Deep Scene Understanding from Images

Matteo Poggi, Fabio Tosi, Pierluigi Zama Ramirez
Computer Vision Lab (CVLab), University of Bologna

6- Multiple Views and Motion

Summary of contents:

- **Multi-View Stereo depth estimation:**
  basic concepts of Multi-View Stereo (MVS), plane-sweep algorithm, modern approaches (CNNs)

- **Estimating pixels motion (optical flow):**
  introduction to optical flow, some classical algorithms, modern approaches (CNNs), supervised vs unsupervised optical flow networks

These slides contains a mini-survey on the two topics.
We will focus on the most important methods over which all the others are built upon, with these latter being reported for completeness

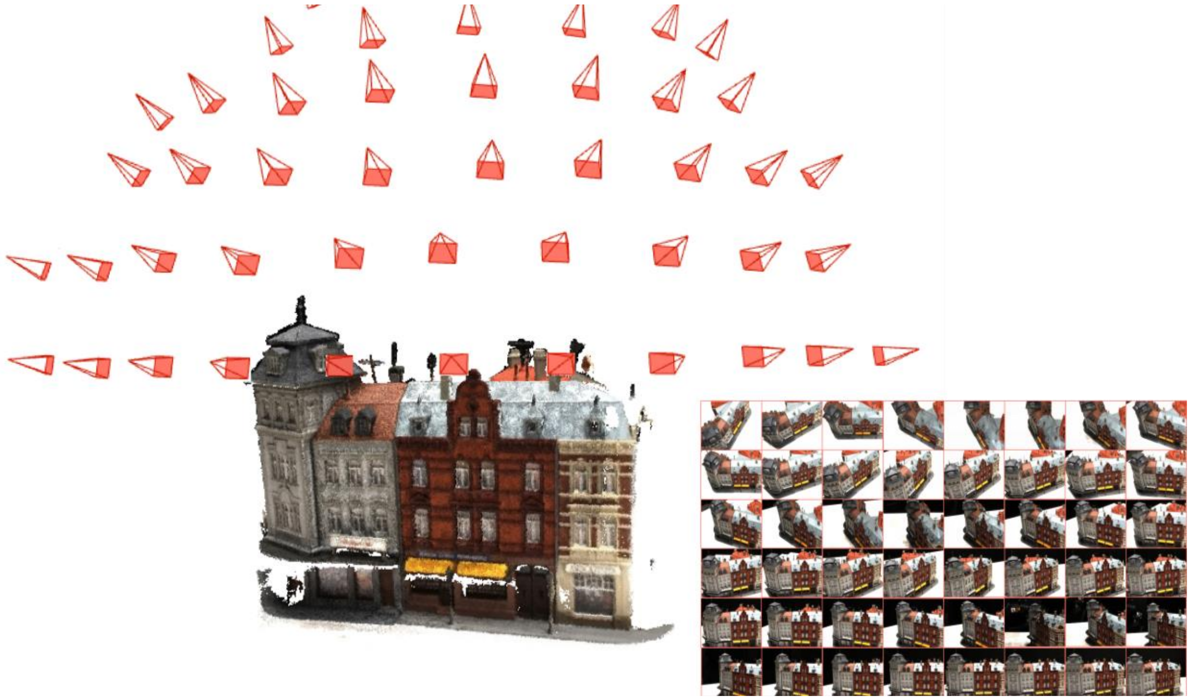
Student: *"What are the three most important problems in computer vision?“*

Takeo Kanade: *“Correspondence, correspondence, correspondence!”*

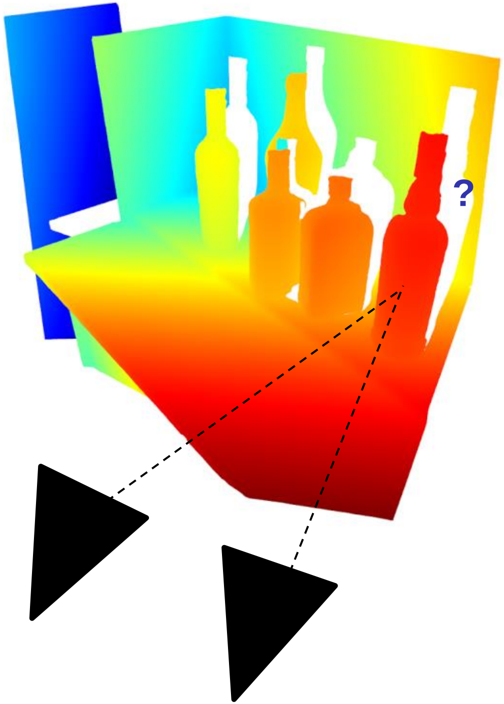
6.1- Multi-View Stereo

Multi-View depth estimation

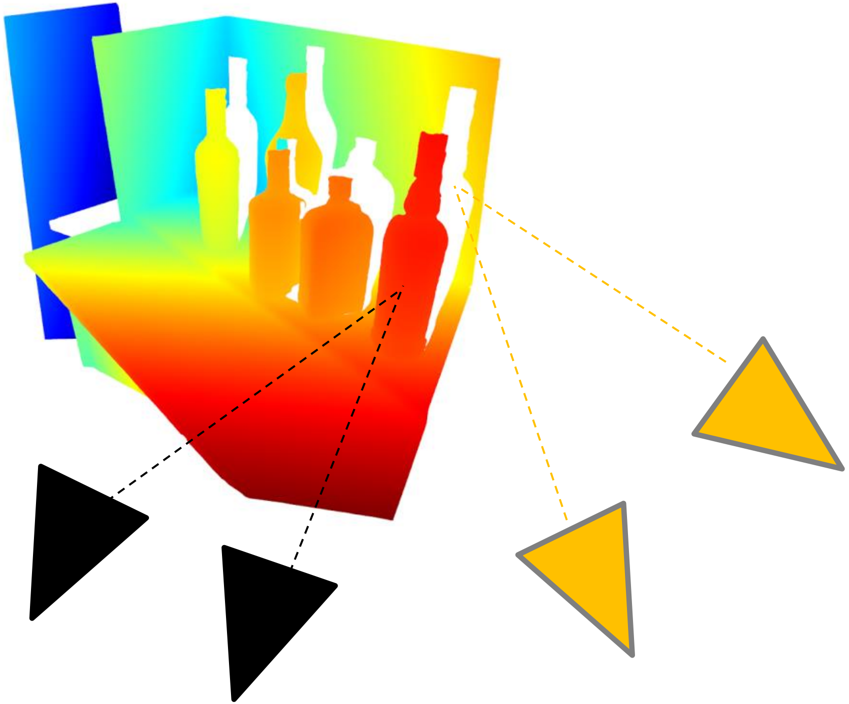
Collecting N images around the scene (N >> 2), we can aim at full **3D reconstruction.**
This would not be possible with binocular stereo matching (no visibility behind objects, etc)



Multi-View depth estimation



Multi-View depth estimation



Multi-View depth estimation

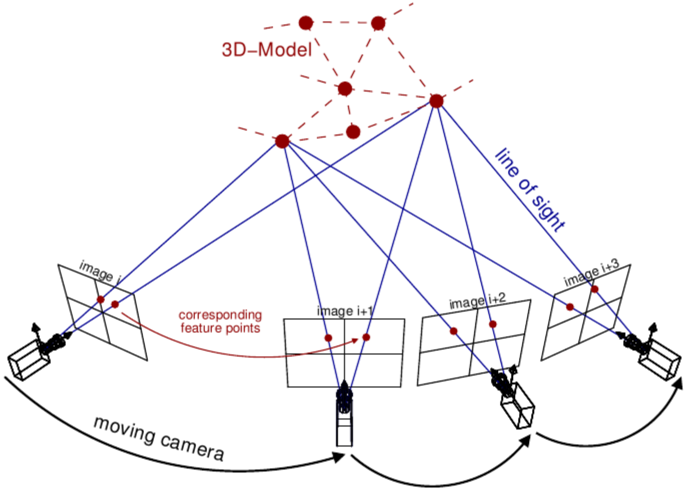
Multi-View Geometry (MVG) is a wide research area in computer vision.
Given N images, we deal with different tasks according to what we already know:

- **Structure-from-Motion** (SfM)
We do not know either the camera position or
the 3D structure of the scene. We aim at estimating both

- **Multi-View Stereo** (MVS)
We know camera positions. We aimg at
estimating the 3D structure of the scene

- **Simultaneous Localization And Mapping** (SLAM)
  We do not know either the camera position or
  the 3D structure of the scene. We aim at estimating both
  in real-time

- …

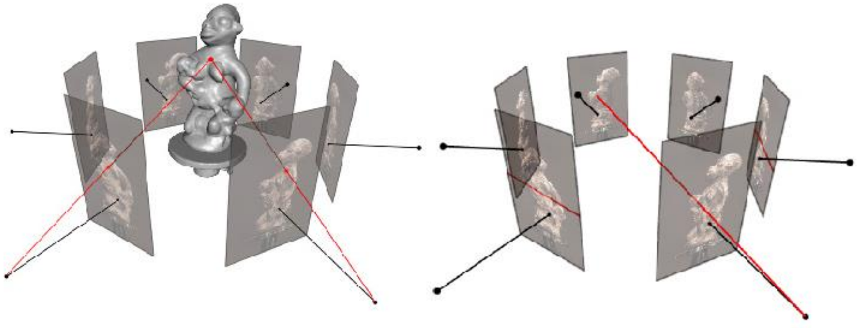


Multi-View depth estimation

We focus on **Multi-View Stereo** which is, on its own, a vast topic as well

In general, when dealing with MVS 3D reconstruction, three main categories of approaches exist:

- Direct point cloud reconstructions (3D points)
- Volumetric reconstructions (voxels)
- Depth map reconstructions



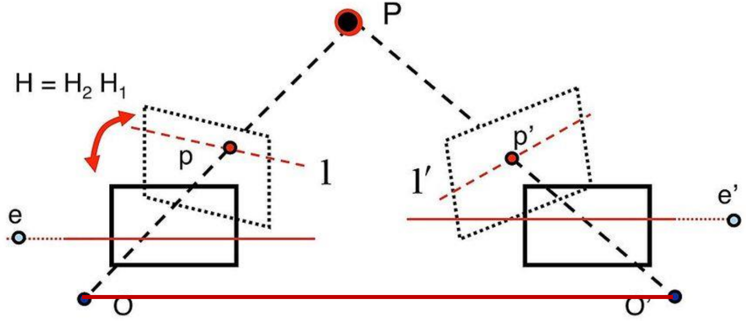
The latter consists into estimating N depth maps, one for each image in the set being assumed as **reference image**, and then fusing them for the final 3D reconstruction.

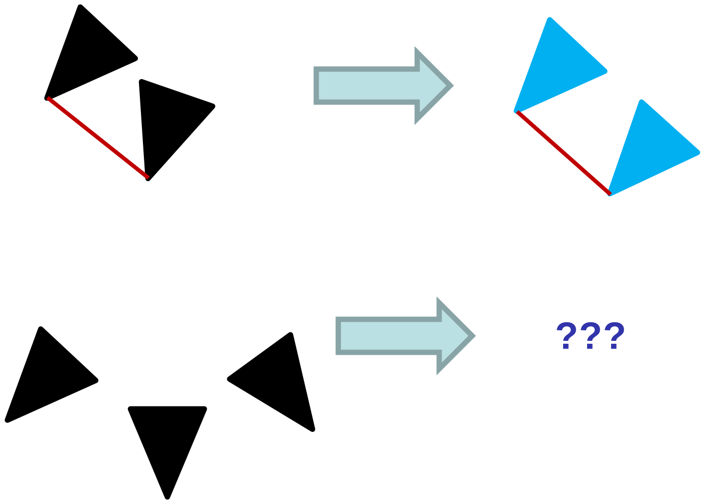
This results as the most scalable approach in terms of computational efforts – and closests to the others we have seen previously ☺

Multi-View depth estimation

Multi-View stereo depth estimation leverages **epipolar geometry** as well

However, in this case we cannot rectify images to obtain **horizontal epipolar** lines



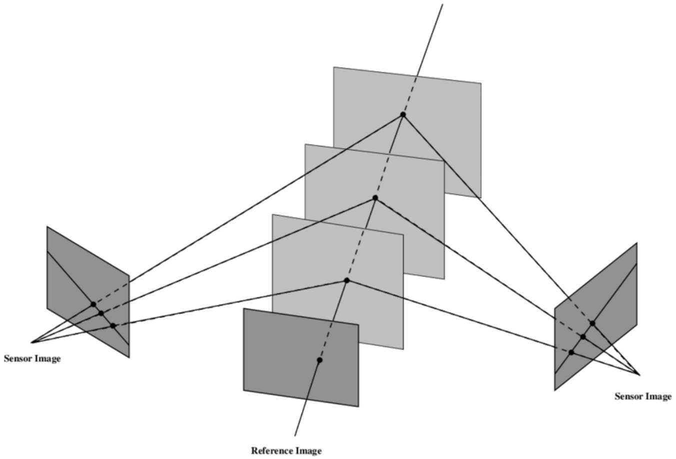


Multi-View depth estimation

Why using **multiple** views (N>2)? To reduce **ambiguity** and **occlusions** (when possible)

Multi-View depth estimation

Plane-sweep algorithm: by assuming that any 3D point in the scene lays on a plane distant d from the camera, we look for **correspondences** between pixels in the reference image and those along the **epipolar lines** in the target images



Stereo Matching algorithms are a particular case of the plane-sweep approach, for which epipolar lines are **horizontal**

A basic Multi-View Stereo algorithm can be implemented applying plane-sweep principles within the Semi-Global Matching pipeline (see **OpenMVS**)

Multi-View depth estimation

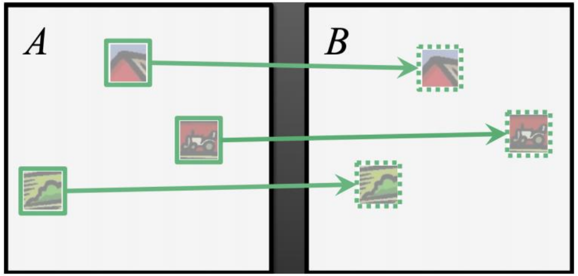
PatchMatch: randomized algorithm for matching patches across two images

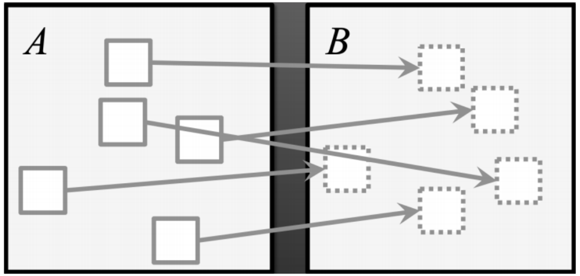
According go the **law of large numbers**, a non-trivial subset of all the possible random assignments will be correct

Three steps:

- **Initialization** – assign random patches (**offsets**)
- **Propagation** – using spatial coherence (nearby patches in one image should match with nearby patches on the second one)
- **Random search** – search for a random offset near the best patch

Repeat steps 2, 3 until **convergence**





How can this work? Given M patches, chance of selecting correct patch 1/M. Chance of selecting atleast one correct patch $p=1-(1-1/M)^M$ (for 100K patches, **p=~0.74**). If we relax this to top C nearest neighbors, we get 1-(1-C/M)M (for C=2, **p=~0.86**. For C=3, **p=~0.95**)

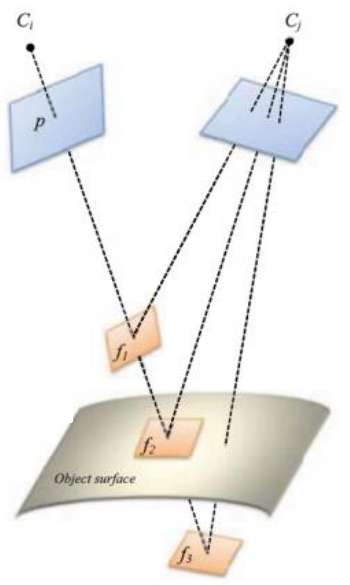
Multi-View depth estimation

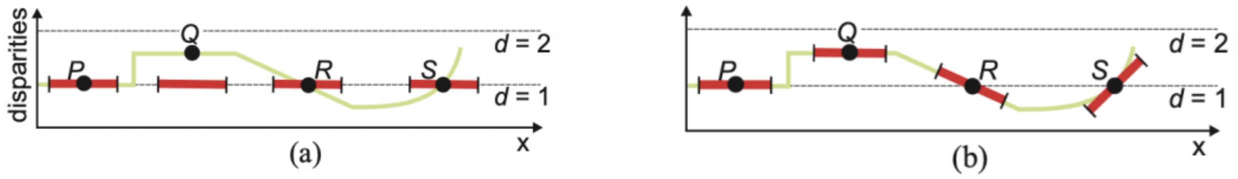
PatchMatch Stereo: based on patchmatch.
Offset replaced by **depth** and **normals**

Assumption: the world is msotly made of **almost planar surfaces**

Three steps:

- **Initialization** – assign random depths and normals
- **Propagation** – using spatial coherence
- **Random search** – sample new random **depths** and **normals**, refine initial estimate

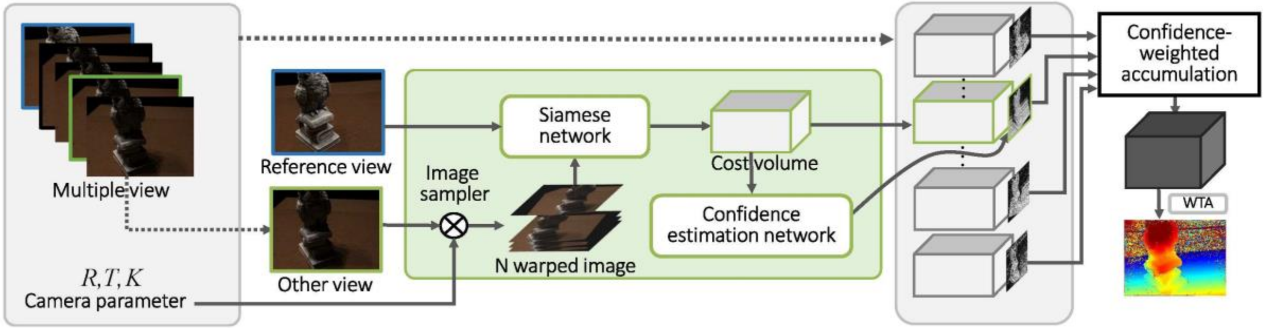




Multi-View depth estimation

Given the analogies with binocular stereo, we may expect a similar trend in the literature :)

Early attempts aimed at learning how to match patches [1] across N views



Picture from [1]

For each image patch in the reference image, a number of patches are sampled from the N-1 remaining views along the **epipolar line**.

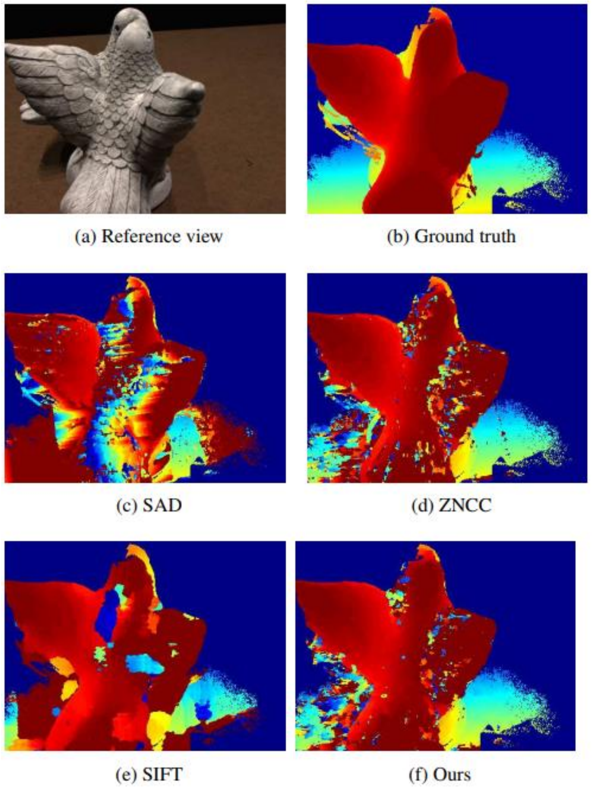
Two-view volumes are built from the reference image and any single remaining view. The N-1 volumes are accumulated by means of a weighted sum. The weights is given by the **confidence** estimated by a specific submodule.

Multi-View depth estimation

The learned matching function results
more robust than hand-crafted alternatives

However, some outliers still remains, due to the high ambiguities which cannot be explained withing a local patch

To solve this, larger image content needs to be taken into account. Solution: **end-to-end networks**!

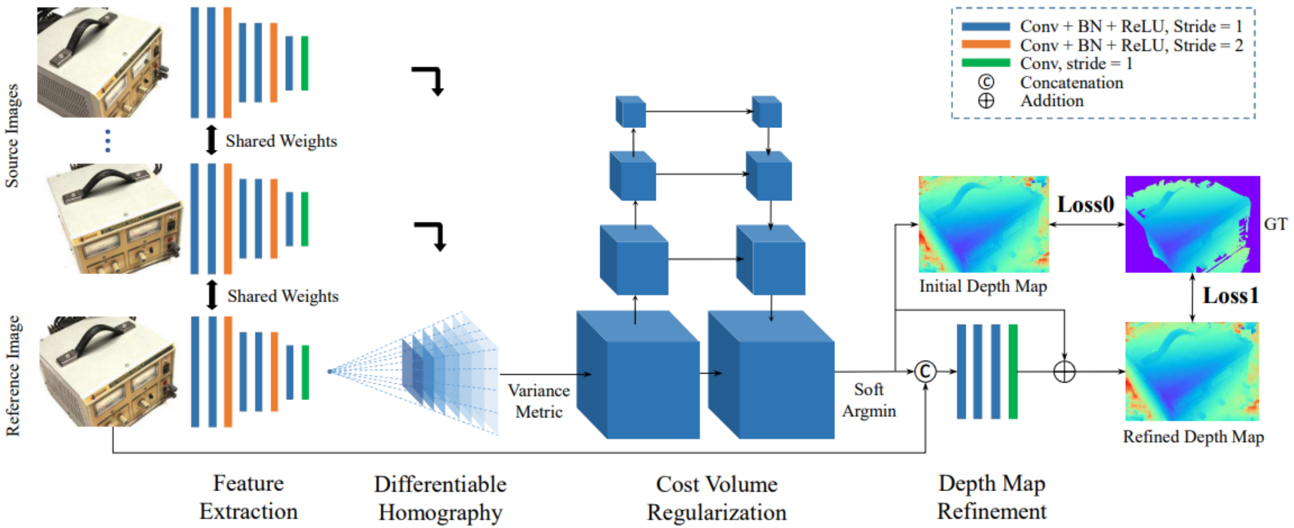


Picture from [1]

MVSNet [2]

First end-to-end multi-view depth estimation network. Very similar to **GCNet**. Four main modules:

1) Feature Extractor
2) Homography-based Cost Volume
3) Cost Volume Regularization
4) Depth Map Refinement

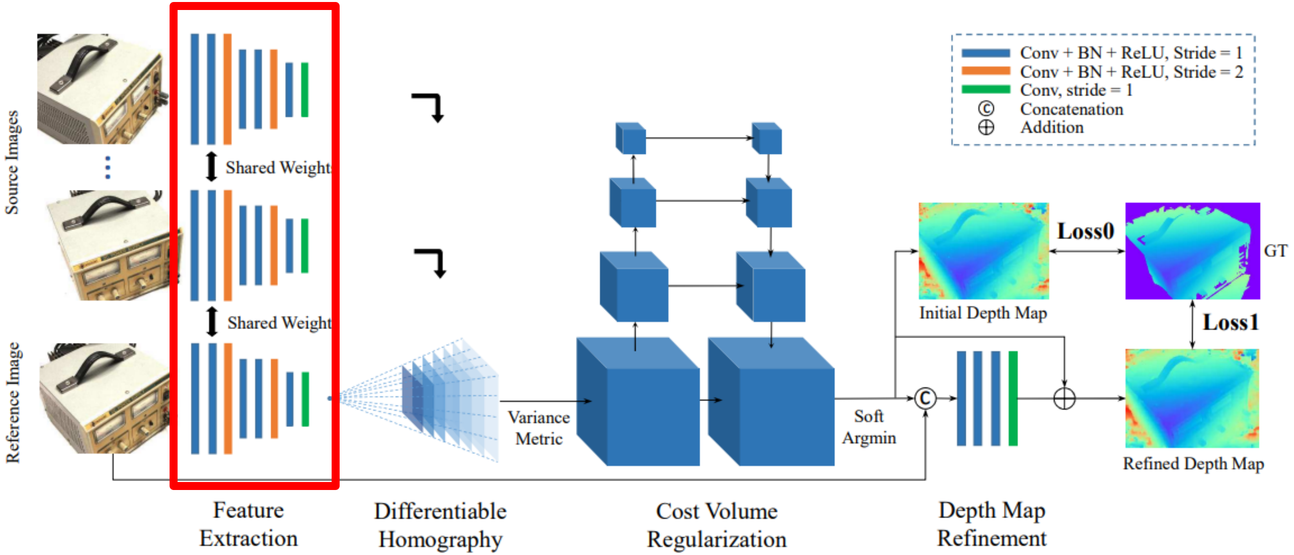


Picture from [2]

MVSNet [2]

Feature extractor

A 2D convNet extracting deep features at lower resolution (quarter), which will be used to measure pixels similarity. N instances are built to process N images (sharing the weights).

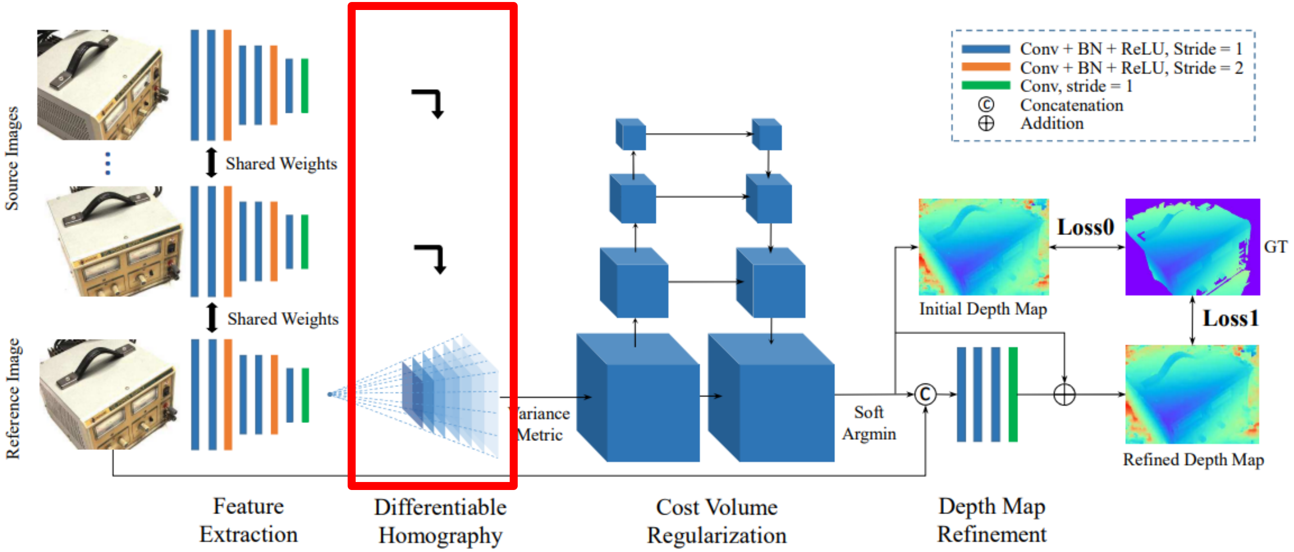


Picture from [2]

MVSNet [2]

Homography-based Cost Volume

A cost volume is built to measure pixels similarity. A single pixel in the reference image is compared to pixels in N-1 target views. Cost function: **features variance**



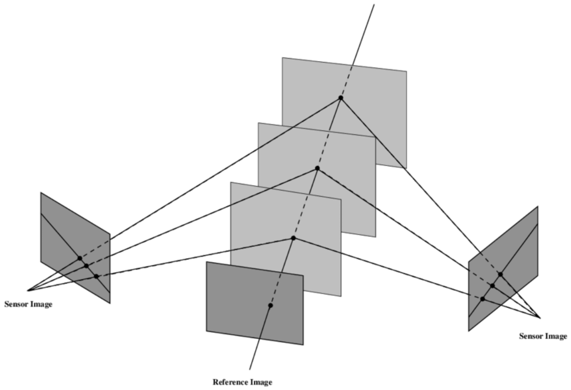
Picture from [2]

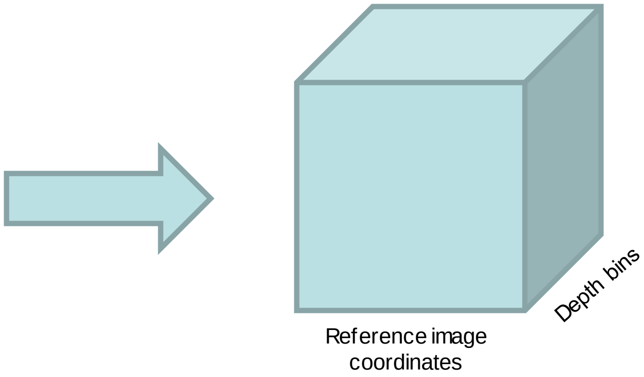
Multi-View depth estimation

By dividing the depth range into D **bins**, for any pixel in the reference image we recovered its corresponding pixels in the N-1 target views.

This can be done by applying D **homographies** to warp the target views (i.e., assuming pixels to lay on a set of D planes, defined by the depth bins themselves). This is equivalent to searching for corresponding pixels along **epipolar lines** in the target views. To measure the similarity between the pixel in the reference image and its D tuples of N-1 candidates, the **variance** on the N pixels is performed at any bin.

The bin with the lowest variance corresponds to the depth hypothesis being most likely to be correct.

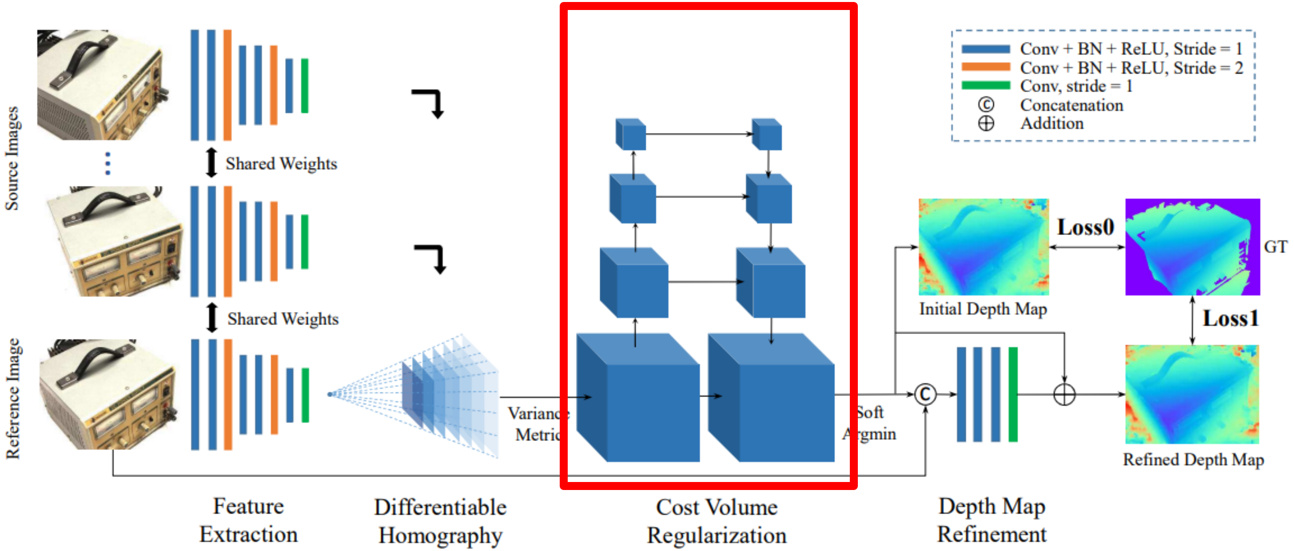




MVSNet [2]

Cost Volume Regularization

A 3D convNet used to refine the cost volume, following UNet design. It requires **high** memory consumption and runtime. From the final output, an initial depth map is obtained through **soft argmin** (at quarter the input resolution)

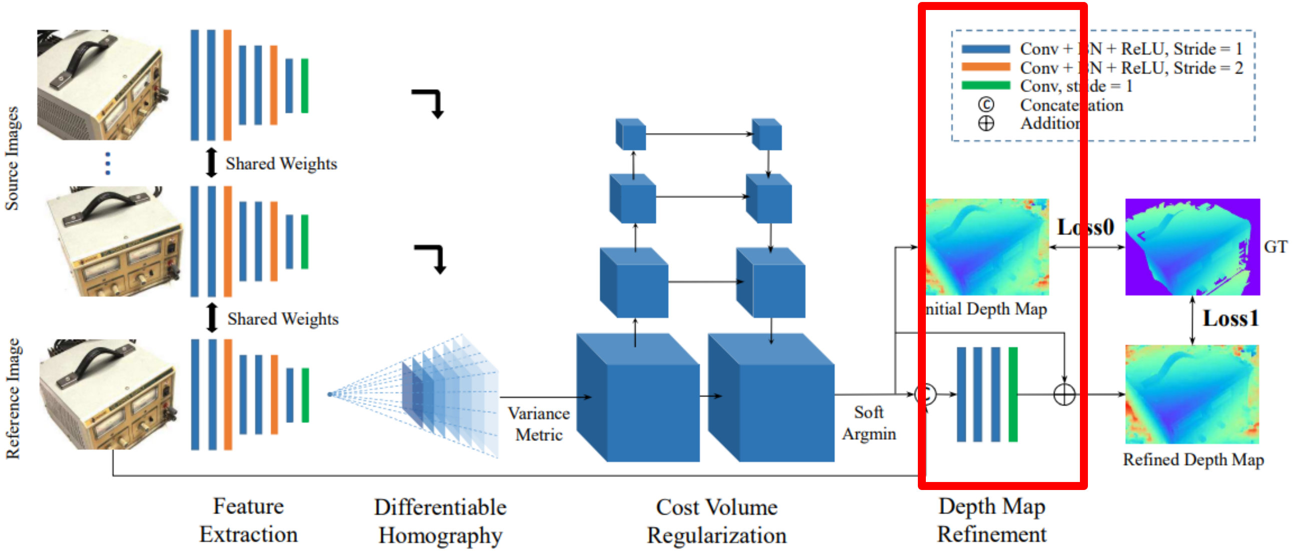


Picture from [2]

MVSNet [1]

Depth Map Refinement

A 2D convNet used to refine the initial depth map, by predicting a **residual** to be summed to the initial prediction.



Picture from [2]

MVSNet [2]

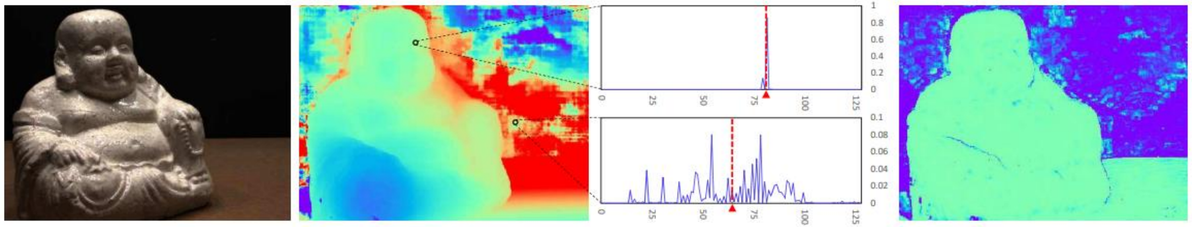
Loss function

The network is trained by minimizing both the difference between the initial and refined depth maps with respect to the ground-truth

$$Loss = \sum_{p \in \mathbf{p}_{valid}} \underbrace{\|d(p) - \hat{d}_i(p)\|_1}_{Loss0} + \lambda \cdot \underbrace{\|d(p) - \hat{d}_r(p)\|_1}_{Loss1}$$

Probability map (aka confidence)

From the 3D convNet output, a confidence map can be obtained by computing the **entropy** over the probability distribution used to obtain the initial depth map



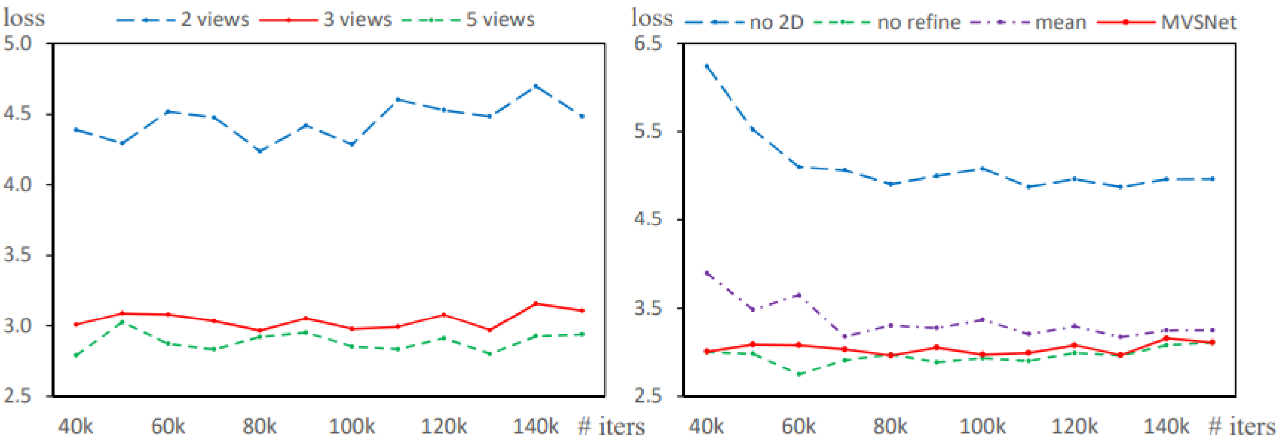
(a) Reference image (b) Inferred depth map (c) Probability distribution (d) Probability Map

Picture from [2]

Multi-View depth estimation

Ablation experiments

- Left: Varying the number of input images has impact on the performance
- Right: The variance-based cost volume results better than possible alternatives such as patches mean. The refinement network has limited impact



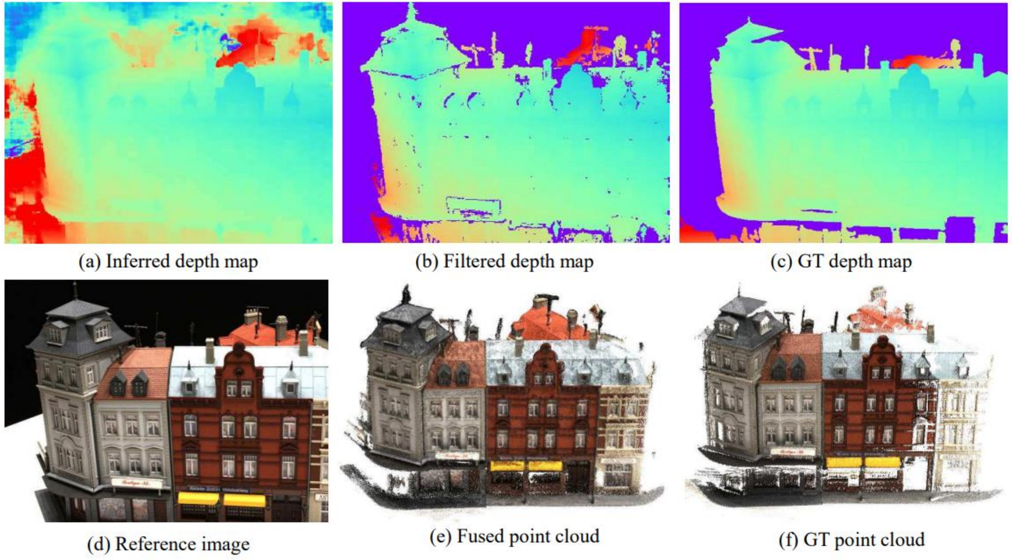
(a) View numbers (b) Components

Picture from [2]

Multi-View depth estimation

Point cloud fusion

Once a depth map has been on any image assumed as reference, they can be fused to obtain a 3D point cloud by reasoning on **visibility** and **occlusions** [3]

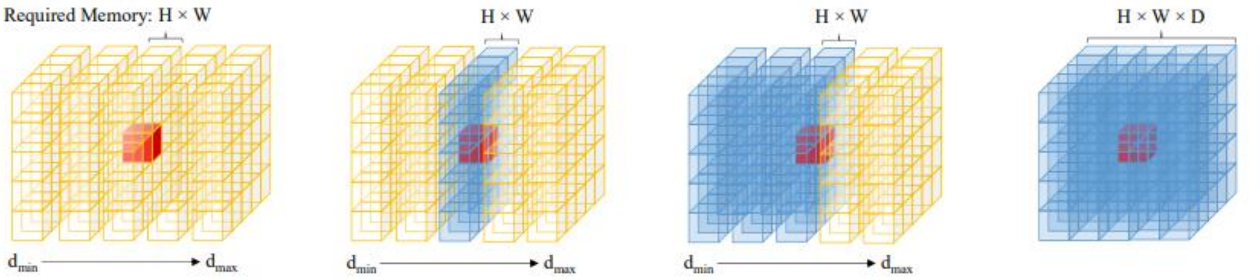


Picture from [2]

Recurrent cost-volume processing

Problem:
3D convolutions are extremely high memory consuming. Using 2D convolutions would reduce the receptive field to a single slice of the cost volume along depth dimension

Possible Solution:
Use a recurrent 2D convNet to process the cost volume



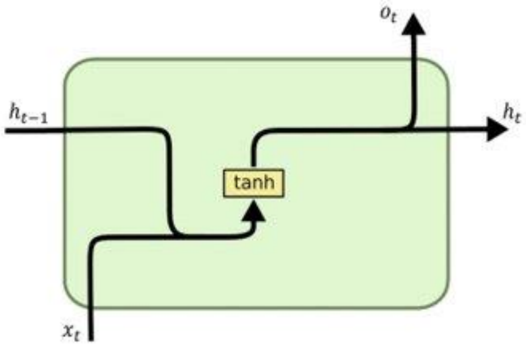
(a) Winner-take-all (b) Spatial Regularization (c) Recurrent Regularization (Proposed) (d) 3D CNNs Regularization

Picture from [4]

Recurrent cost-volume processing

Recurrent layers:
nodes processing data **sequentially**, by maintaining an internal **memory** (or **state**)

Recurrent NN (RNN)



x_t : input vector $(m \times 1)$.
h_t : hidden layer vector $(n \times 1)$.
o_t : output vector $(n \times 1)$.
b_h : bias vector $(n \times 1)$.
U, W : parameter matrices $(n \times m)$.
V : parameter matrix $(n \times n)$.
σ_h, σ_y : activation functions.

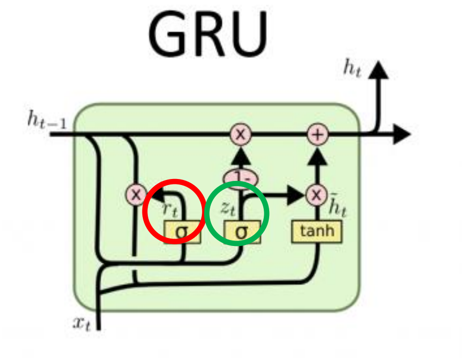
$$h_t = \sigma_h(i_t) = \sigma_h(U_h x_t + V_h h_{t-1} + b_h)$$

$$o_t = \sigma_y(a_t) = \sigma_y(W_y h_t + b_h)$$

short-term memory (suffers of **vanishing gradients** over longer sequences)

Recurrent cost-volume processing

Gated Recurrent Unit (GRU) and **Long-Short Term Memories (LSTM)** are thought to overcome the vanishing gradients problem



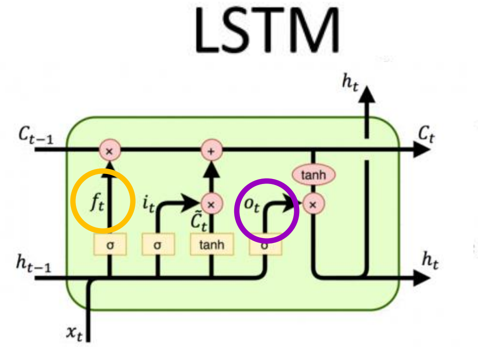
h_t : hidden layer vectors.
x_t : input vector.
b_z , b_r , b_h : bias vector.
W_z , W_r , W_h : parameter matrices.
σ , tanh : activation functions.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$



h_t , C_t : hidden layer vectors.
x_t : input vector.
b_f , b_i , b_c , b_o : bias vector.
W_f , W_i , W_c , W_o : parameter matrices.
σ , tanh : activation functions.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$h_t = o_t \odot \tanh(C_t)$$

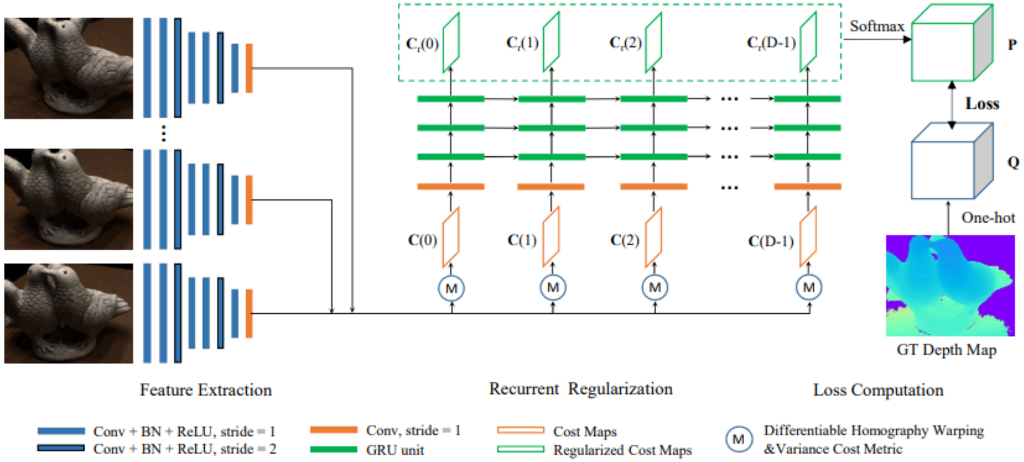
GRU adds **reset** and **update** gates

LSTM also adds **forget** and **output** gates

R-MVSNet [4]

Architecture similar to MVSNet. The 3D networks used to regularize the cost volume is replaced by a recurrent 2D network.

Othe minor differences:
pixels sampled in inverse depth space, network trained for multi-class classification followed by variational refinement

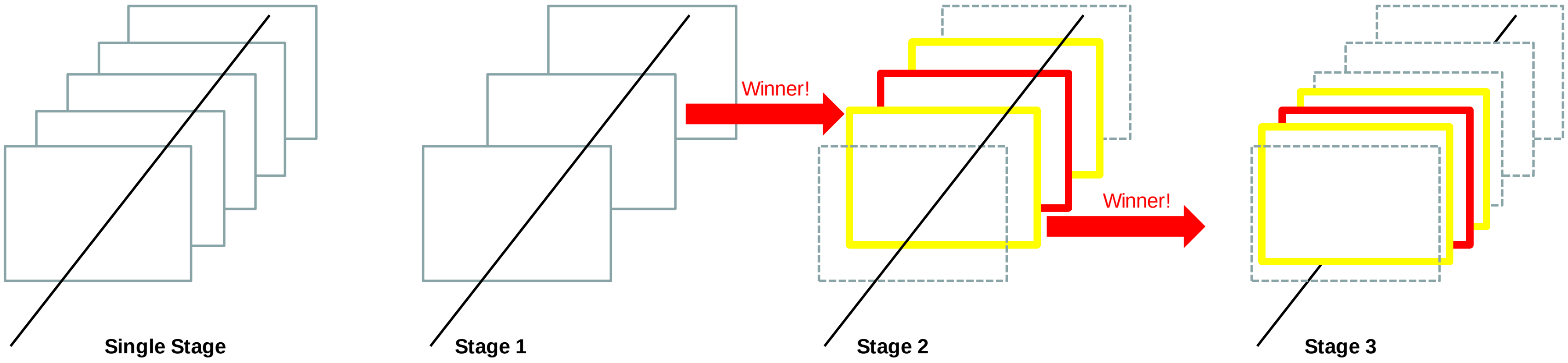


Picture from [4]

Coarse-to-fine processing

Problem:
3D convolutions are extremely high memory consuming.

Possible Solution:
Coarse-to-fine strategy to build smaller cost volumes



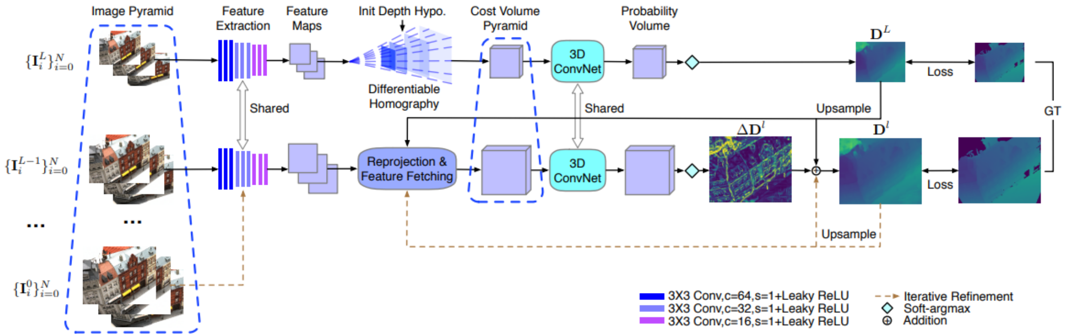
CVP-MVSNet [5], CAS-MVSNet [6]

The feature extractor is designed to output several sets of features at **different resolutions** (from coarser to finer).

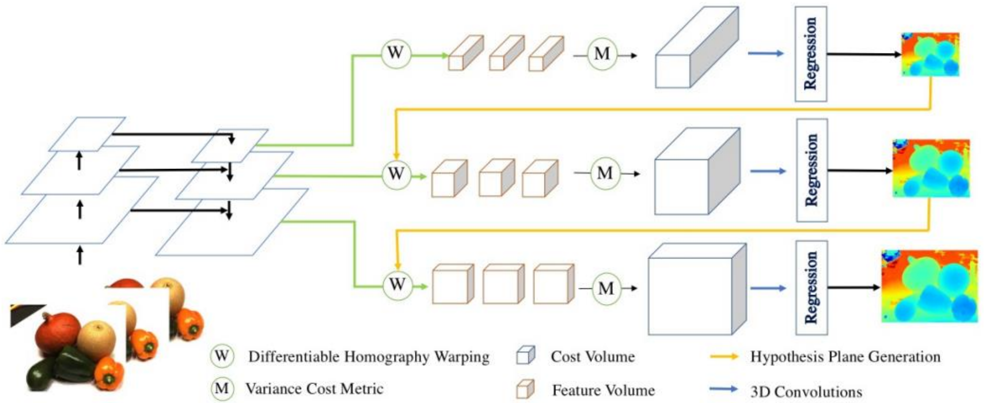
Starting from the finest, a cost volume is built and processed by the 3D convNet to output an initial depth map.

Such depth map is upsampled and used to guide cost-volume building at the higher resolutions, until depth is estimated at the highest resolution.

This sequential protocol allows for **smaller** cost volumes, which are sequentially built at **finer** levels.



Picture from [5]

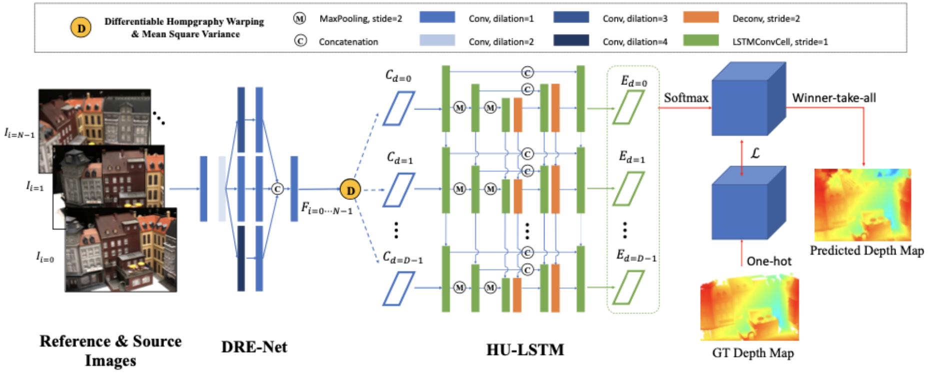


Picture from [6]

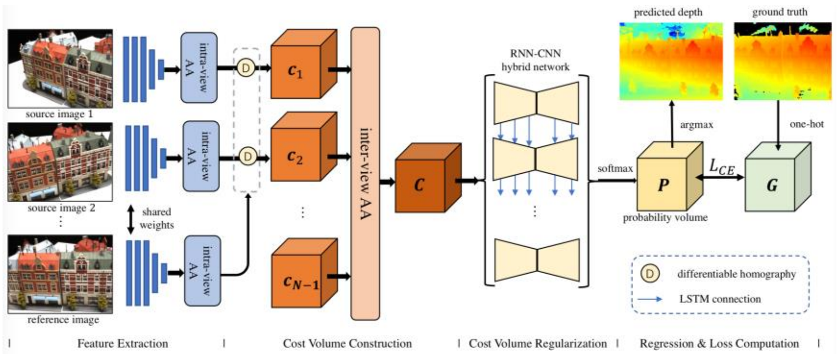
Follow-ups

Several architectures have been built on the two, aforementioned strategies. Among them:

Recurrent cost-volume processing
D2HC-RMVSNet [7], AA-RMVSNet [8], …



Picture from [7]

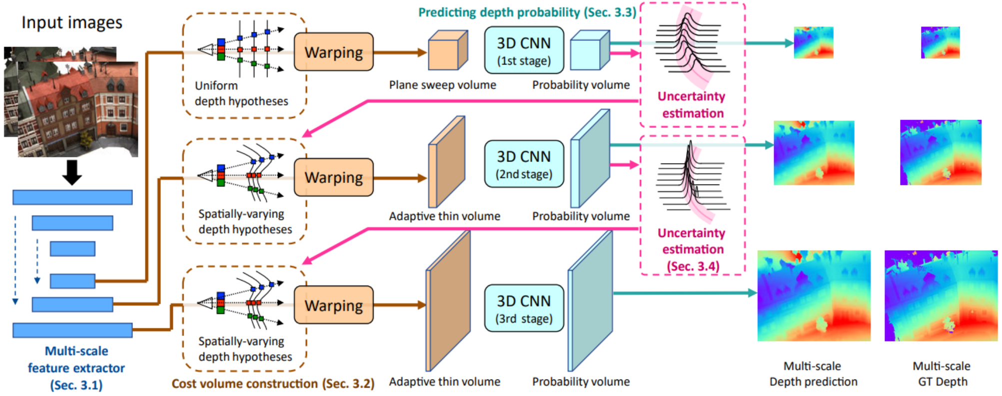


Picture from [8]

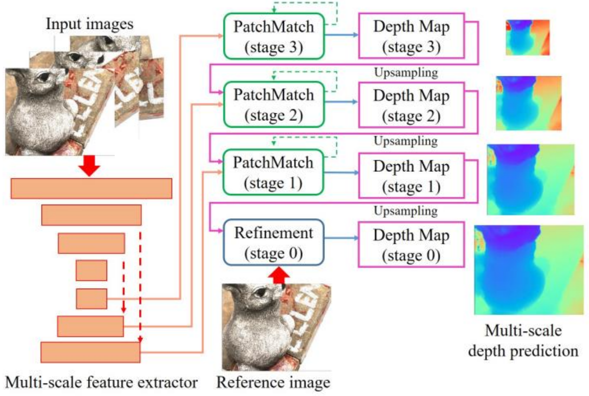
Follow-ups

Several architectures have been built on the two, aforementioned strategies. Among them:

Coarse-to-fine strategies
UCSNet [9], PatchMatchNet [10], …



Picture from [9]



Picture from [10]

In the interest of time, we won’t see them in detail – good candidates for the **final assignment :)**

Self-supervised MVS

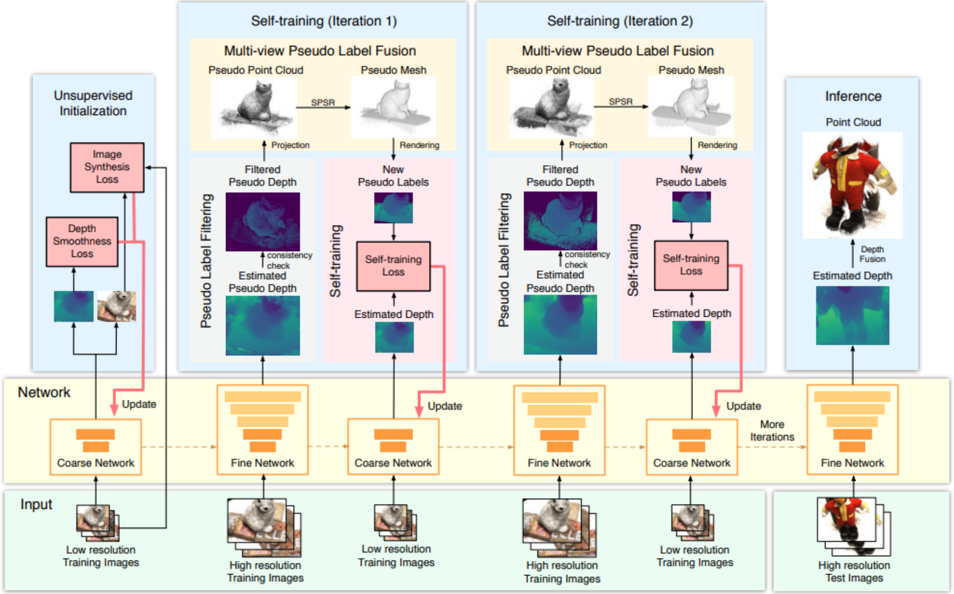
As for other depth-related tasks (stereo and mono), some works deal with self-supervised strategies. However, occlusions are much more severe in this setting. Then, most approaches design some **self-training** mechanism to improve supervision in occluded regions.

SS-CVP-MVSNet [11]
A framework built on top of CVP-MVSNet [5]

A 2-levels CVP-MVSNet is first trained with image synthesis losses (similarly to monodepth).

Then, the model is extended to 5 levels and used to distill pseudo-labels, fused across the entire scene to render filtered labels.
A 2-level instance of CVP-MVSNet is fine-tuned on such labels.

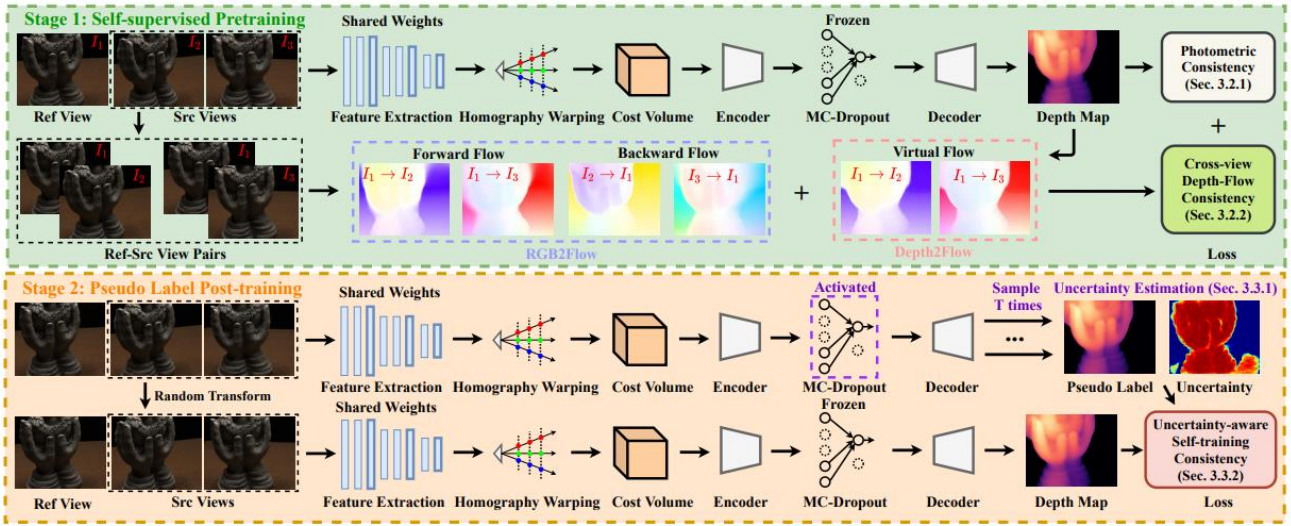
This is repeated for a few iterations.



Picture from [11]

Self-supervised MVS

U-MVSNet [12]
A first instance of a MVSNet is trained with image synthesis losses + depth-flow consistency.
Then, a second stage is performed by self-training on the pseudo-labels produced by the model itself, by taking into account the **uncertainty** modeled with Monte-Carlo Dropout.

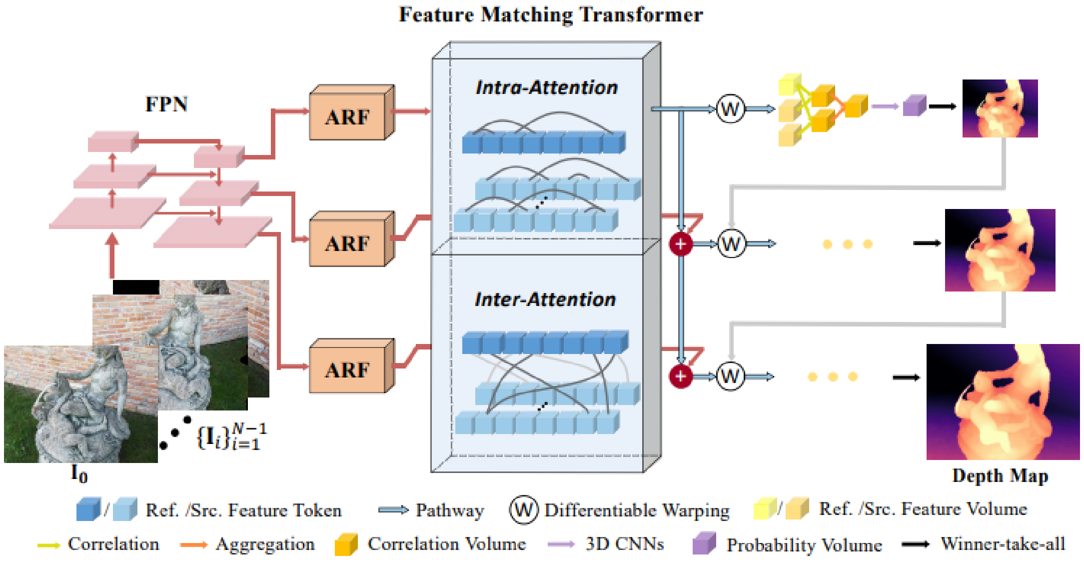


Picture from [12]

Multi-View Stereo with Transformers

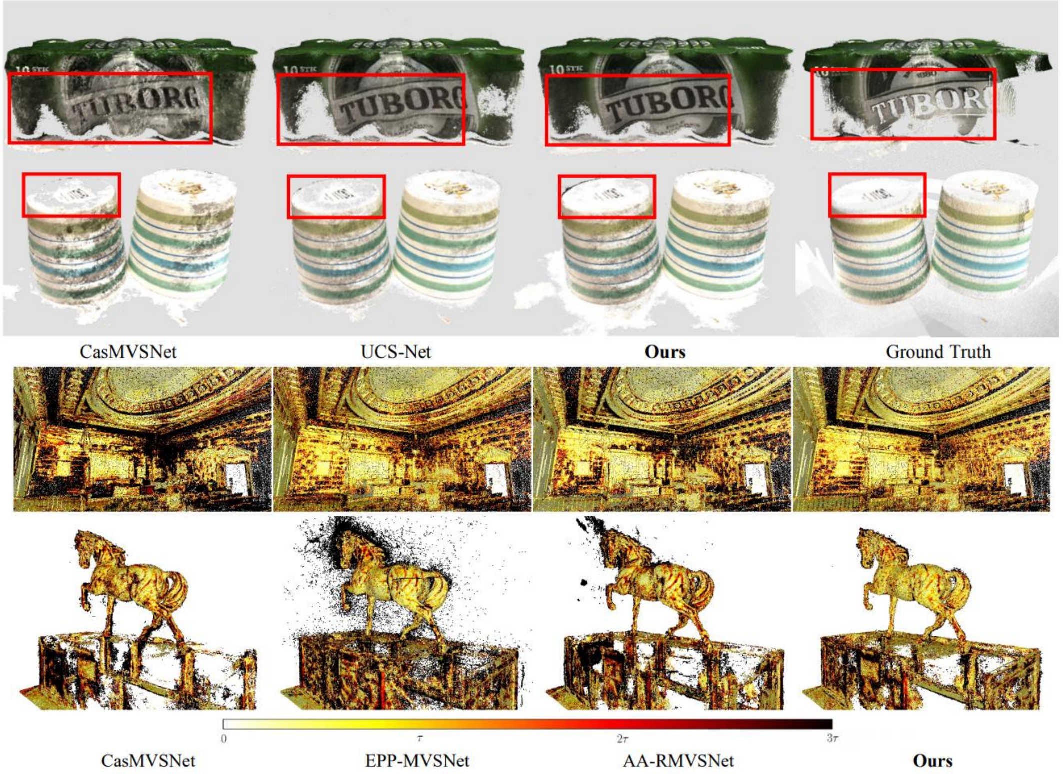
TransMVSNet [13] applies principles from Transformers (intra and inter attention across features) to build a coarse-to-fine architecture.

Cost volumes are built by **pair-wise features correlations**, then combined as a weighted sum according to each pair-wise highest per-pixel correlation.



Picture from [13]

Multi-View Stereo with Transformers



Picture from [13]

6.2- Optical Flow

Optical Flow

With optical flow, we usually refer to the motion vectors connecting pixels coordinates in one image to the corresponding coordinates in a second (usually subsequent) one.

Motion field: projection of 3D motion into an image (**real motion**)
Optical Flow: motion of pixels in the image caused by brightness changes (**apparent motion**)

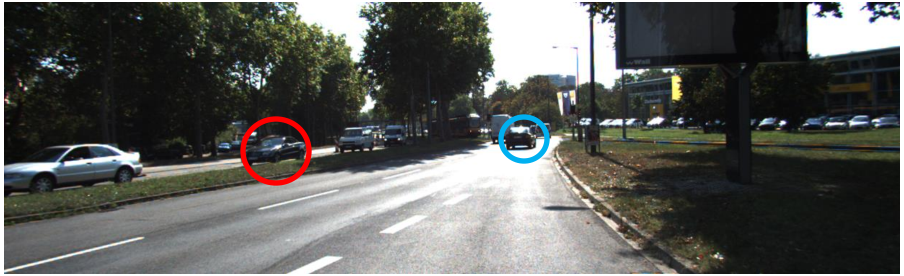
Ideally, the two are **the same.** In practice: shadows, brightness consistency violation, etc.

Optical Flow

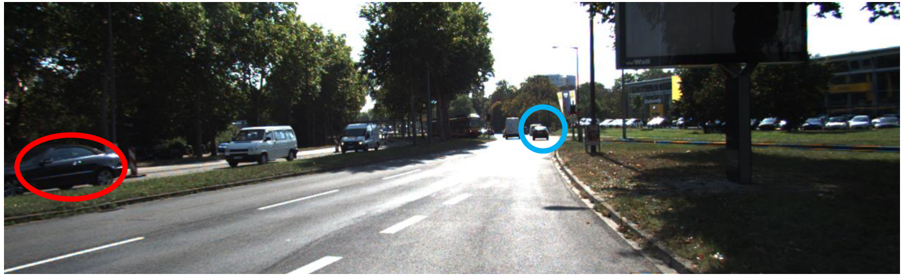
Let’s focus on **motion field**

Optical flow as motion field is consequence of two kinds of motion: **camera motion** (ego-motion) and **independent motions** (objects motion)

In both cases, the magnitude of flow vectors is also consequence of the **distance** from the camera (with a given speed, an object closer to the camera will produce flow vectors with higher magnitude)



Optical flow can be computed by knowing depth and camera poses for **static points** in the scene

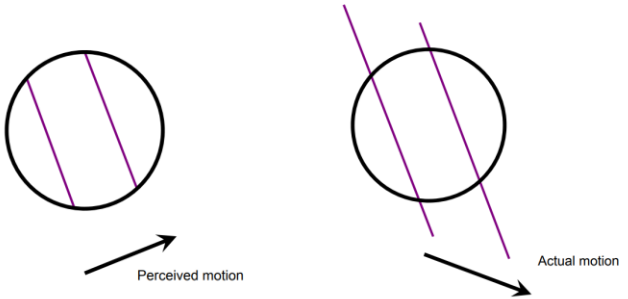


Challenges of Optical Flow

What makes optical flow hard as a **matching problem**?

Search range: 2D, potentially very large search space
Solution: coarse-to-fine strategies

Aperture problem: the lack of context can result in wrong motion estimation (consequence of 2D search)
Solution: wider context + spatial coherence (nearby pixels share the same motion)



Occlusions: pixels disappearing because of objects motion itself

Motion blur: blurring artefacts caused by high-speed motion

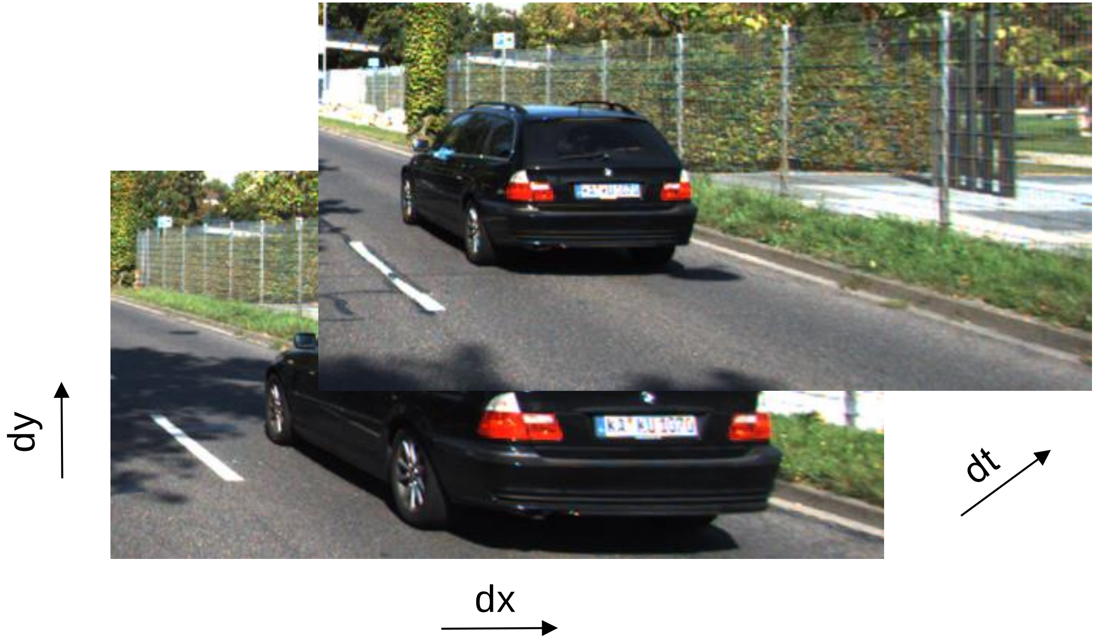
…

MPI-Sintel dataset

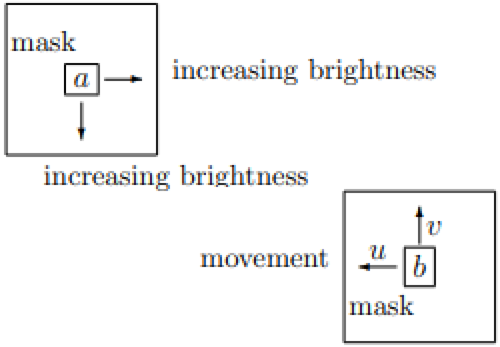
Optical flow in unconstrained conditions is extremely hard!

Optical Flow

Image derivatives



Optical Flow



Lucas-Kanade algorithm is a **differential method**.
Let's assume we look at the scene through a square patch. At a certain time frame, its **intensity is a**. After moving, its intensity **increases to b**.

$$I_x(x,y) \cdot u + I_y(x,y) \cdot v = -I_t(x,y)$$

We can apply this relationship to all pixels in the patch

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\ &\vdots \\ I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n) \end{aligned} \quad \Longrightarrow \quad Av = b$$

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \quad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix} \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

We get a system with variables << equations. A compromise solution is obtained by solving the following 2x2 system with **least square principle**

$A^T A v = A^T b$ or

$\mathbf{v} = (A^T A)^{-1} A^T b$

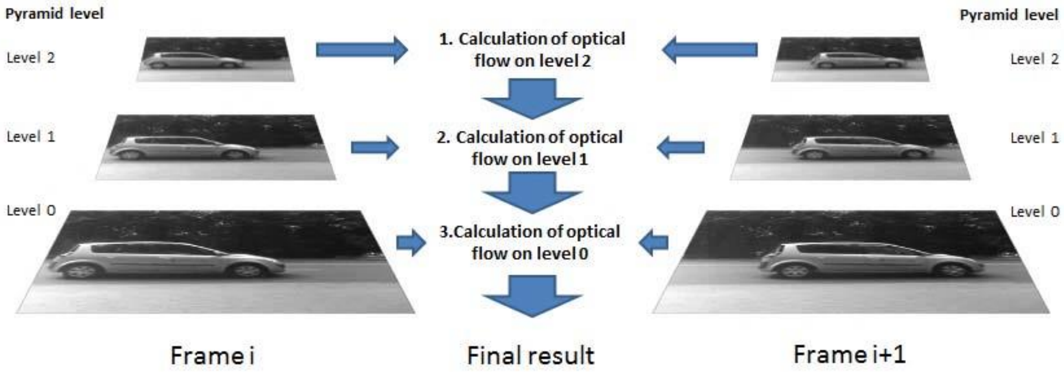
$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_y(q_i)I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix}$$

Optical Flow

Pre deep-learning methods (just a list, in the interest of time…)

Some classical algorithms: **Lucas-Kanade** (1981), **Horn and Schunck** (1981)
More recent methods: **DeepFlow** (2013), **EpicFlow** (2015), **CPM** (2016 – PatchMatch!), **RICFlow** (2017)

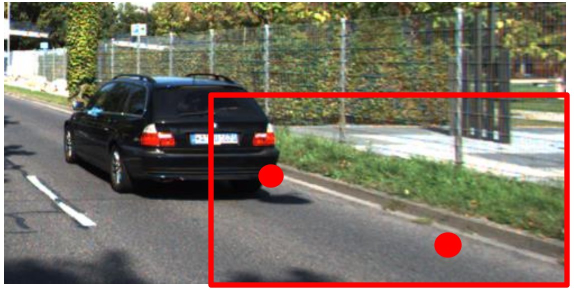
Most of them run coarse-to-fine estimation



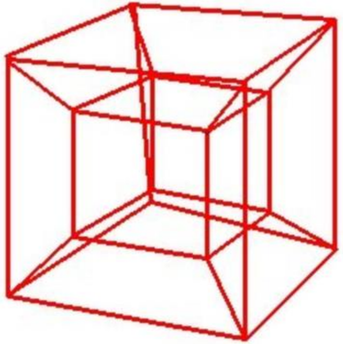
The advent of deep learning revolutioned this field, both in terms of **accuracy** and **speed**

Optical Flow

Cost volume search
As for stereo matching (and MVS), we can directly search for correspondences across the two images – with **deep learning**, maybe? :)



We can compare each pixel (or patch) in the first image with a set of candidates in the second one



Unfortunately, this time our search domain is 2D and can possibly be **huge**!

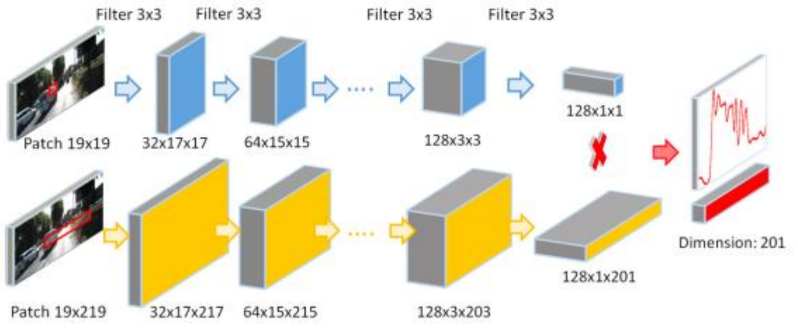
This would lead to a **4D cost volume** (HxWxdHxdW, with dHxdW being the 2D search range)

Such a data structure cannot be handled on full resolution images (**coarse-to-fine strategies**)

Optical Flow

Semantic Information and Deep Matching [14]

This work follows the trend started by Zbontar and LeCun with MC-CNN.



Picture from [14]

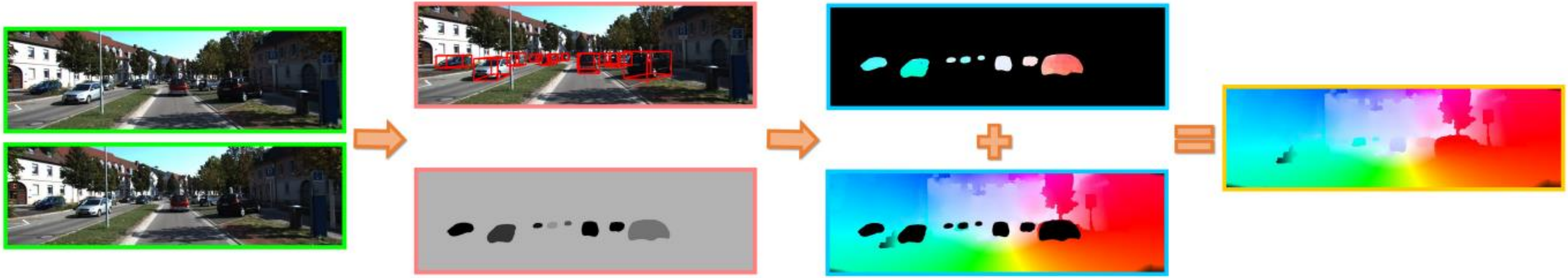
A siamese network processes 19x19 patches extracted from the two images. Since optical flow demands much more complexity (a single patch in the first image should be compared with RxR patches in the second, in a 2D search range), a few heuristic are introduced:

- At training time, patches are matched along a single axis (vertical or horizontal)
- At test time, only top-K scores are kept (K=30) to save memory

Then, the top-K costs are refined through iterative local aggregation (box-filtering like).

Optical Flow

Semantic Information and Deep Matching [14]



Picture from [14]

Then, objects are detected and segmented by means of a CNN, to distuingish **moving vehicles** from the **static background.**

Finally, the optical flow initially estimated by the network is **refined** by means of hand-crafted algorithms modeling the motion of the scene for **static** and **dynamic** agents independelty**

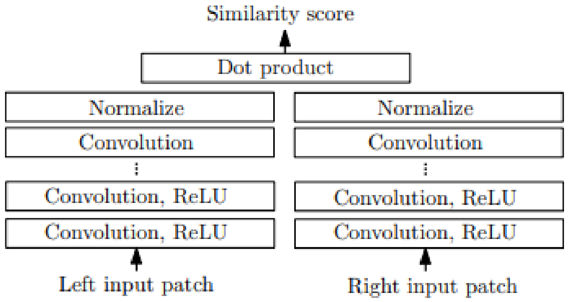
**based on a pipeline combining RANSAC (to get fundamental matrices), SGM (to perform matching along epipolar lines) and more…

Optical Flow

Direct Cost volume optimization – **DC-Flow** [15]

A siamese network processes 9x9 patches extracted from the two (**downsampled** by a factor 3) images

It computes similarity scores between a single patch on the first frame and RxR patches in the second frame (2D search range)



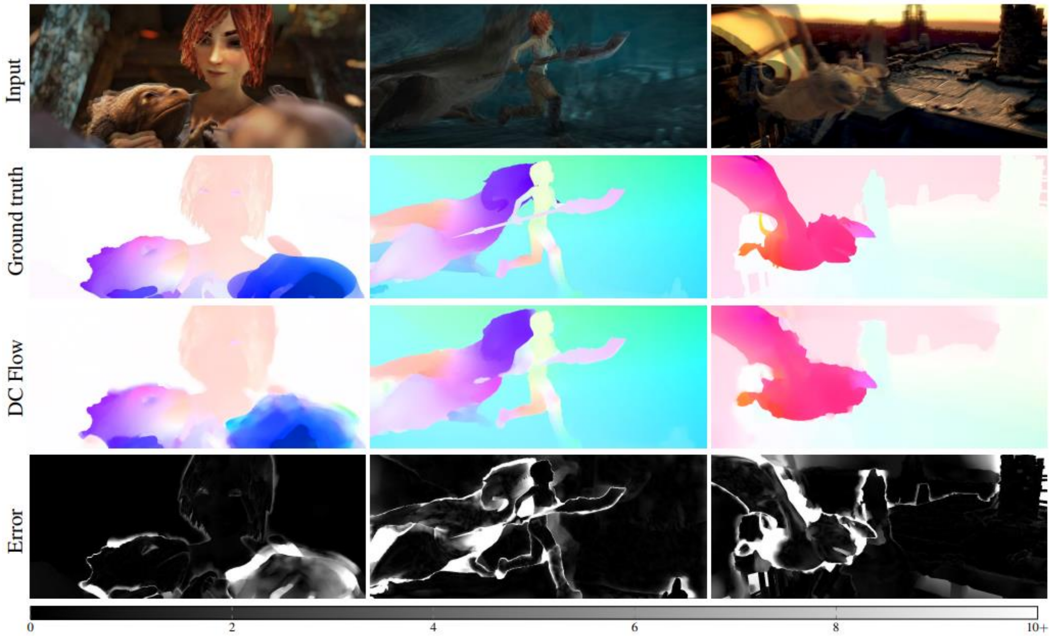
A 4D cost-volume is built and then refined by a variant of Semi-Global Matching (SGM) specifically designed to deal with 4D volumes, **Flow-SGM.** The downsampling factor allows to reduce the memory requirements by a factor 3^4!

$$E(\mathbf{V}) = \sum_p \Bigg(\sum_{q \in \mathcal{N}(p)} P_1 [\|\mathbf{V}_p - \mathbf{V}_q\|_1 = 1] + \sum_{q \in \mathcal{N}(p)} P_2^{p,q} [\|\mathbf{V}_p - \mathbf{V}_q\|_1 > 1] + \mathbf{C}(p, \mathbf{V}_p) \Bigg)$$

With V_p, V_q being flow vectors hypotheses

The final flow is obtained through **WTA** and **upsampled** by a factor 3

Optical Flow



Results looks good, yet showing limitations in occlusions, large motions, …

… all aspects that can be dealt with an **end-to-end model**!

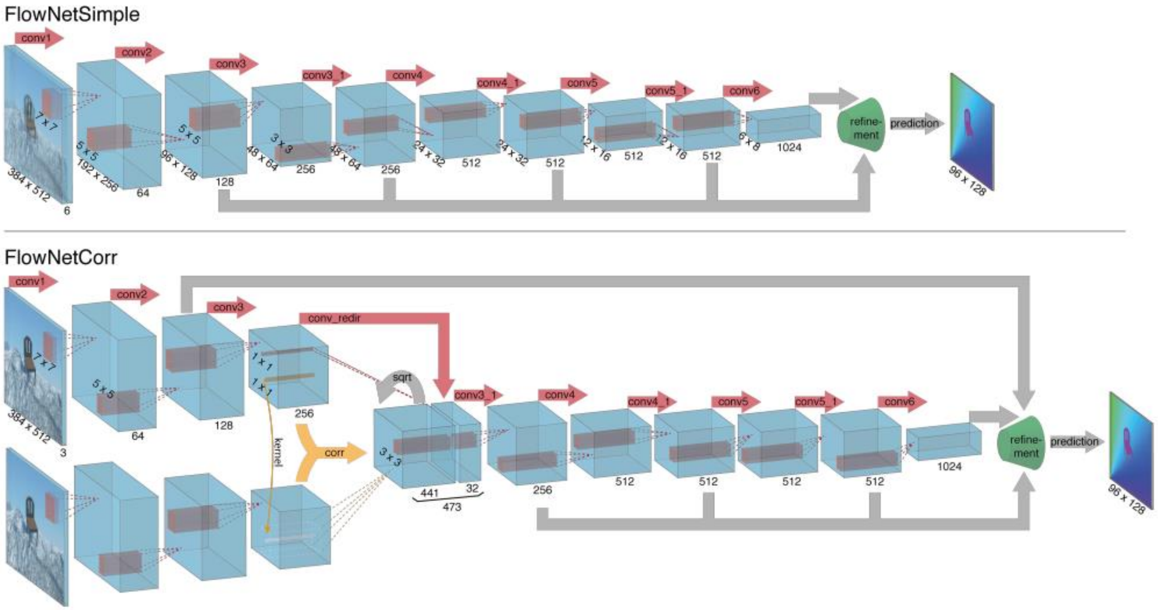
FlowNet [16]

First end-to-end optical flow estimation network. Two main components:

1) Features extractor (encoder)
2) Refinement module (decoder)

Two variants:

1) FlowNetS
2) FlowNetC (Correlation)



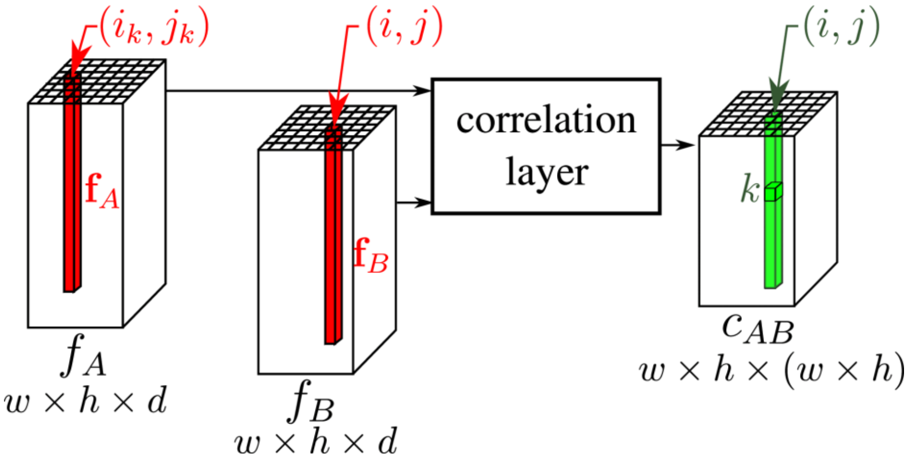
Picture from [16]

FlowNet [16]

Correlation Layer: a module computing the correlation scores between features extracted from two different images.

For a given pixel (i,j) in f_B, this layer computes correlation between it and k pixels in f_A in a neighborhood around (i,j).

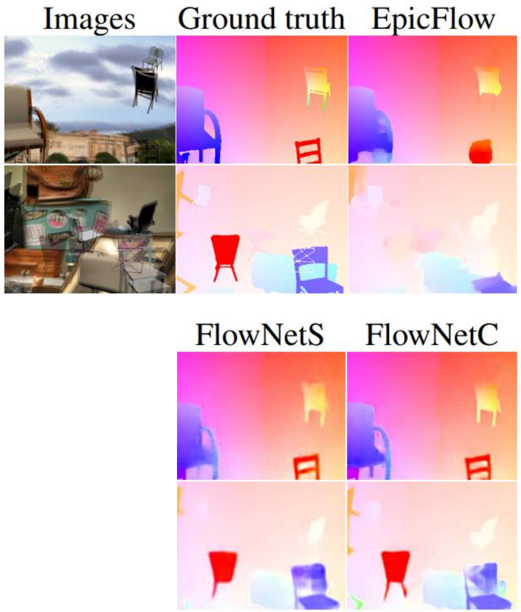
The results are stored in a new features map C_{AB} (the k scores are encoded along the channels dimension)

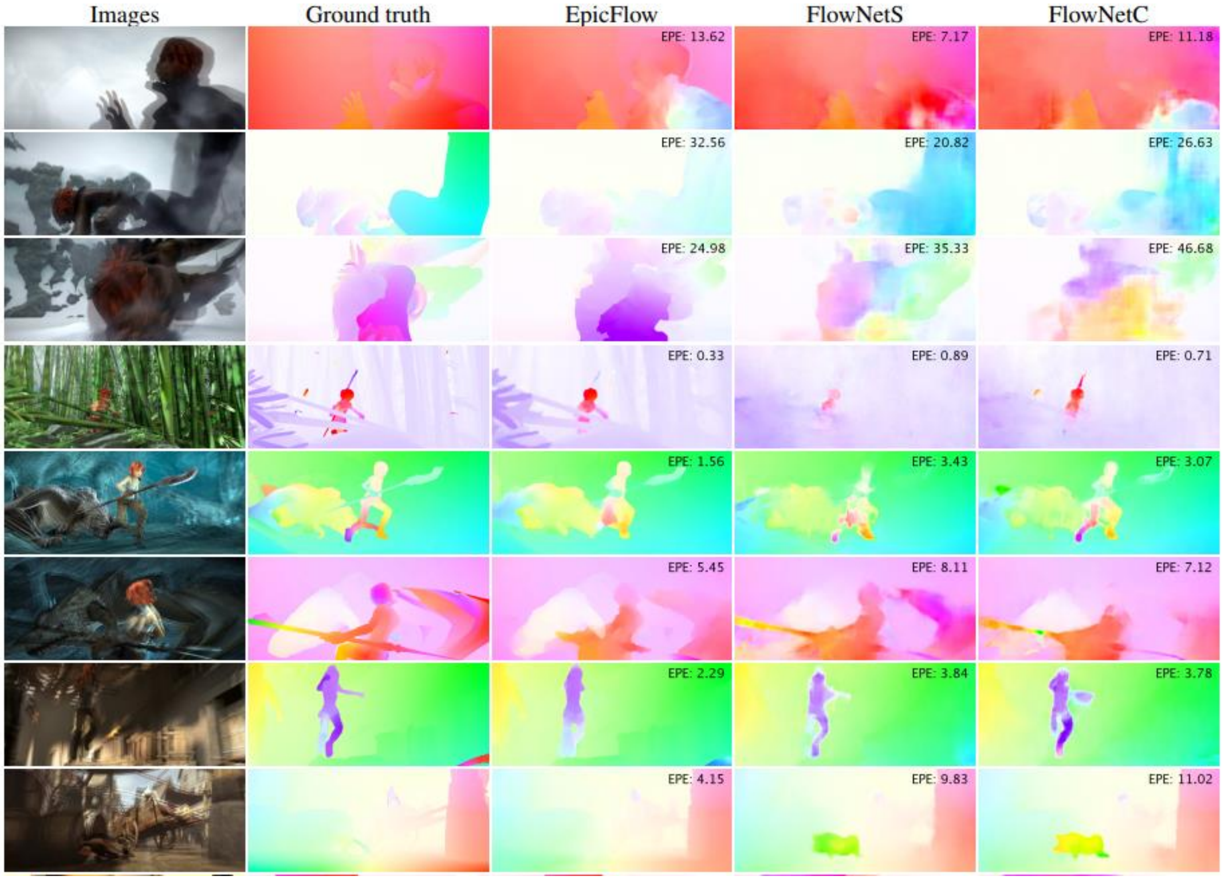


Picture from [17]

FlowNet [16]

Still less effective than existing solutions…





Improving the accuracy with residual refinement

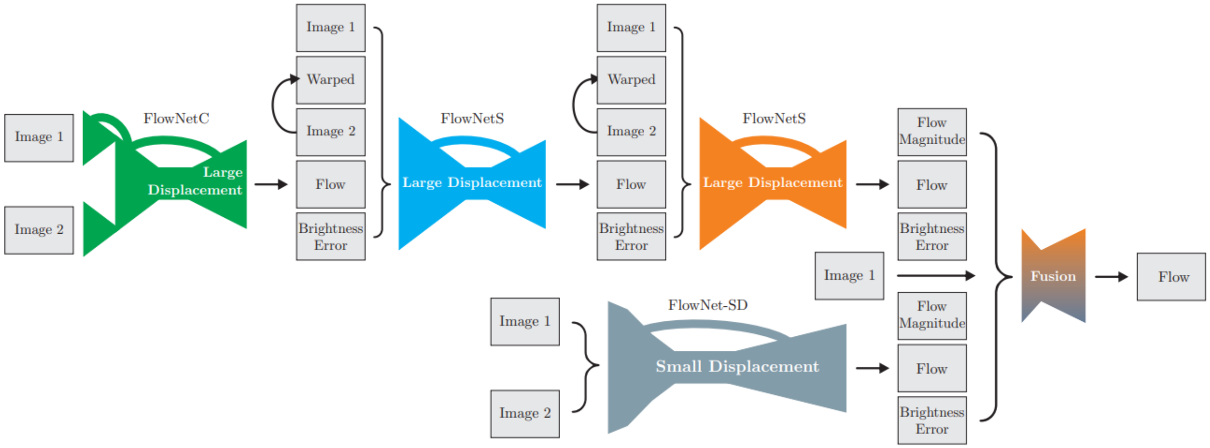
FlowNet2 [18], made of several instances of FlowNetC/S

1) A first FlowNetC, for large displacements

2) Two FlowNetS, to compute residual flow given the two images and the first estimate by FlowNetC

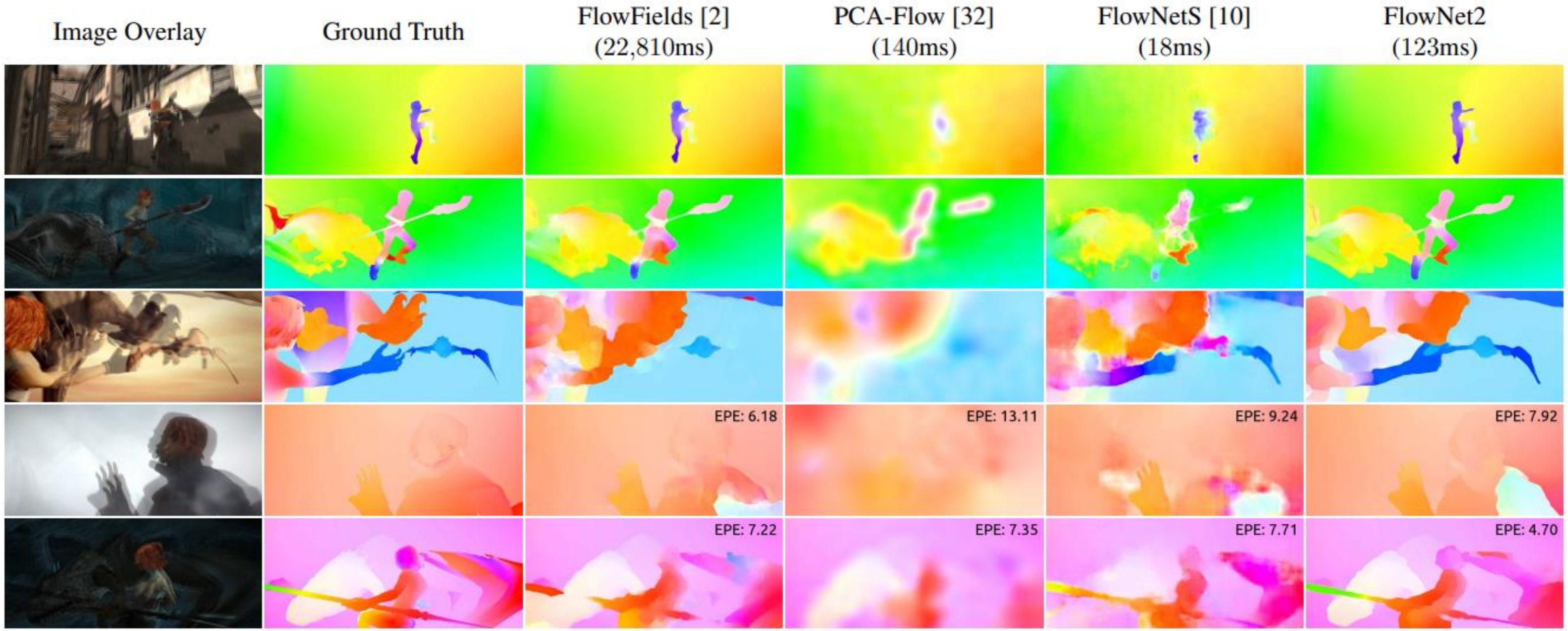
3) A further FlowNet for small displacements (SD)

4) A final fusion module



Picture from [18]

Improving the accuracy with residual refinement



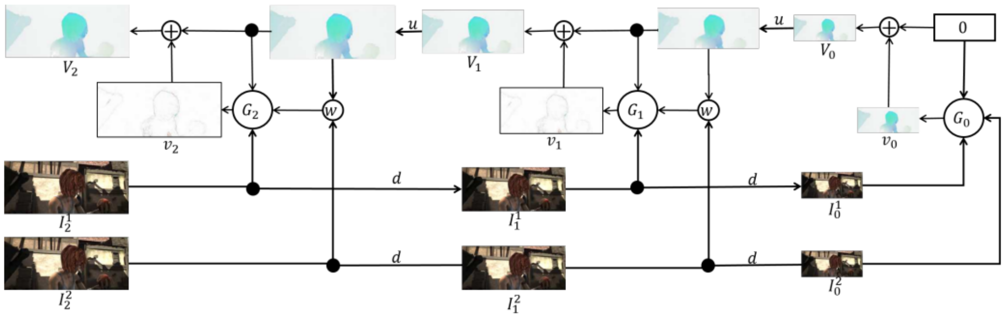
Picture from [15]

Coarse-to-fine processing

Problem: processing full-resolution images is expensive

Possible solution: coarse-to-fine processing!

SpyNet [19] computes optical flow on an **image pyramid**, starting from coarse resolution and going up until reaching full resolution. **No explicit correlation** between pixels is computed.



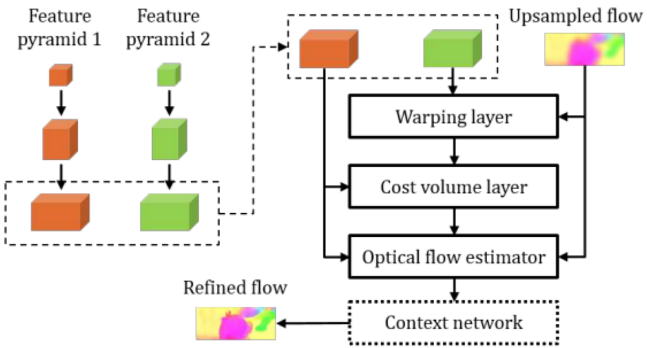
Picture from [19]

Coarse-to-fine processing

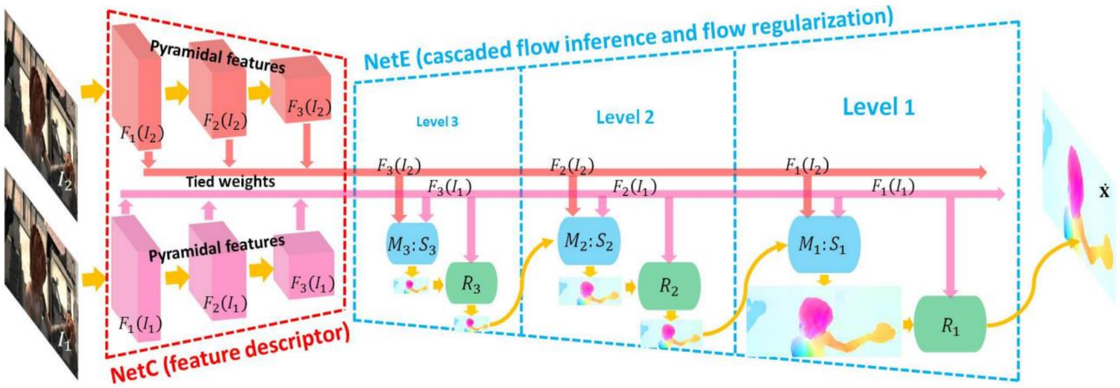
PWCNet [20], LiteFlowNet[21]

Combine established design strategies:

- Pyramidal features extraction
- Cost-volume computation (correlation layer)
- Coarse-to-fine estimation
- Refinement



Picture from [20]



Picture from [21]

More: LiteFlowNet2 [22], IRR-PWCNet [23], LiteFlowNet3 [24], … – good candidates for the **final assignment :)**

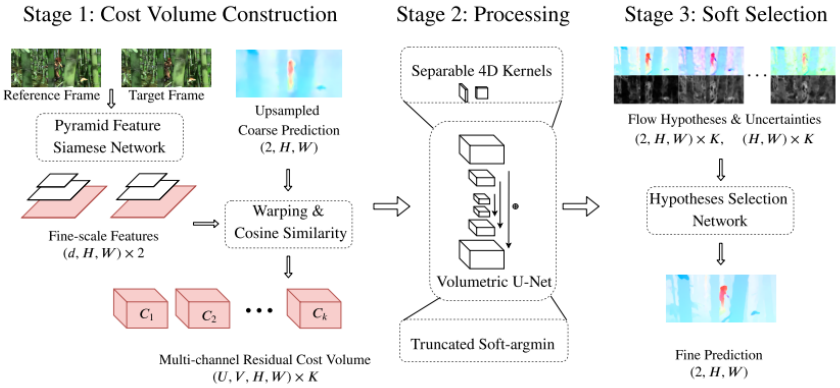
Volumetric representation

VCN [25], DICL [26]

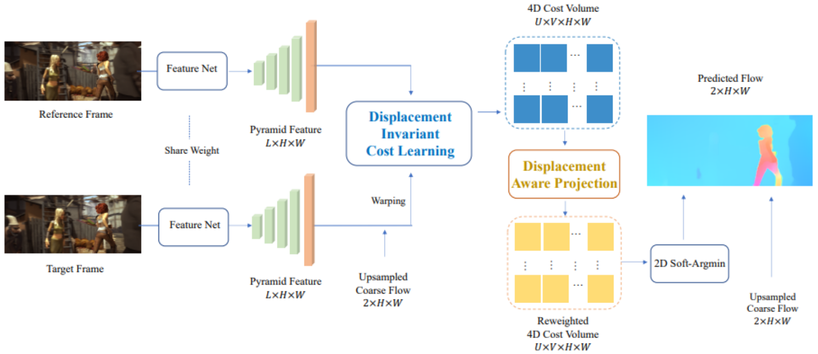
A volumetric representation of the matching costs is more powerful (offset invariance, more general in terms of search windows size). However, for optical flow the cost volume would be a 5D features tensor and require 4D convolutions! So, some **efficient strategies** to avoid 4D convolutions are necessary:

- **VCN: Separable 4D conv**
  splits a 4D conv into a 2D conv + 2D WTA

- **DICL: 2D matching cost net**
  a 2D network processes each «slice» of the 4D volume separately



Picture from [25]



Picture from [26]

RAFT [27]

Iterative optimization, inspired by traditional methods for estimating optical flow

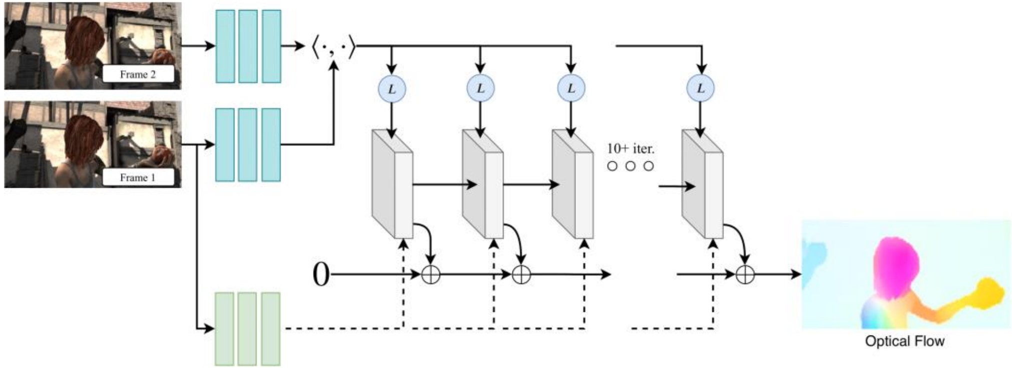
After extracting features from both images, a **correlation look-up table** (LUT) is built, storing the correlation scores among all across the two images (HxW x HxW)

Then, optical flow is iteratively estimated by looking at the look-up table and to some context features

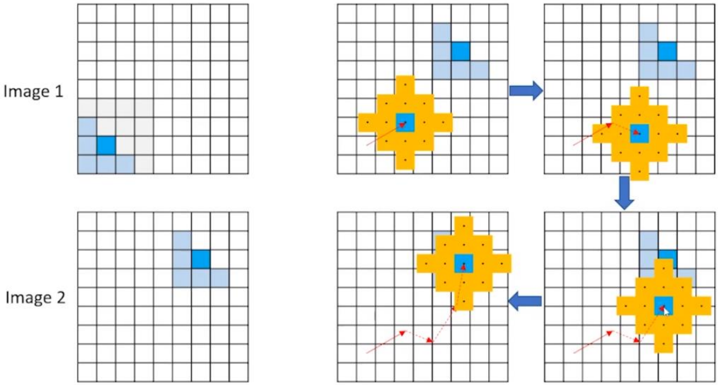
For pixel (i,j) with an initial flow estimate of (u,v), the look-up table is queried at (i+u, j+v) at multiple scales

An updated flow vector (u',v') is estimated, to access to the LUT again and refine it again and again…

LUT scores and context features are processed by **GRUs**



Picture from [27]

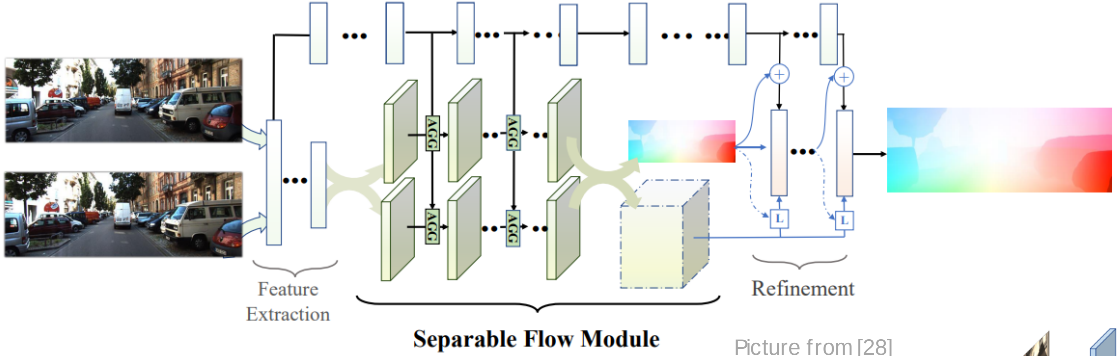


Picture from [https://www.youtube.com/watch?v=r3ZtW30exoo]

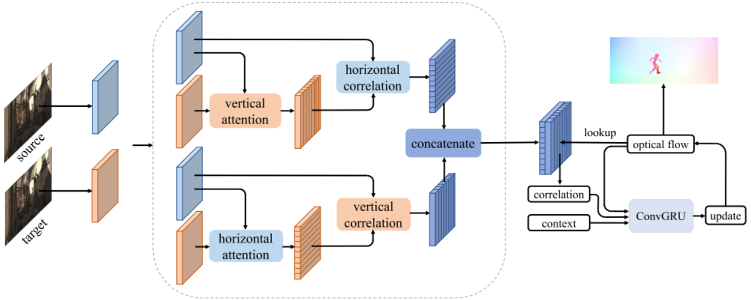
Decomposing 2D flow into 1D flows

SeparableFlow [28], Flow1D [29]

Using a 4D representation while reducing complexity (see VCN and DILC).
In the interest of time, we won't see them in detail – good candidates for the **final assignment :)**



Picture from [28]

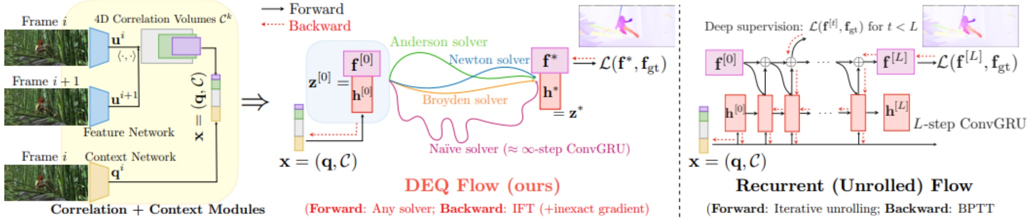


Picture from [29]

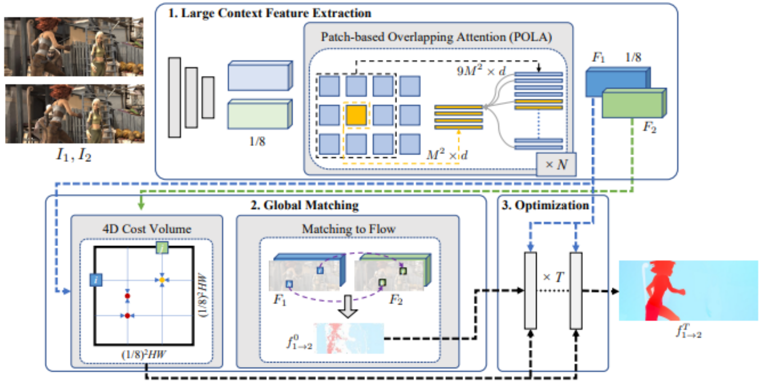
Follow-ups

More recent architectures:

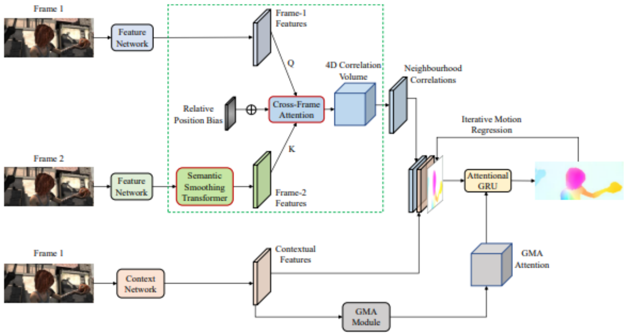
GMFlowNet [30], DEQ [31], CRAFT [32], …



Picture from [31]



Picture from [30]



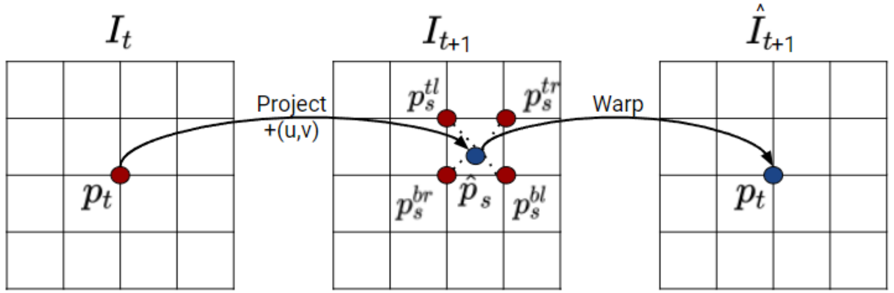
Picture from [32]

In the interest of time, we won't see them in detail – good candidates for the **final assignment :)**

Self-supervised Optical Flow

As for other matching-based tasks (stereo and MVS), some works deal with self-supervised strategies.

Optical flow estimation as an image reconstruction task, by using estimated flow to reconstruct I_t from I_{t+1} (as seen with stereo)



Challenges: occlusions, light changes, shadows from moving objects, …

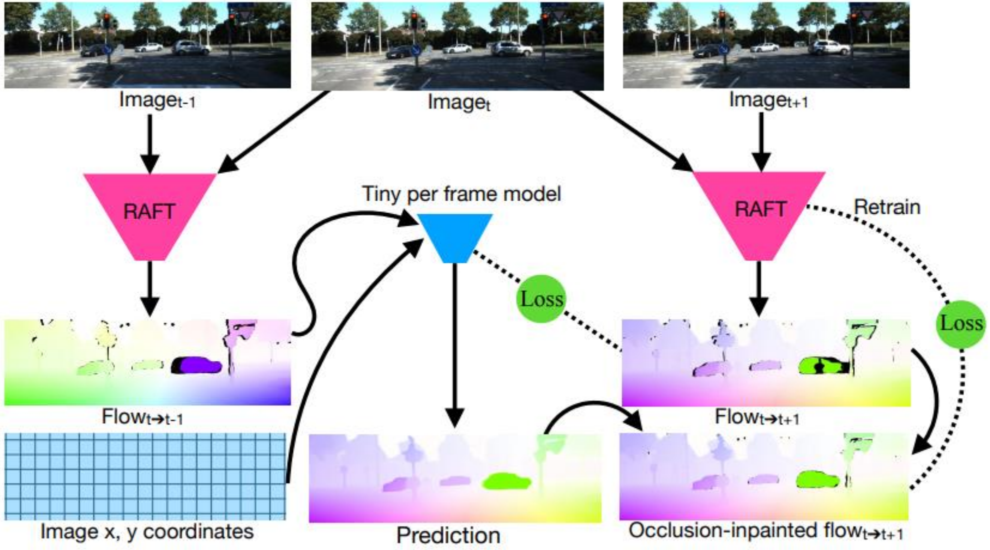
Best practices: [33]
State-of-the-art: [34]

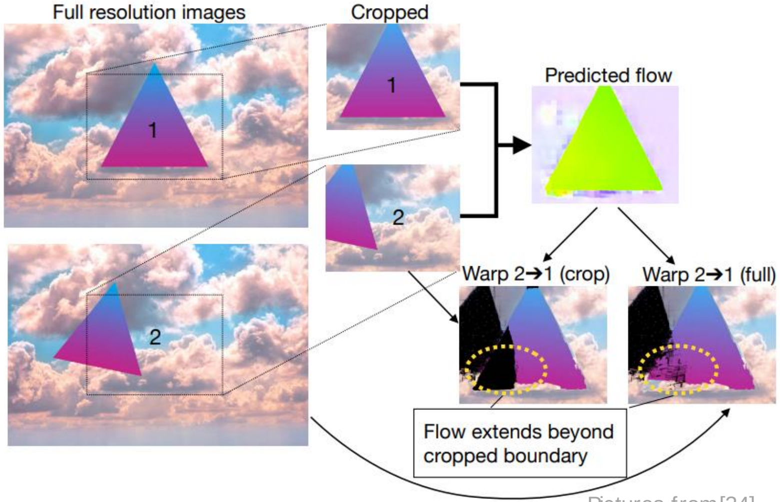
Picture from [34]

Self-supervised Optical Flow

SMURF [34]: RAFT variant designed for self-supervised optical flow

Two main factors:
- **Crop augmentation** – warping performed on full resolution images (handling out of image content)
- **Occlusions inpainting –** a dedicated, per-frame model si trained to inpaint occlusions (generating proxy labels) by inverting the backward flow





Pictures from [34]

References

[1] “Learning descriptor, confidence, and depth estimation in multi-view stereo”, Choi et al., CVPR workshops 2018
[2] “MVSNet: Depth Inference for Unstructured Multi-view Stereo”, Yao et al., ECCV 2018
[3] “Real-Time Visibility-Based Fusion of Depth Maps”, Merrel et al., ICCV 2007
[4] “Recurrent MVSNet for High-resolution Multi-view Stereo Depth Inference”, Yao et al., CVPR 2019
[5] “Cost Volume Pyramid Based Depth Inference for Multi-View Stereo”, Yang et al., CVPR 2020
[6] “Cascade Cost Volume for High-Resolution Multi-View Stereo and Stereo Matching”, Gu et al., CVPR 2020
[7] “Dense Hybrid Recurrent Multi-view Stereo Net with Dynamic Consistency Checking”, Yan et al., ECCV 2020
[8] “AA-RMVSNet: Adaptive Aggregation Recurrent Multi-view Stereo Network”, Wei et al., ICCV 2021
[9] “Deep Stereo using Adaptive Thin Volume Representation with Uncertainty Awareness”, Cheng et al., CVPR 2020
[10] “PatchmatchNet: Learned Multi-View Patchmatch Stereo”, Wang et al., CVPR 2021
[11] “Self-supervised Learning of Depth Inference for Multi-view Stereo”, Yang et al., CVPR 2021
[12] “Digging into Uncertainty in Self-supervised Multi-view Stereo”, Xu et al., ICCV 2021
[13] “TransMVSNet: Global Context-aware Multi-view Stereo Network with Transformers”, Ding et al., CVPR 2022

References

[14] "Exploiting Semantic Information and Deep Matching for Optical Flow", Bai et al., ECCV 2016
[15] "Accurate Optical Flow via Direct Cost Volume Processing", Xu et al., CVPR 2017
[16] "FlowNet: Learning Optical Flow with Convolutional Networks", Dosovitskiy et al., ICCV 2015
[17] "Convolutional Neural Network Architecture for Geometric Matching", Rocco et al., TPAMI 2019
[18] "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks", Ilg et al., CVPR 2017
[19] "Optical Flow Estimation using a Spatial Pyramid Network", Ranjan et al., CVPR 2017
[20] "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume", Sun et al., CVPR 2018
[21] "LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation", Hui et al., CVPR 2018
[22] "Iterative Residual Refinement for Joint Optical Flow and Occlusion Estimation", Hur and Roth, CVPR 2019
[23] "A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization", Hui et al., TPAMI 2020
[24] "LiteFlowNet3: Resolving Correspondence Ambiguity for More Accurate Optical Flow Estimation", Hui and Loy, ECCV 2020
[25] "Volumetric Correspondence Networks for Optical Flow", Yang and Ramanan, NeurIPS 2019
[26] "Displacement-Invariant Matching Cost Learning for Accurate Optical Flow Estimation", Wang et al., NeurIPS 2020
[27] "RAFT: Recurrent All Pairs Field Transforms for Optical Flow", Teed and Deng, ECCV 2020
[28] "Separable Flow: Learning Motion Cost Volumes for Optical Flow Estimation", Zhang et al., ICCV 2021
[29] "High-Resolution Optical Flow from 1D Attention and Correlation", Xu et al., ICCV 2021
[30] "Global Matching with Overlapping Attention for Optical Flow Estimation", Zhao et al., CVPR 2022
[31] "Deep Equilibrium Optical Flow Estimation", Bai et al., CVPR 2022
[32] "CRAFT: Cross-Attentional Flow Transformer for Robust Optical Flow", Sui et al., CVPR 2022
[33] "What Matters in Unsupervised Optical Flow", Jonschkowski et al., ECCV 2020
[34] "SMURF: Self-Teaching Multi-Frame Unsupervised RAFT with Full-Image Warping", Stone et al., CVPR 2021