* **Bottom** *shows masks for pixels known in one image but not in the other.* Image credit *Flying chairs dataset.*

Experiment shows it is quite difficult for the MAE to reconstruct both images in such pairs well. Reconstructing a token should depend on the spatial relations between tokens, rather than the absolute position of the tokens. The MAE doesn't "know" this, because it will not have encountered these effects in training. Changing the positional encoding from one that encodes absolute token position in the sequence to one that encodes relative token position will help. This is because shifting the image or moderate changes in scale should (very roughly) fix relative token positions, but won't fix absolute token positions.

ROPE or *rotary position embedding* is a positional embedding that represents relative, rather than absolute, position in the token stream []. Rather than adding an offset to the embedded image patch, you transform the representation inside the self-attention of Section 22.1.2. Recall that self attention uses two $d_k \times e_t$ matrices $\mathcal{M}_q$ and $\mathcal{M}_k$ (which were learned), and produces a matrix (written $\mathcal{Q}\mathcal{K}^T$ in Procedure 22.1), which then goes through a softmax operator to produce a matrix of weights. These are used to combine the tokens into a new set of tokens. For self-attention, the m, n'th entry of the matrix $\mathcal{S}$ (recall this has the form $\mathcal{Q}\mathcal{K}^T$) is given by

$$\mathbf{v}_m^T \mathbf{M}_q \mathbf{M}_k^T \mathbf{v}_n.$$

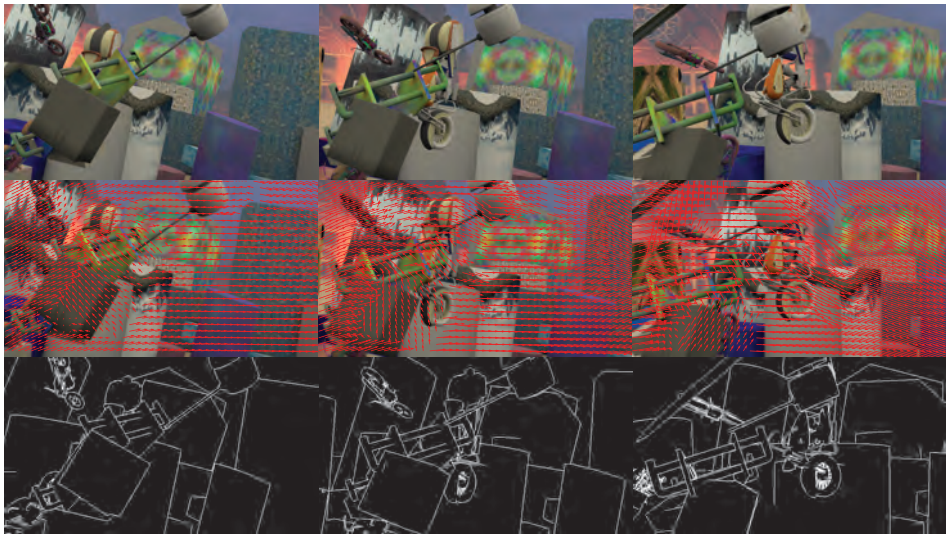


FIGURE 36.15: *Frames (***top***), flow fields (***center***) and motion boundaries (***bottom***) from the FlyingThings3D dataset.* Image credit *FlyingThings3D.*

Procedure: 36.3 *Rotary position embedding*

Assume that e_{t} is even, and write $e_{\mathrm{t}} = 2d$. Construct a matrix $\mathcal{R}_m$ for the m'th token, using

$$\theta_u = 1000^{-2(u-1)/(2d)} \text{ and } c_{m,u} = \cos m\theta_u \text{ and } s_{m,u} = \sin m\theta_u$$

where $i \in \{1, \ldots d\}$. Here

$$\mathcal{R}_m = \begin{bmatrix} c_{m,1} & -s_{m,1} & 0 & \ldots & 0 & 0 & 0 \\ s_{m,1} & c_{m,1} & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & c_{m,2} & -s_{m,2} & 0 & \ldots & 0 \\ 0 & 0 & s_{m,2} & c_{m,2} & 0 & \ldots & 0 \\ & & & \ldots & & & \\ & & & \ldots & & & \\ 0 & 0 & 0 & 0 & \ldots & c_{m,d} & -s_{m,d} \\ 0 & 0 & 0 & 0 & \ldots & s_{m,d} & c_{m,d} \end{bmatrix}$$

Compute the m, n'th entry of $\mathcal{S}$ using

$$\mathbf{v}_m^T \mathbf{M}_q \mathcal{R}_m \mathcal{R}_n^T \mathbf{M}_k^T \mathbf{v}_n.$$

Notice that the form of ROPE forces the positional component of the self-attention between tokens to depend on $m - n$ rather than m or n directly **exer-**



FIGURE 36.16: *A variant of FlowNet2 can predict occlusions as well as flow or disparity. For each of these pairs of frames from SINTEL, the figure shows the ground truth occlusions (known because SINTEL is rendered from 3D models;* **above***) and the predicted occlusions (***below***). Note how fine occlusions are predicted very accurately.* Image credit *Part of Figure 2 of "Occlusions, Motion and Depth Boundaries with a Generic Network for Disparity, Optical Flow or Scene Flow Estimation", by E. Ilg, T. Saikia, M. Keuper, and T. Brox*

cises.

36.2.3 Predicting Flow or Stereo

Now assume you have a transformer encoder pretrained with CroCoV2. Obtain a very large number of stereo pairs with known disparity. Pass both images of a stereo pair through the pretrained encoder. Use a dense decoder, as in Section 22.2.1, but adjusted to see tokens from both images (Figure **??**). You can then train the decoder to produce disparity or flow.

Some details require care. Each disparity map is with respect to one of the two images in a pair. Similarly, each flow map is with respect to one of the two images in a pair. This means you can't swap images in a pair without making changes to the disparity or flow. But notice the two input images are handled slightly differently by the process (Figure **??**; also, the discussion of masked and reference images in Section 36.2.1). You must choose a convention where the image defining the disparity (or flow) is always one of mask or reference image.

At some points in an image, disparity or flow isn't really known because there is no corresponding point in the other image (Sections 36.3 and 36.3). This suggests you should use an uncertainty measure in the loss. For disparity, use a decoder that predicts a pixel value $\mathbf{p}_{ij}$ and a scale d_{ij} at the i, j'th location. Write $\mathbf{t}_{ij}$ for the true value at that location. The loss is then

$$\sum_{ij} \left[\frac{\| \mathbf{p}_{ij}, \mathbf{t}_{ij} \|_1}{d_{ij}} + 2 \log d_{ij} \right].$$

This loss has a probabilistic interpretation **exercises**. Alternatively, you can think of d_{ij} as an uncertainty measure. The term $2 \log d_{ij}$ will become more negative as d_{ij} gets smaller, and the term $\frac{\|\mathbf{p}_{ij}, \mathbf{t}_{ij}\|_1}{d_{ij}}$ will get larger. In turn, for a prediction the network is confident will be very accurate, the d_{ij} can be small. If the prediction is likely to be less accurate, the d_{ij} will be larger **exercises**. The scale needs to

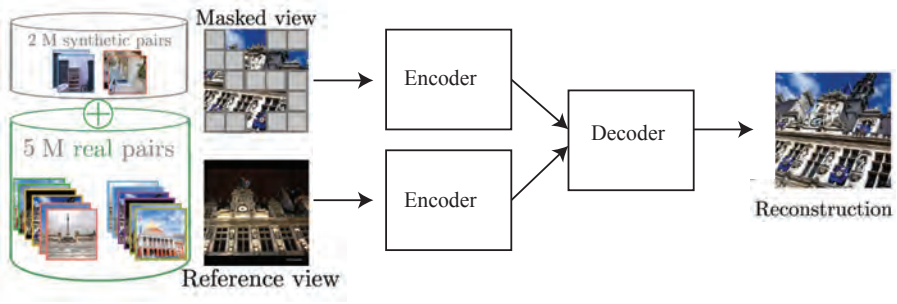


FIGURE 36.17: *Pretraining a CroCo encoder uses pairs of views of a scene. One view, the masked image, has 90% of its tokens dropped at random. The other, the reference image, is complete. Tokens are passed to two copies of an encoder, which is used to produce tokens that can be used by a dense decoder to reconstruct the missing parts of the masked input image. The procedure forces the encoder to produce token representations that contain information useful for two-view problems. Example images appear in Figure* **??**. Image credit *Uses parts of Figure 2 of "CroCo v2: Improved Cross-view Completion Pre-training for Stereo Matching and Optical Flow", by Philippe Weinzaepfel, Thomas Lucas, Vincent Leroy, Yohann Cabon, Vaibhav Arora, Romain Brégier, Gabriela Csurka, Leonid Antsfeld, Boris Chidlovskii, Jérôme Revaud, ICCV, 2023*

be positive, and you can achieve this by (for example) predicting d'_{ij} then setting $d_{ij} = e^{d'_{ij}}$. The original paper uses a fairly complex expression for the scale ([**?**], sec. 4).

36.3 DATASETS AND RESOURCES

As of writing, a very good place to start for stereo datasets is the Middlebury stereo page, `https://vision.middlebury.edu`, which offers datasets, evaluation procedures and leaderboards for a number of algorithms. There are links to other resources there, too. I have reproduced some links you will find there below. Further resources, mostly aimed at regression methods, appear in Section 36.3.

Resource: Stereo dataset collection at Middlebury A collection of datasets appears at `https://vision.middlebury.edu/stereo/data/`. These datasets were first described in [**?**, **?**, **?**, **?**, **?**].

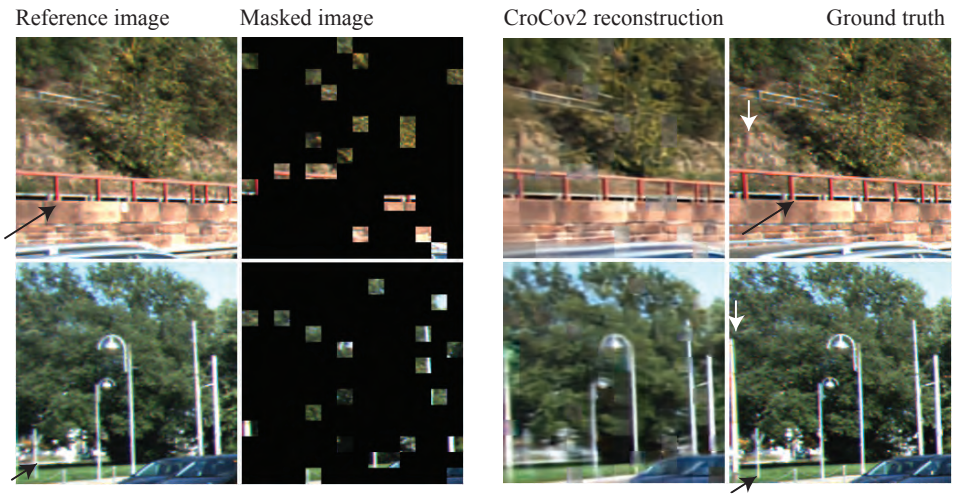


FIGURE 36.18: *CroCoV2 uses tokens from a reference image (***left***) and a small subset of tokens produced by a masked image (***center left***) to reconstruct the masked image. The reconstruction produced by a trained encoder-decoder pair is shown on the* **center right** *and the true image on the* **right**. *The reference and masked images are slightly different views of the same scene (check this: dark arrows show some features that have obvious disparity, and light arrows show some features that appear in only one view). The reconstruction loss forces the encoder to represent cross-view relations.* Image credit *Uses parts of Figure 9 of "CroCo v2: Improved Cross-view Completion Pre-training for Stereo Matching and Optical Flow", by Philippe Weinzaepfel, Thomas Lucas, Vincent Leroy, Yohann Cabon, Vaibhav Arora, Romain Brégier, Gabriela Csurka, Leonid Antsfeld, Boris Chidlovskii, Jérôme Revaud, ICCV, 2023*

Resource: Kitti stereo datasets There are two datasets at `https://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo`. The first – the KITTI 2012 dataset – consists of 194 training image pairs and 195 test image pairs. This dataset was first described in [**?**]. The second – the KITTI 2015 dataset – consists of 200 training scenes and 200 test scenes. This dataset was described in [**?**, **?**].

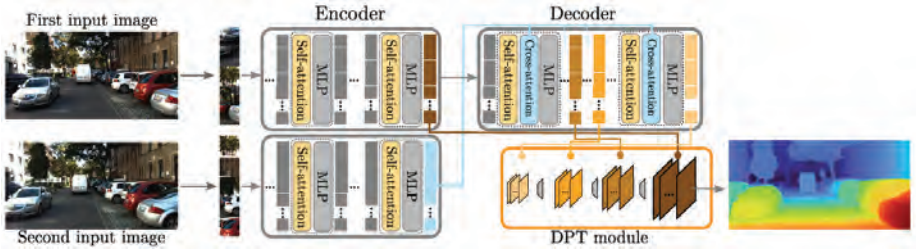


FIGURE 36.19: *Once a transformer encoder is pretrained with CroCoV2, you can train a decoder to predict either disparity or flow. Tokens from the first image are passed through the encoder, and then to a transformer decoder. Tokens from the second image are passed through the encoder, and cross-attend with the first image's tokens. The state of the transformer decoder is exposed to a dense decoder (as in Section 22.2.1) and decoded into either disparity or flow.* Image credit *Uses Figure 5 of "CroCo v2: Improved Cross-view Completion Pre-training for Stereo Matching and Optical Flow", by Philippe Weinzaepfel, Thomas Lucas, Vincent Leroy, Yohann Cabon, Vaibhav Arora, Romain Brégier, Gabriela Csurka, Leonid Antsfeld, Boris Chidlovskii, Jérôme Revaud, ICCV, 2023*

Resource: Stereo compilation There is a curated list of state-of-the-art deep stereo matching resources maintained by Fabio Tosi, Matteo Poggi and Luca Bartolomei, from the University of Bologna. This extensive compilation of resources, including paper references, pointers to github pages, and pointers to datasets can be found at `https://github.com/fabiotosi92/Awesome-Deep-Stereo-Matching`. This compilation includes methods well out of the scope of this chapter.

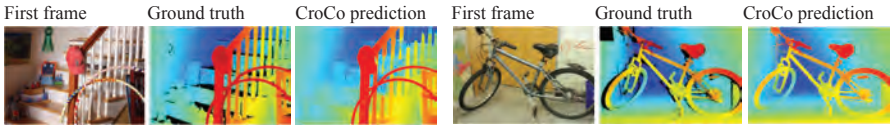


FIGURE 36.20: *Stereo predictions by CroCoV2. I do not know whether the false color represents disparity or depth, and it is hard to tell from the images alone. However, the encoding is the same for prediction and ground truth, so you can compare. Notice some artifacts in the stair rails, cause by a tiling procedure described in the original paper [?].* Image credit *Uses parts of Figure 6 of "CroCo v2: Improved Cross-view Completion Pre-training for Stereo Matching and Optical Flow", by Philippe Weinzaepfel, Thomas Lucas, Vincent Leroy, Yohann Cabon, Vaibhav Arora, Romain Brégier, Gabriela Csurka, Leonid Antsfeld, Boris Chidlovskii, Jérôme Revaud, ICCV, 2023*

Resource: Stereo leaderboards There are a number of stereo leaderboards:

- At `https://vision.middlebury.edu/stereo/eval3/`, find a leaderboard maintained by Daniel Scharstein, Richard Szeliski and Heiko Hirschmüller. This evaluates a wide range of methods on a standard dataset, using a range of metrics. Very recent methods are present.
- There is a leaderboard for methods using the Kitti 2015 dataset at `https://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo`, maintained by A. Geiger, P. Lenz, C. Stiller and R. Urtasun. This evaluates a wide range of methods on a standard dataset, using a range of metrics. Very recent methods are present.

Resource: SceneView

Resource: ETH3D



FIGURE 36.21: *Flow predictions by CroCoV2. Here the color components yield the flow arrow. The encoding is the same for prediction and ground truth, so you can compare. Notice some artifacts in the fingers, likely a result of smoothing flow on small, fast-moving objects.* Image credit *Uses parts of Figure 7 of "CroCo v2: Improved Cross-view Completion Pre-training for Stereo Matching and Optical Flow", by Philippe Weinzaepfel, Thomas Lucas, Vincent Leroy, Yohann Cabon, Vaibhav Arora, Romain Brégier, Gabriela Csurka, Leonid Antsfeld, Boris Chidlovskii, Jérôme Revaud, ICCV, 2023*

Resource: CroCo At `https://github.com/naver/croco`, find code and weights for CroCo and CroCoV2, together with pretraining instructions.

Insert dataset generators - what are these?

It is not easy to find natural image pairs where the flow is known. One important strategy is to build a synthetic image dataset for which the flow is known. Important synthetic flow datasets are listed below.

Resource: Sintel Sintel is a dataset of frames rendered from an open-source animated film. The film is "Sintel" which is an open source animated short film produced by Ton Roosendaal and the Blender Foundation. The dataset consists of long sequences with known flow, and can be found at `http://sintel.is.tue.mpg.de`.

Resource: Flying chairs and Flying chairs 2 Flying chairs is a dataset consisting of 22872 image pairs with known flow fields. The frames show renderings of 3D chair models moving in front of images downloaded from the internet. The motion of chairs is planar, as is the motion of backgrounds. Flying chairs 2 contains more information about the frames, including information about occlusions and motion boundaries. These datasets can be found at `https://lmb.informatik.uni-freiburg.de/resources/datasets/FlyingChairs.en.html`.

Resource: Flyingthings 3D Flyingthings3D contains more than 39000 stereo frames rendered from synthetic sequences. There are associated annotations of a variety of kinds, including optical flow maps and full intrinsic and extrinsic camera data. The dataset can be found at `https://lmb.informatik.uni-freiburg.de/resources/datasets/SceneFlowDatasets.en.html#downloads`. This dataset appears not to have the occasional problems encountered with the Flying chairs datasets (Figure **??**).

Resource: ChairsSDHom and ChairsSDHom 2 ChairsSDHom was built to produce flow magnitude histograms similar to those estimated from a real world motion dataset (UCF 101, a dataset of human actions, `ucf101`). Displacements tend to be small, and there are many untextured regions. ChairsSDHom 2 contains more information about the frames, including information about occlusions and motion boundaries. These datasets can be found at `https://lmb.informatik.uni-freiburg.de/resources/datasets/FlyingChairs.en.html` (this isn't a typo; scroll down).

36.4 YOU SHOULD

36.4.1 remember these facts:

36.4.2 remember these procedures:

Building a cost volume 650
Building a reduced cost volume 651
Rotary position embedding 665

36.4.3 exploit these resources:

Stereo datasets at Middlebury 667
Kitti stereo datasets 668
Stereo compilations 669
Stereo leaderboards 670
SceneView 670
ETH3D 670
CroCo 671
Optic flow datasets - Sintel 671
Optic flow datasets - Flying chairs 1 and 2 672
Optic flow datasets - Flyingthings 3D 672
Optic flow datasets - ChairsSDHom 672

36.4.4 be able to:

EXERCISES

QUICK CHECKS

36.1. Section 35.2.1 requires a least-squares fit to an image intensity in a window; how would you do this?

36.2. Section 35.2.1 has: "Ignore the effect of flow on the constant. It is inconvenient to work with. It is also not a particularly reliable cue." Explain.

36.3. Check that the estimate of Procedure 35.2 involves solving a small linear system at each pixel location.

36.4. Check that the estimate of Procedure 35.3 also involves solving a small linear system at each pixel location.

36.5. Check that the estimate of Procedure 35.4 couples the linear system at each pixel location so that you cannot solve them independently (at least, not without something like an iterative method).

36.6. Section 35.3.1 has: "If the parametric flow model is linear in the parameters, you will need to solve a linear system." Check this.

36.7. Section 35.3.2 has: "One significant advantage of this approach is that the estimated flow field can be discontinuous in space." Explain.

LONGER PROBLEMS

36.8. Section 35.2.1 asserts that if $\mathcal{A}_1(\mathbf{x}_1)$, $\mathbf{b}_1(\mathbf{x}_1)$ and $c(\mathbf{x}_1)$ are an accurate local representation of $\mathcal{I}_1(\mathbf{x}_1)$, if $\mathcal{I}_2(\mathbf{x}) = \mathcal{I}_1(\mathbf{x} - \mathbf{t})$, and if $\mathbf{t}$ is sufficiently small, then

$$\mathcal{A}' = \mathcal{A}_1,\ \mathbf{b}' = \mathbf{b}_1 - 2\mathcal{A}_1\mathbf{t}, \text{ and } c' = c - \mathbf{b}_1^T\mathbf{t} + \mathbf{t}^T\mathcal{A}_1\mathbf{t}$$

are an accurate representation of $\mathcal{I}_2(\mathbf{x}_1)$. Show that this is true.

36.9. Figure 35.5 has: "Layered motion can be applied to longer sequences than just pairs of images".

(a) Explain how you would estimate flow in the case of a sequence of more than two frames.

(b) How can you use this to recover the intensity of pixels that happen to be occluded in some (but not all) frames?

PROGRAMMING EXERCISES