EECS504 Final Report: Optical Flow Estimation using PWC-Net

Brian Purnomo
brianpur@umich.edu

Chengyang Huang
chengyah@umich.edu

Drake Svoboda
drakes@umich.edu

Abstract

In this paper, we implement a smaller variant of PWC-Net [1], a CNN model for optical flow estimation. We also propose a novel improvement to PWC-Net's warping layer. Deep optical flow methods often get stuck at poor local minima early in training [1]. We believe that this is caused by the warping layer adding noise to the features of the second image early in training when the flow predictions are poor. Our modification to the warping layer corrects this and improves training stability. We compare our results to the classical Gunner-Farneback [2] and Pyramidal Lucas-Kanade [3] algorithms. Our implementation out-performs these baselines; however, our simplification of the model as well as reduced training time results in worse performance than PWC-Net's reported results.

1. Introduction

Optical flow estimation is an important computer vision problem. Optical flow arises from the relative motion of objects in a scene and the viewer [4]. Optical flow can reveal important information about the spatial location and velocity of these objects. Some application of optical flow include object segmentation [5], object tracking [6], autonomous vehicle navigation [7], and visual odometry [8]. As optical flow estimation has proven to be an important problem, much research has been done to improve flow estimation accuracy and computational speed.

We re-implement PWC-Net, a recent landmark optical flow model. PWC-Net borrows many principals from classical coarse-to-fine flow methods such as Pyramid Lucas-Kanade [3]. PWC-Net improves upon the performance of classical methods by leveraging the representational power of CNNs.

Our implementation of PWC-Net matches the original except that we use one fewer pyramid levels and we use a novel modification to the warping layer that improves training stability.

2. Related Works

Energy Minimization Approach. Horn and Schunck define flow estimation as an energy minimization problem and solve it using a variational inference approach [4]. Since Horn and Schunck introduced their method, many more energy minimization methods have been proposed [9, 10, 11]. Lucas and Kanade use a differential method that assumes the displacement of the objects between two subsequent frames is small and approximately constant within a neighborhood of the pixel under consideration [12, 13]. Gunnar proposes a method to estimate displacement fields from the polynomial expansion coefficients [14]. A coarse-to-fine scheme and warping-based approach is often adopted to avoid expensive computation [15]. Although energy minimization methods are able to estimate flow precisely, the expensive computational requirements impede their adoption for real-time applications such as self-driving and robotics.

Deep learning Approach. Recently, convolutional neural networks have become popular for estimating optical flow because they can both accurately predict optical flow and reduce solving time—sometimes by a factor of 100—when compared to energy minimization approaches. FlowNet introduces a paradigm shift towards CNNs for flow estimation [16]. Flownet-2 improves upon Flownet by stacking multiple networks to iteratively refine the estimated flow and introducing a differentiable warping operation to compensate for large displacements [17]. SpyNet proposes a spatial pyramid network in order to reduce the model size at the cost of estimation performance [18]. To balance model size and estimation accuracy, PWC-Net combines traditional pyramid processing into a CNN based model [1].

Mixed Approach. The current leading method on the KITTI benchmark is UberATG-DRISF[19], whose flow module is akin to PWC-Net[1]. Ma *et al.* formulate the flow estimation problem as energy minimization in a deep structured model, which can be solved efficiently in the GPU by unrolling a Gaussian-Newton solver. They propose a novel deep rigid instance scene flow (DRISF) model that post-processes the flow estimation and eliminates the drawback of the deep learning based methods. Although it slows down the running time, it reduces the average percentage of outliers by half.

3. Method

Figure 1 shows the complete structure of our implementation of PWC-Net. PWC-Net borrows many principals traditional coarse-to-fine optical flow estimation methods. First, a learnable feature pyramid representation of the two images is constructed using a shared CNN. The model estimates flow at each pyramid level using a sepearate CNN called an optical flow estimator. At each level, the features of the second image are warped using the up-sampled flow prediction from the previous level. After warping, the features of the first image and the warped features of the second image are passed through a cost-volume layer. The optical flow estimator take as input a concatenation of the features of the first image, the cost-volume, and the up-sampled flow prediction from the previous layer. At the lowest level in the pyramid, the flow prediction is considered to be all zero. At the final level, the flow prediction is post-processed by a CNN called the context network.

Our implementation follows the original implementation except for a few subtle changes. In particular, we remove a level from the feature pyramid extractor and modify the warping layer in a novel way to improve stability at the beginning of training.

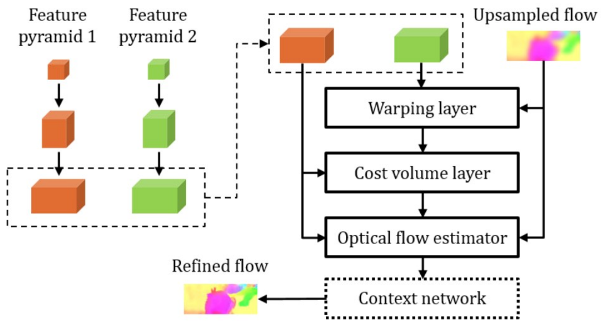


Figure 1: PWC-Net. Image taken from [1]

Feature pyramid. Given two images I_1 and I_2, we generate a L-level feature pyramid using a shared CNN. The bottom most level is the input image. Call c_1^l and c_2^l the features for the two images at the lth level of the pyramid. c_1^l and c_2^l are computed using convolutions filters that down sample the features from the previous level by a factor of 2. The original PWC-net implementation uses 7-Levels with 3, 16, 32, 64, 96, 128 and 196 channels respectively. We omit the final pyramid level and downsample the input image by a factor of 2 using bilinear interpolation. Thus, we have a 6-level pyramid with 3, 16, 32, 64, 96, and 128 channels that takes half-resolution images as input when compared to the original.

Warping layer. At each level l, the features from the second image are warped using the up-sampled flow prediction from the $l+1$th level. Define c_w^l as the warped features of the second image at the lth level

At the very beginning of training, we warp the second image's features using a weighted average of the ground truth flow and the upsampled flow prediction from the l+1th level:

$$c_w^l = c_2^l(x + \frac{\alpha_1 w_{UP}^{l+1}(x) + \alpha_2 w_{GT}^l(x)}{\alpha_1 + \alpha_2}) \tag{1}$$

where w_{UP}^{l+1} is the up-sampled predicted flow from the l+1th level and w_{GT}^l is the ground truth flow down-sampled and scaled to match the dimensions at the lth level. For the first several epochs of training α_1 is set to 0 and α_2 to 1. This warps the features using only the ground truth flow. As training progresses, α_2 is decreased and α_1 is increased until $\alpha_1 = 1$ and $\alpha_2 = 0$. At this point, the features are warped as in the original implementation (without the ground truth flow):

$$c_w^l = c_2^l(x + w_{UP}^{l+1}(x)) \tag{2}$$

We empirically find that this modification to the warping layer improves training stability at the beginning of training. The original authors note that deeper optical flow estimators often get stuck at poor local minima early in training [1]. We believe that this problem is in part caused by the warping layer adding noise to the features of the second image. If the flow prediction from the previous level is not accurate, as it is at the very beginning of training since the model is randomly initialized, the warping operation adds noise to the features of the second image and hurts the model's predictive power. Our novel warping scheme corrects this issue by using the non-noisy ground truth flow at the beginning of training. The model is then weaned off the ground truth flow as each level begins to make better flow predictions.

Cost volume layer. The warped features are used to compute a "cost-volume" by passing the features from the first image and the warped features from second image through a correlation layer:

$$\mathbf{CV}^l(\mathbf{X}_1, \mathbf{X}_2) = \frac{1}{N}\langle c_1^l(\mathbf{X}_1), c_w^l(\mathbf{X}_2)\rangle \tag{3}$$

where $\langle , \rangle$ is the dot product and $N = \dim(c_1^l(\mathbf{X}_1))$.

For computation efficiency, the displacement of the correlation is limited to a 9×9 region. That is, for each $C^l \times 1 \times 1$ vector in the first image's feature map (where C^l

is the number of channels at that pyramid level), the correlation layer computes the dot product of that vector and every $C^l \times 1 \times 1$ vector in a 9×9 region centered at that location in the warped features of the second image. This produces 81 scalars for each position. These scalars are concatenated to produce the $81 \times H^l \times W^l$ cost-volume representation where H^l and W^l is the height and with at the lth pyramid level. The original authors find that the cost-volume representation significantly improves performance [1].

Optical flow estimator. The optical flow estimator for level l is a multi-layer CNN that takes as input the cost-volume, the features from the first image at level l, and the up-sampled optical flow at the the $l+1$th level. Each estimator has 5 convolutional layers with 128, 128, 96, 64 and 32 features. Each estimator makes a $2 \times H^l \times W^l$ flow prediction. The flow prediction is made at $1/20$th the magnitude of the ground truth flow. That is, to retrieve the models final prediction, the output of the optical flow estimator must be scaled by a factor of 20. The original PWC-Net paper evaluates two implementations, one with DenseNet connections in the optical flow estimator and one without [20]. We omit the DenseNet connections in favor of computational efficiency.

Context network. Classical flow methods often post-process flow predictions using contextual information [21, 22]. PWC-net uses a context network to refine the predicted flow at a given pyramid level. The context network is a multi-level CNN that takes as input the predicted flow and the features from the previous level's flow estimator and computes the residual flow used to refine the flow prediction.

The context network is only used for the highest level flow prediction. It makes use of dilated convolutions layers to expand it's receptive field. Our implementation matches the original.

Training loss. PWC-Net makes flow predictions at each pyramid level. The ground truth flow is used as a learning signal at each level. We use the same loss as in the original PWC-Net implementation; however, we omit the $L2$ penalty term. In particular, we use

$$L(\Theta) = \sum_{l=l_0}^{L} \alpha_l \sum_{X} |\mathbf{w}_{\Theta}^l - \mathbf{w}_{GT}^l|_2 \quad (4)$$

where α_l is the weight for level l, $\mathbf{w}_{\Theta}^l$ is the predicted flow and $\mathbf{w}_{GT}^l$ is the ground truth flow down-sampled using bilinear interpolation to match the dimension of the predicted flow at level l. Note that the magnitude of the ground truth is not scaled when down-sampling, thus, each level predicts flow at the same magnitude.

Additionally, we use the following robust loss when fine-tuning:

$$L(\Theta) = \sum_{l=l_0}^{L} \alpha_l \sum_{X} (|\mathbf{w}_{\Theta}^l - \mathbf{w}_{GT}^l|_1 + \epsilon)^q + \gamma|\Theta|_2 \quad (5)$$

4. Experiments

4.1. Data

FlyingChairs The FlyingChairs dataset is a staple training dataset for many optical flow networks [23]. It contains 22872 synthetic training examples created by rendering 3D chair models over 964 Flickr images of various cities, landscapes and mountains. Each training example consists of two images and the corresponding ground-truth optical flow. We select a sub-set of FlyingChairs for training. We prepare the data into pre-defined mini-batches that are loaded into Colab through Google Drive. Using pre-defined mini-batches allows us to overcome an I/O bottleneck in the Google Colab/Drive environment by compressing each minibatch into the H5 file format; however, this reduces the amount of variation the model is able to see during training. We choose a random a crop of 448 by 384 then down-sample by half for each training example. We then down-sample the examples by half before preparing pre-defined mini-batches. When down-sampling, we were careful to account for the change in the magnitude of the ground-truth flow.

KITTI The KITTI optical flow dataset contains 200 training examples obtained by cameras mounted inside of a vehicle while driving [24]. KITTI contains examples with various lighting conditions captured when the car was either stationary or moving. We randomly split the 200 examples into training and test sets of 100 examples each. The ground-truth annotations in KITTI are semi-dense. That is, the ground-truth flow is only provided for a subset of pixels in the scene. Our evaluation only considers pixels where there are ground-truth annotations.

4.2. Training Setup

We randomly initialize our model and train on FlyingChairs before fine-tuning on KITTI.

FlyingChairs We trained the network on the FlyingChairs dataset using a batch size of 8 as suggested in the original paper. The training loss (4) was used with $\alpha_2 = 0.32$, $\alpha_3 = 0.08$, $\alpha_4 = 0.02$, and $\alpha_5 = 0.01$. We use the Adam optimizer with learning rate of $1e-4$. The model was trained on the FlyingChairs dataset for 577 epochs; this took around 2 days.

KITTI While fine tuning our model on KITTI, we used a batch size of 4. At each training step we take a random crop of 320 by 896 and randomly horizontally flip with probability 0.5. When flipping the image, we are also careful to flip the horizontal component of the flow. As in FlyingChairs, we down-sampled the images by half to reduce training time. The training loss (5) was used with $\alpha_2 = 0.32$, $\alpha_3 = 0.08$, $\alpha_4 = 0.02$, $\alpha_5 = 0.01$, $q = 0.4$ and $\gamma = 0.01$. Again we use the Adam optimizer was used with a learning rate of $1e-4$. Since KITTI has semi-dense ground-truth annotation, the training loss is only calculated for pixels where the ground-truth is known.

4.3. Evaluation

Baseline approaches. We select the Gunner-Farneback algorithm implemented by Opencv (Opencv-GF) [14], Pyramid-LK [13], and PWC-Net [1] implemented by Nvidia as baselines to evaluate our model against. Although the performance of Opencv-GF and Pyramid-LK are below the current state-of-the-art, Opencv-GF and Pyramid-LK are very classic and widely used methods for dense optical flow estimation. Because our PWC-Net is simplified, its performance is expected to be below the state-of-the-art, and we believe it is reasonable to compare with these classic methods. The PWC-Net implemented by Nvidia is selected to evaluate the degradation due to our simplifications and limited training time.

Evaluation metrics and methods. There are three major evaluation metrics, Percentage of flow outliers (Fl-all) in non-occluded pixels, Average end-point error (AEPE), and running time. Because the KITTI 2015 does not make ground-truth labels for its test set publicly available, we divide the training set evenly into a 100 example training set and a 100 example test. The reported numbers for Nvidia's implementation of PWC-Net are for the official KITTI test set instead of our custom test set; however, we still believe we can make a meaningful comparison despite this. Table 1 shows a comparison of the baselines and our implementation. Figures 2 and 3 show prediction examples when the camera is stationary and moving.

Performance. Our PWC-Net outperforms Opencv-GF and Pyramid-LK on our KITTI test set. Table 1 shows that our PWC-Net has less average end-point error than Opencv-GF and Pyramid-LK, which suggests that our PWC-Net is able to estimate the flow more accurately than these two baseline methods. The percentage of flow outliers of our PWC-Net is lower than Opencv-GF, but slightly higher than Pyramid-LK. In the aspect of running time, our PWC-Net is 220 times faster than Pyramid-LK and 3 times faster than the Opencv-GF, when dealing with images in the same size. Therefore, our PWC-Net is able to achieve a lower error with improved inference time.

PWC-Net-Nvidia is Nvidia's PWC-Net without fine-tuning on KITTI (with only the pre-training step). Our implementation has less average end-point error than this model mainly because our model fine tunes on KITTI 2015 data set. We also predict flow at half the resolution leading to relatively denser ground-truth annotations and smaller average error. However, the fine-tuned PWC-Net by Nividia (PWC-Net-Nvidia-ft) has a much lower average error. The percentage of flow outliers of our PWC-Net is higher than both models, which indicates that our PWC-Net is still unable to track the moving object precisely. The inference time is not comparable because the input image size and the hardware are different. Our results suggest that further training is necessary to improve accuracy and simplifying PWC-Net will degrade performance.

Table 1: Results on evaluation set separated from KITTI 2015 training set

Methods	AEPE	Fl-all	Time (s)
Opencv-GF	6.84	55.2%	0.03
Pyramid-LK	5.93	48.1%	2.20
PWC-Net	5.74	50.8%	0.01
PWC-Net-Nvidia [1]	10.35	33.7%	0.03
PWC-Net-Nvidia-ft [1]	2.16	6.12%	0.03

Figure 2 shows that, when the camera is stationary, the flow estimation from our PWC-Net is more clear and less noisy. It captures all three cars in the scene accurately, although it predicts the speed of the background incorrectly. When the camera is moving in figure 3, it can be seen that the Opencv-GF and Pyramid-LK do not have clear flow estimation contour, while the flow estimation from our PWC-Net is more close to the ground truth label. It is more challenging for the traditional method when the camera is moving because they rely on the assumption that the movement between two frames should be small.

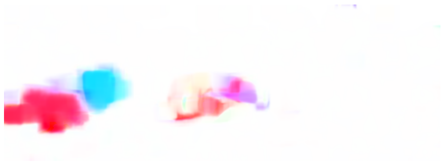
5. Conclusions

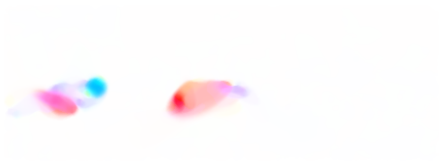
PWC-Net made a substantial contribution in using CNNs to estimate optical flow. In our version of PWC-Net, we reduce the size of the model by removing a pyramid level and cutting the dense connections in the estimator layer. In addition, we make a novel improvement to the warping layer that prevents the model from getting stuck at poor local minima early in the training. Though we were not able to train our network as long as the original PWC-Net, the results seems to be satisfactory as we have a lower average endpoint error than both the Pyramidal Lucas Kanade and Gunner-Farneback algorithms. One weakness of our

(a) First frame

(b) Second frame

(c) Ground truth


(d) Opencv-GF


(e) Pyramid-LK

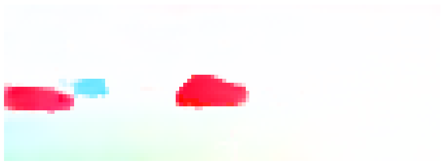
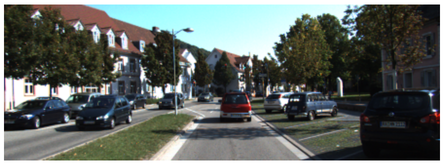
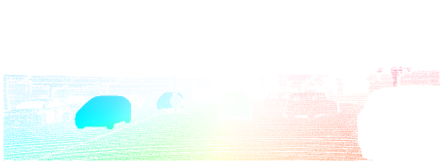
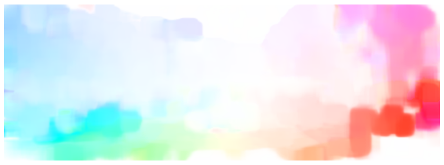

(f) PWC-Net

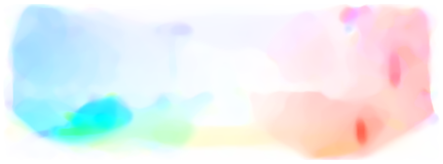
Figure 2: Visual results on evaluation set separated from KITTI 2015 training set using Opencv-GF, Pyramid-LK, PWC-Net, when the camera is stationary


(a) First frame

(b) Second frame


(c) Ground truth


(d) Opencv-GF


(e) Pyramid-LK

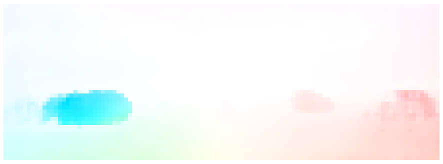

(f) PWC-Net

Figure 3: Visual results on evaluation set separated from KITTI 2015 training set using Opencv-GF, Pyramid-LK, PWC-Net, when the camera is moving

network was that the results has a slightly higher percentage of flow outliers on non-occluded pixels of 2.7% compared to the Lucas Kanade baseline. Though, this seems to be the cost of simplifying the network and not having enough training. We believe that by more closely matching the training specification in the original paper our performance could be significantly improved. We would also like to test our model on the the Sintel optical flow dataset [25] which is another popular optical flow dataset.

References

[1] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," *arXiv preprint arXiv:1709.02371*, 2017.

[2] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," vol. 2749, pp. 363–370, 06 2003.

[3] J.-Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm," 2000.

[4] B. G. S. Berthold K.P. Horn, "Determining optical flow," *Artificial Intelligence*, 1981.

[5] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang, "Segflow: Joint learning for video object segmentation and optical flow," in *Proceedings of the IEEE international conference on computer vision*, pp. 686–695, 2017.

[6] J. Shin, S. Kim, S. Kang, S.-W. Lee, J. Paik, B. Abidi, and M. Abidi, "Optical flow-based real-time object tracking using non-prior training active feature model," *Real-Time Imaging*, vol. 11, no. 3, pp. 204–218, 2005.

[7] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE con-*

ference on computer vision and pattern recognition, pp. 3061–3070, 2015.

[8] J. Campbell, R. Sukthankar, I. Nourbakhsh, and A. Pahwa, "A robust visual odometry and precipice detection system using consumer-grade monocular vision," in *Proceedings of the 2005 IEEE International Conference on robotics and automation*, pp. 3421–3427, IEEE, 2005.

[9] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 75 – 104, 1996.

[10] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *European conference on computer vision*, pp. 25–36, Springer, 2004.

[11] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert, "Highly accurate optic flow computation with theoretically justified warping," *International Journal of Computer Vision*, vol. 67, pp. 141–158, 04 2006.

[12] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," vol. 81, pp. 121–130, 04 1981.

[13] J.-Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker," 1999.

[14] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis* (J. Bigun and T. Gustavsson, eds.), (Berlin, Heidelberg), pp. 363–370, Springer Berlin Heidelberg, 2003.

[15] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/kanade meets horn/schunck: Combining local and global optic flow methods," *International journal of computer vision*, vol. 61, no. 3, pp. 211–231, 2005.

[16] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," *CoRR*, vol. abs/1504.06852, 2015.

[17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[18] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," 2016.

[19] W.-C. Ma, S. Wang, R. Hu, Y. Xiong, and R. Urtasun, "Deep rigid instance scene flow," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[20] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016.

[21] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers, *An Improved Algorithm for TV-L1 Optical Flow*, pp. 23–45. 07 2009.

[22] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi, "Bilateral filtering-based optical flow estimation with occlusion detection," pp. 211–224, 05 2006.

[23] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[24] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

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