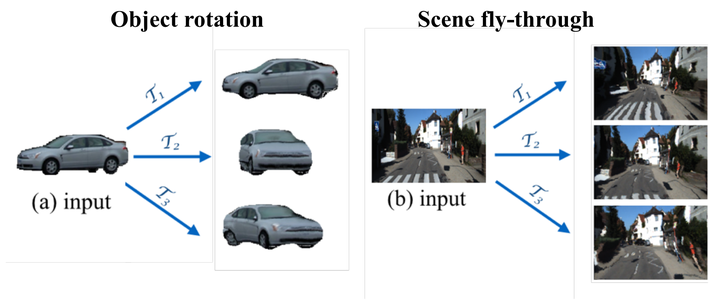
View Synthesis by Appearance Flow

{ Tinghui Zhou, Shubham Tulsiani, Weilun Sun,
Jitendra Malik and Alexei A. Efros }@UC Berkeley

Problem

Novel view synthesis: given an input image, synthesizing new images of the same object or scene seen from novel viewpoints.



Why is it interesting:

- Humans excel at “mental rotation" [5].
- 3D geometry + appearance modeling
- Key problem in virtual reality.

Previous approaches:

- *Geometry based* [6, 7]: high visual quality, but requires user intervention and 3D shape collection at test time.
- *Learning based* [1, 4]: no user intervention or shape collection needed, but misses high-frequency texture details.

Method

Key insight: high visual correlation across different views → pixels to be synthesized likely exist in the input view.

Appearance flow: 2-D coordinate vectors specifying where to *copy* pixels to reconstruct the target view.

Learning to predict appearance flow:

$$\text{minimize} \sum_{<I_s, I_t, T> \in \mathcal{D}} \|I_t - g(I_s, T)\|_p, \quad \text{subject to } g^{(i)}(I_s, T) \in \{I_s\}, \forall i$$

Target view; Synthesis CNN; Source view; Relative Viewpoint; Reuse pixels from the source

The above constraint can be rewritten in the form of bilinear sampling [2]:

$$g^{(i)}(I_s, T) = \sum_{q \in \{\text{neighbors of } (x^{(i)}, y^{(i)})\}} I_s^{(q)} (1 - |x^{(i)} - x^{(q)}|)(1 - |y^{(i)} - y^{(q)}|)$$

Learning to leverage multiple input views:

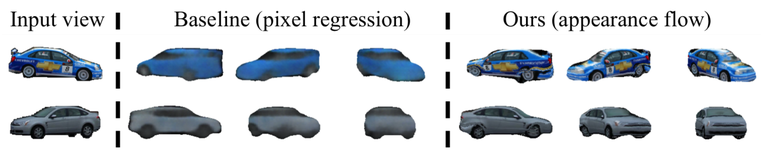
1. Extra output channel for confidence of each single-view prediction.
2. Final prediction = per-pixel weighted sum (weight determined by confidence) over all single-view predictions.

References

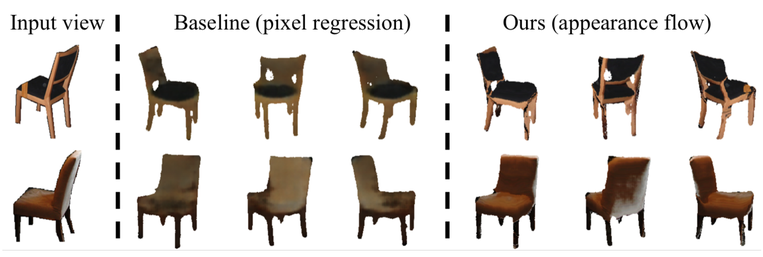
[1] Multi-view 3D Models from Single Images with a Convolutional Network, Tatarchenko *et al.*, *ECCV'16*
[2] Spatial Transformer Networks, Jaderberg *et al.*, *NIPS'15*
[3] DeepStereo: Learning to Predict New Views from the World's Imagery, Flynn *et al.*, *CVPR'16*
[4] Deep convolutional inverse graphics network, Kulkarni *et al.*, *NIPS'15*
[5] Mental Rotation of Three-Dimensional Objects, Shepard *et al.*, *Science'71*
[6] Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach, Debevec *et al.*, *SIGGRAPH'96*
[7] 3D Object Manipulation in a Single Photograph using Stock 3D Models, Kholgade *et al.*, *SIGGRAPH'14*

Why appearance flow

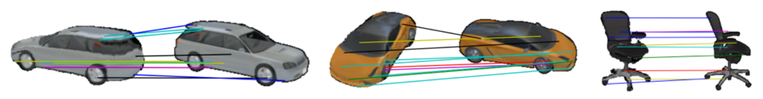
1) Avoids perceptual blurriness caused by naive L_p loss minimization – no longer allowed to predict the ‘mean’ that minimizes the error but loses high-frequency details.



2) Color identity is preserved by construction – can only use existing pixels.

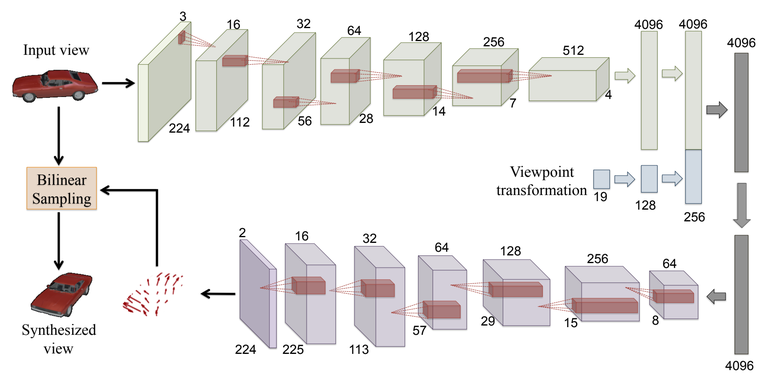


3) Interpretable results – can visualize exactly how each output image is constructed.

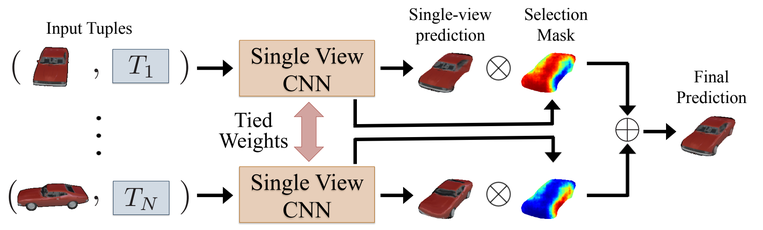


Network architecture

Single input view:



Multiple input views:



Our multi-vew architecture can in theory utilize arbitrary number of input views.

ShapeNet results

Data setup:

- $7,497$ cars and 700 chairs split into 80% training and 20% testing.
- Each shape is rendered for 504 viewing angles (azimuth = $0° - 355°$, elevation = $0° - 30°$ both at steps of $5°$)
- Viewpoint transformation limited to azimuth only with the range of $-180° - 160°$ at steps of $20°$.

Quantitative measure (mean L_1 pixel error):

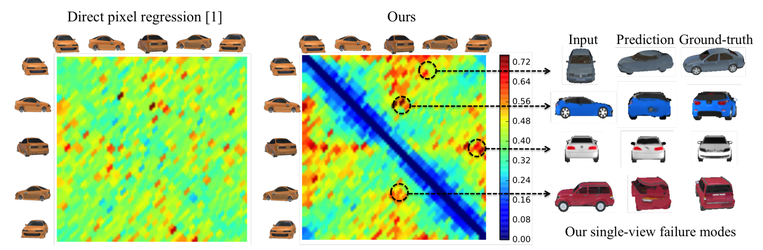
Input	Method	Car	Chair	KITTI
Single-view	Tatarchenko *et al.* [1]	0.404	0.345	0.492
	Ours	**0.368**	**0.323**	**0.471**
Multi-view	Tatarchenko *et al.* [1]	0.385	0.334	0.471
	Ours	**0.285**	**0.248**	**0.409**

ShapeNet results (contd)

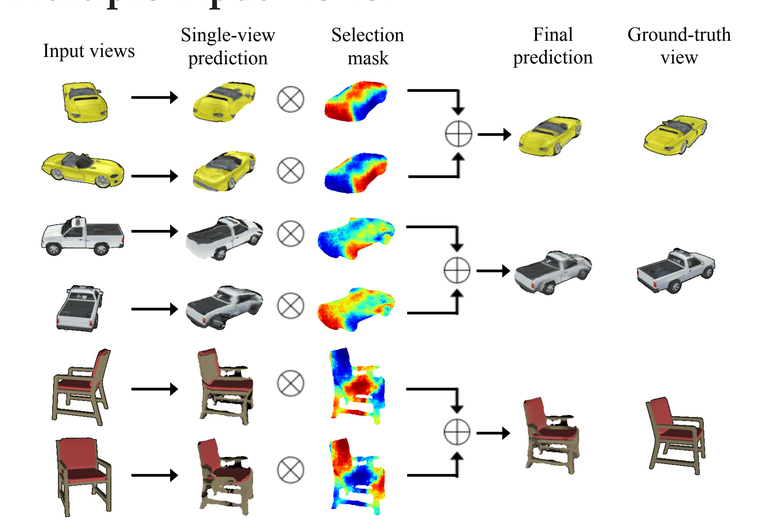
Single input view:

Error analysis for single view prediction:

- Ours is especially strong when cross-view correlation is high (within ± 45 azimuth variation or along the corresponding symmetry planes).
- Ours struggles when synthesizing ‘new’ pixels is required (e.g. wheels when going from frontal to the side view), which is effectively resolved when given multiple input views.



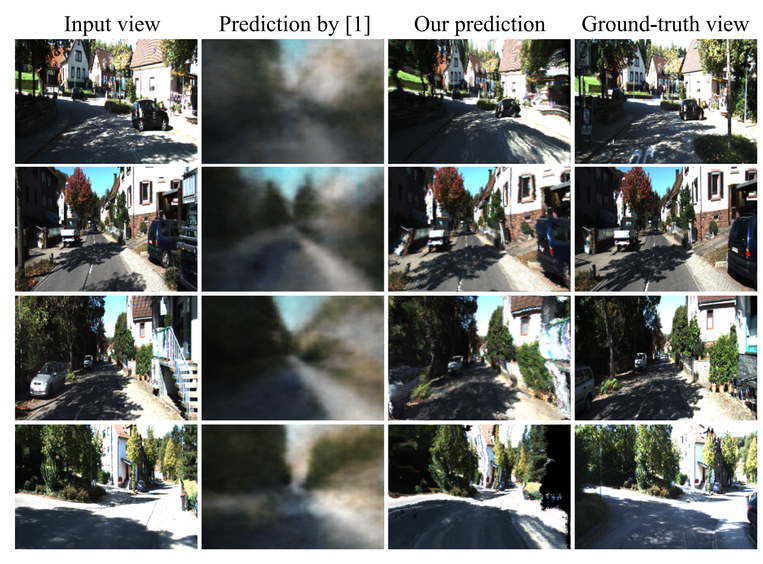
Multiple input views:



KITTI results

Data setup:

- 11 driving sequences through urban scenes (9 for training, 2 for testing)
- Viewpoint transf. = car ego-motion



Source Code

Code will be available at:
`https://github.com/tinghuiz/appearance-flow`