EE209AS
Final Project

1/28/2019

- Option_1: Traditional CV based algorithm acceleration for license plate detection and recognition
- Option_2: Deep learning based algorithm acceleration for license plate detection and recognition
- Option_3: Acceleration for image registration operations

License Plate Detection and Recognition

- Background
  - Surveillance video can be useful in a lot of scenarios
  - A detection + segmentation + classification problem
  - Recognize license plate from still video images of CCTV

License Plate Detection and Recognition

- Dataset:
  - Application oriented license plate (AOLP) Databased by NTUST
    - http://aolpr.ntust.edu.tw/lab/index.html
    - https://drive.google.com/open?id=1t2zesc4jJDkhj6IF6SeyR6iMPTHTPxfn
  - Supplement dataset:
    - License plate recognition dataset
    - https://drive.google.com/open?id=1Na_bT2pKYdEiv0UvBPwzG0_zmi8wwCNd

License Plate Detection and Recognition

Access control camera

Traffic law enforcement camera

Road patrol camera

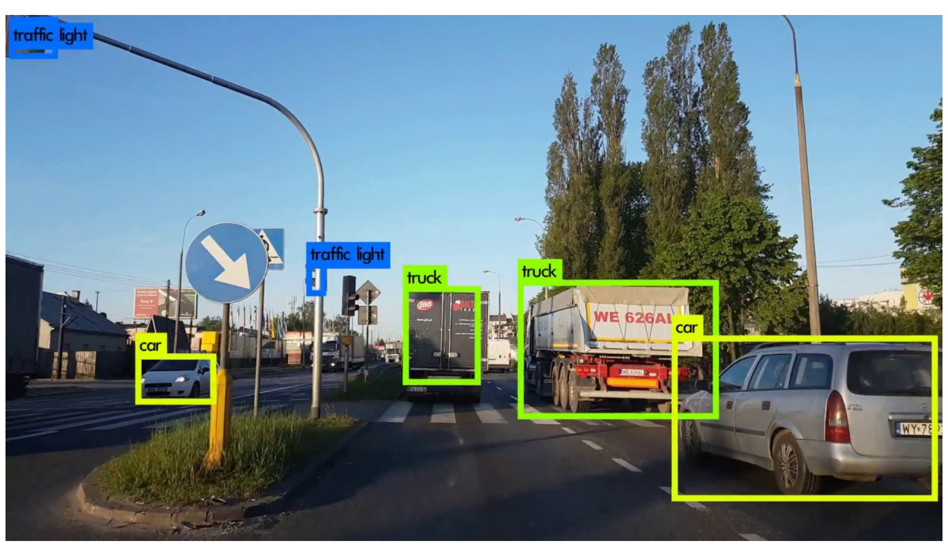

```
{
"annotation":{
"folder":"AOLP_LE_test",
"filename":"1.jpg",
"source":{
"database":"AOLP",
"annotation":"LE_test",
"image":"AOLP"
},
"size":{
"width":"640",
"height":"480",
"depth":"3"
},
"segmented":"1",
"object":{
"name":"plate",
"platetext":"3968XJ",
"pose":"Frontal",
"truncated":"1",
"difficult":"0",
"bndbox":{
"xmin":"173",
"ymin":"200",
"xmax":"316",
"ymax":"275"
}
}
}
}
```

License Plate Detection and Recognition

- Dataset:
    - Application oriented license plate (AOLP):
        - AC: 100 training + 581 testing
        - LE: 100 training + 687 testing
        - RP: 100 training + 511 testing
    - License Plate dataset:
        - 3922 images with labels
        - Not necessary to be used

License Plate Detection and Recognition

- Past work
  - 1. The recognition is often done in steps. And for each step, a (deep) model is trained.
  - 2. The model can either be of traditional CV methods, or deep learning methods.
  - **References see the next page**



Step 1: Car detection

Step 2:
License plate cropping

Step 3:
number recognition

License Plate Detection and Recognition

- **Tradition methods for license plate recognition**

K. K. Kim, K. I. Kim, J. B. Kim and H. J. Kim, "Learning-based approach for license plate recognition," Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No.00TH8501), Sydney, NSW, Australia, 2000, pp. 614-623 vol.2.

Zehang Sun, G. Bebis and R. Miller, "On-road vehicle detection using Gabor filters and support vector machines," 2002 14th International Conference on Digital Signal Processing Proceedings. DSP 2002 (Cat. No.02TH8628), Santorini, Greece, 2002, pp. 1019-1022 vol.2.

Wei Zheng and Luhong Liang, "Fast car detection using image strip features," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, 2009, pp. 2703-2710.

C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos and E. Kayafas, "A License Plate-Recognition Algorithm for Intelligent Transportation System Applications," in IEEE Transactions on Intelligent Transportation Systems, vol. 7, no. 3, pp. 377-392, Sept. 2006.

Anagnostopoulos, C., Anagnostopoulos, I., Psoroulas, I.D., Loumos, V., & Kayafas, E. (2008). License Plate Recognition From Still Images and Video Sequences: A Survey. *IEEE Trans. Intelligent Transportation Systems, 9*, 377-391.

- **Deep learning methods for license plate recognition**

Li, H., & Shen, C. (2016). Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs. *CoRR, abs/1601.05610*.

Yamwong, P., Hou, S., Wang, Z., & Zha, Z. (2018). Towards Human-Level License Plate Recognition. *ECCV*.

Xie, L., Ahmad, T., Jin, L., Liu, Y., & Zhang, S.X. (2018). A New CNN-Based Method for Multi-Directional Car License Plate Detection. *IEEE Transactions on Intelligent Transportation Systems, 19*, 507-517.

Laroca, R., Severo, E., Zanlorensi, L.A., Oliveira, L.E., Gonçalves, G.R., Schwartz, W.R., & Menotti, D. (2018). A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector. *2018 International Joint Conference on Neural Networks (IJCNN)*, 1-10.

Option 1: Traditional CV based algorithm acceleration for license plate detection and recognition

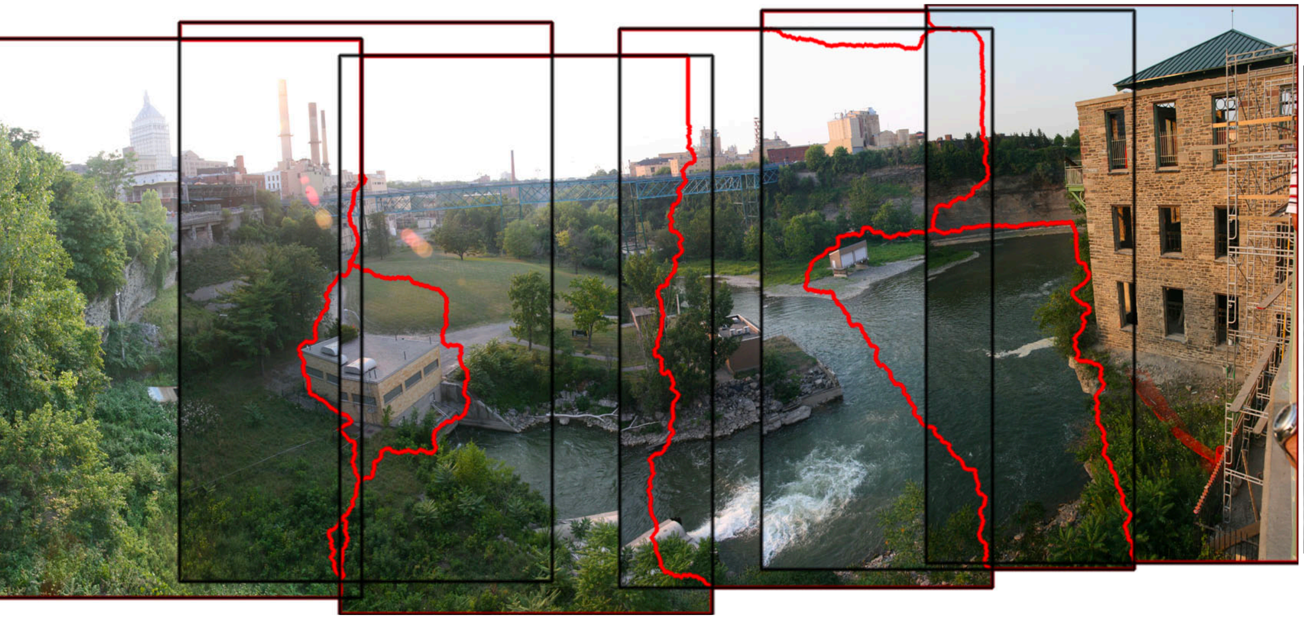
- Task
  - 1. Review traditional CV based methods for license plate recognitions.
  - 2. Build the framework for license plate recognition. The models can be self implemented or from references.
  - 3. Analyze framework and profiles the steps.
  - 4. Select one step to accelerate by using FPGA.
  - 5. Run simulations in Verilog and analyze the acceleration.

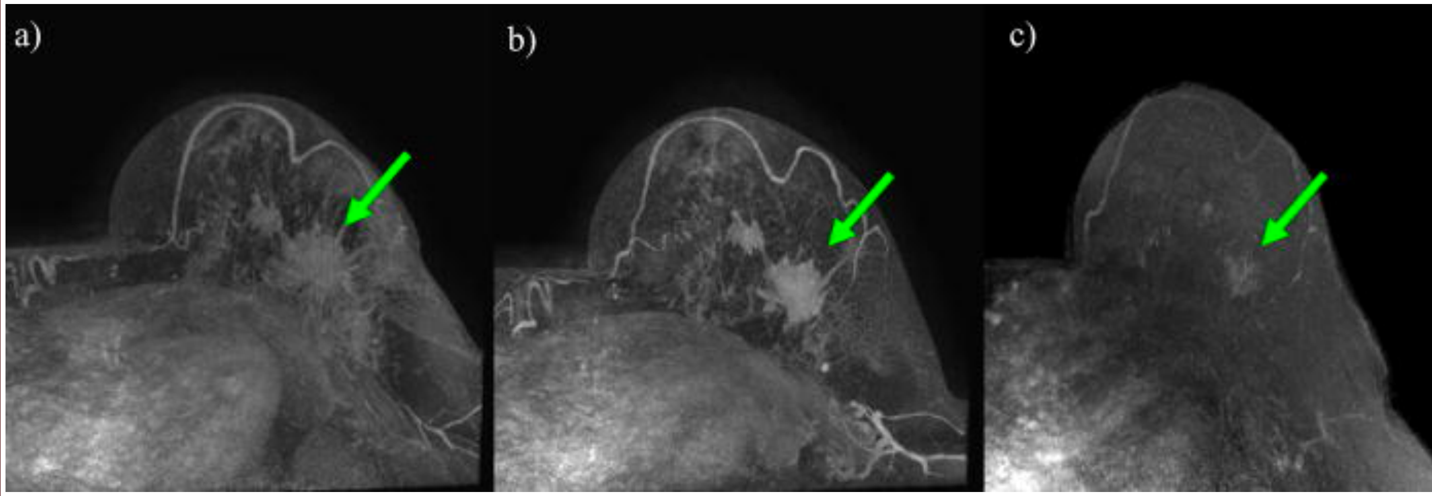
Option 2: Deep learning based algorithm acceleration for license plate detection and recognition

- Task
  - 1. Review deep learning based methods for license plate recognitions.
  - 2. Build the framework for license plate recognition. The models can be self implemented/trained or from references/trained models.
  - 3. Analyze framework and profiles the steps.
  - 4. Select one network to accelerate by network pruning and FPGA.
  - 5. Run simulations in Verilog and analyze the acceleration.

Image Registration

- Background
  - Finding a spatial one-to-one mapping from pixels in one image to pixels in the other image
  - Useful in a lot of scenarios

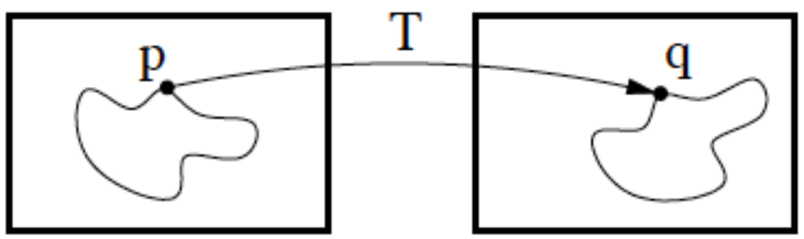




By Noso1 (talk) - Own work (Original text: I created this work entirely by myself.), Attribution, https://commons.wikimedia.org/w/index.php curid=12084707

Li X, Dawant BM, Welch EB, et al. A nonrigid registration algorithm for longitudinal breast MR images and the analysis of breast tumor response. *Magn Reson Imaging*. 2009;27(9):1258-70.

Image Registration



Moving Image $I_M(x)$ $\Omega_M \subset R^d$

Fixed Image $I_F(x)$ $\Omega_F \subset R^d$

Finding $T_u : \Omega_F \to \Omega_M$, **such that** $I_M(T_u(x))$ **aligns with** $I_F(x)$

Solving $\hat{u} = \arg\min_u C(u; I_F; I_M)$

s.t. $C(u; I_F, I_M) = -S(u; I_F, I_M) + \gamma P(u)$

	$T_u : \Omega_F \to \Omega_M$	Formulation	# of parameters
Rigid models	**Translation**	$T_u(x) = x + t$	**3**
	Rigid	$T_u(x) = R(x - c) + t + c$	**9**
	Affine	$T_u(x) = A(x - c) + t + c$	**12**
Non-Rigid (deformable) models	**B-splines**	$T_u(x) = x + \sum_{x_k \in N} p_k\left(\frac{x - x_k}{\sigma}\right)$	**N*P**
	Thin-plate splines	$T_u(x) = x + Ax + t + \sum_{x_k \in N} c_k G(x - x_k)$	**N*C**
Non-parametric models	**Non-parametric models**	......	

Image Registration

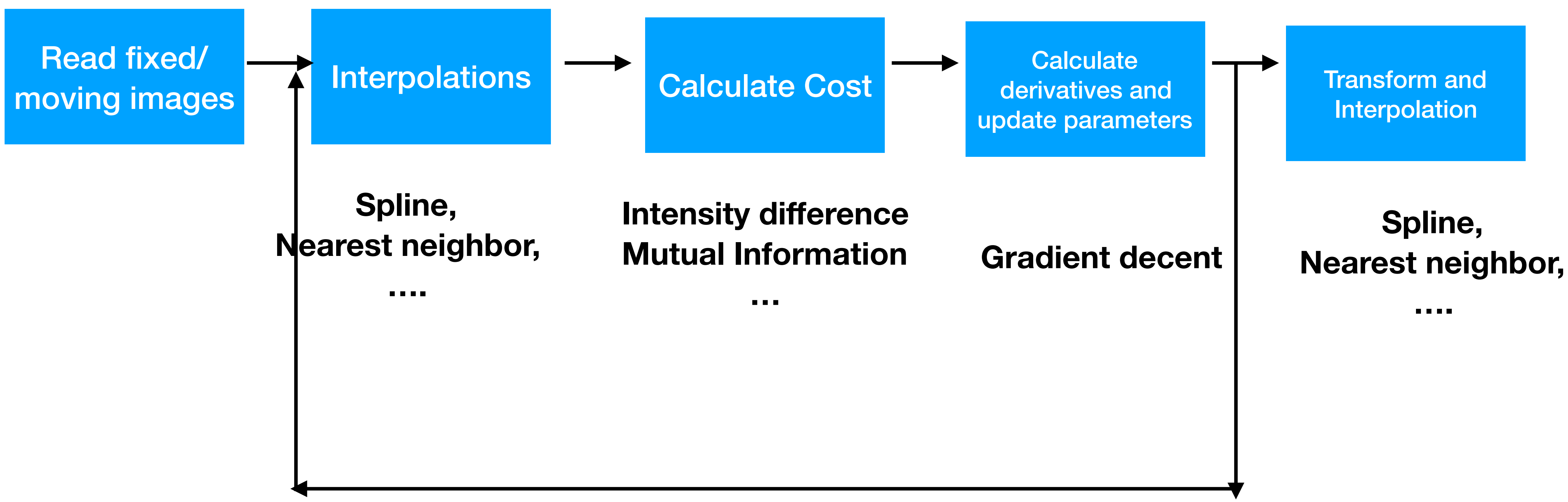
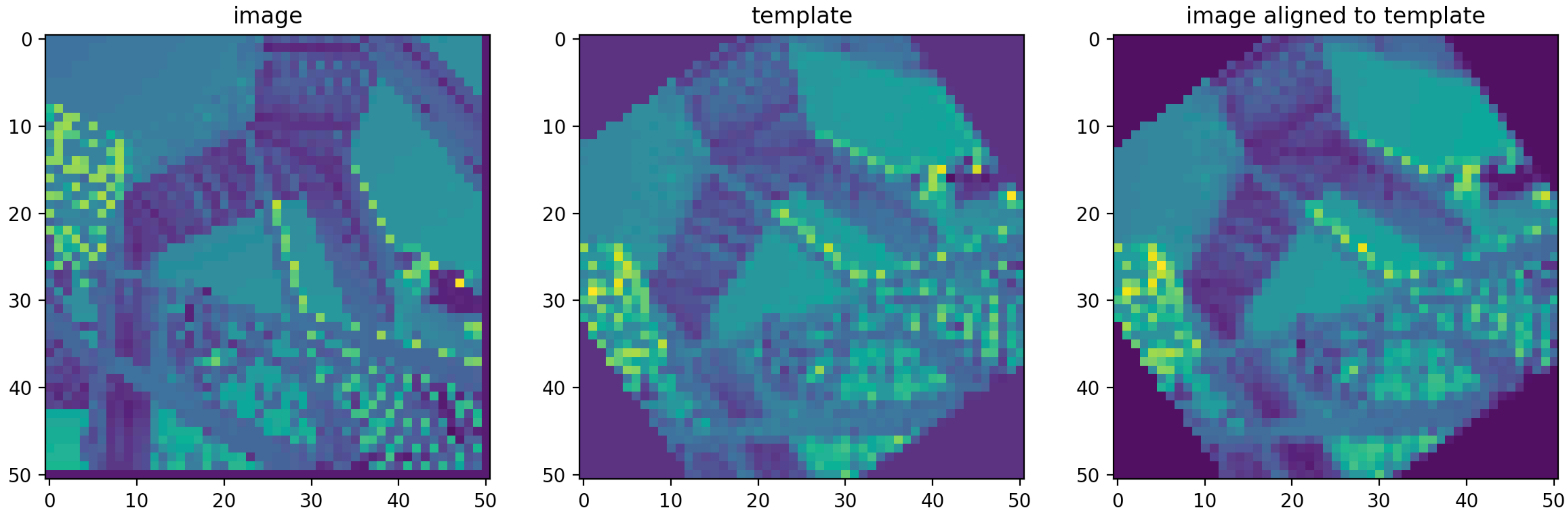


Image Registration

- Source code in Python
  - https://github.com/pyimreg/imreg
  - Modified and uploaded on google drive:
    - https://drive.google.com/file/d/1jECiAbWrnxTPVhIousIoYQvbnOx2YxvW/view?usp=sharing
  - We will work with rigid registration, bilinear interpolation, and intensity difference as cost function. See code for more information.
  - To learn more about the code, see references we recommend as below:
    - Code implementation: S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition, Volume 1, Pages 1090 – 1097, December 2001.
    - General registration: https://www.creatis.insa-lyon.fr/~srit/tete/2012_master_eeap_si_m5.pdf

Image Registration



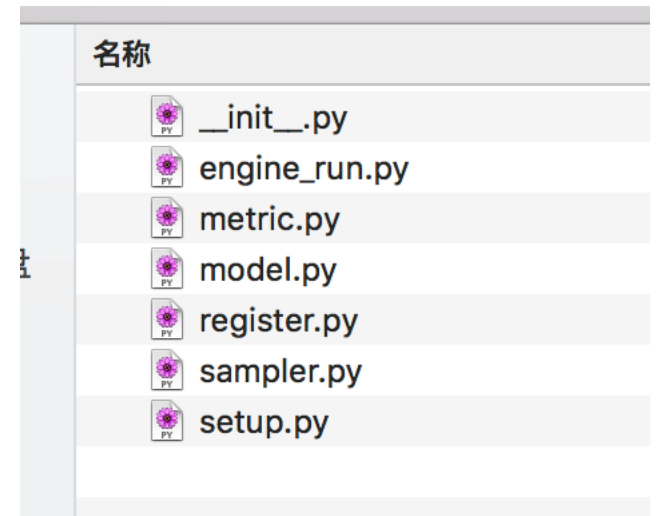
$$T_u(x) = A(x-c) + t + c$$

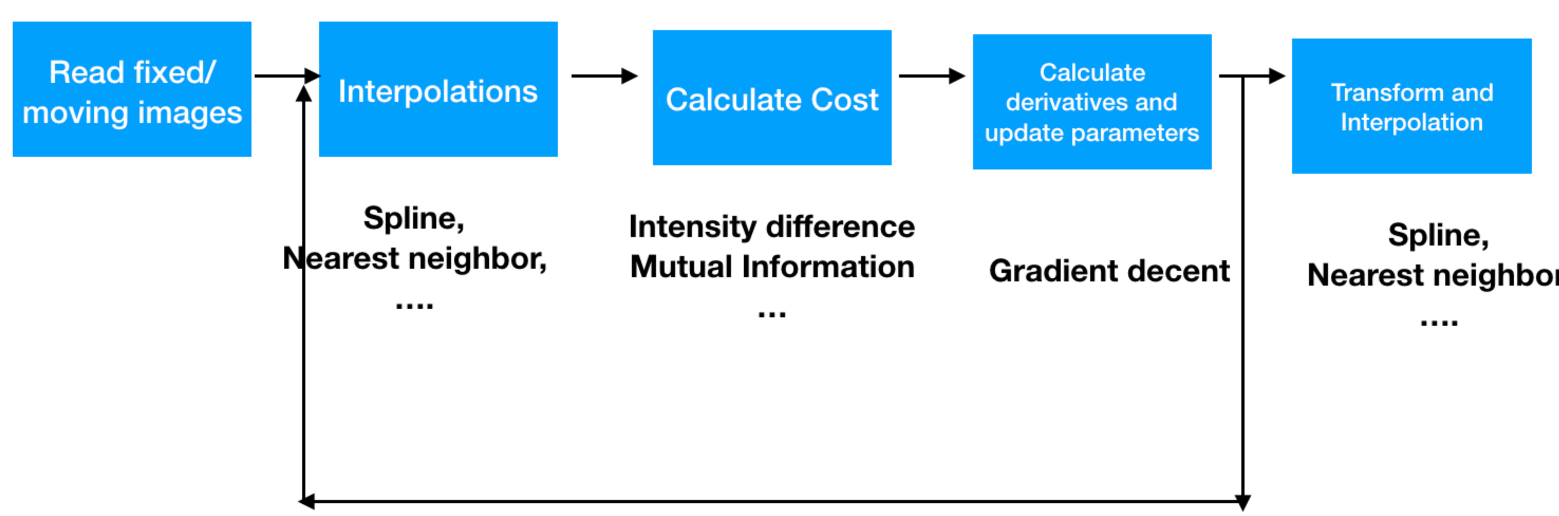
6 parameters in total for rotation, scaling, and transportation.

```
      1756909 function calls (1748951 primitive calls) in 4.036 seconds

Ordered by: cumulative time

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     1    0.011    0.011    4.038    4.038 engine_run.py:1(<module>)
     1    0.000    0.000    2.137    2.137 pyplot.py:236(show)
     1    0.000    0.000    2.137    2.137 backend_bases.py:178(show)
     1    0.000    0.000    1.483    1.483 backend_macosx.py:208(mainloop)
     1    1.482    1.482    1.483    1.483 {matplotlib.backends._macosx.show}
     1    0.210    0.210    0.654    0.654 {method 'show' of '_macosx.FigureManager' objects}
     1    0.001    0.001    0.530    0.530 common.py:123(ascent)
     1    0.000    0.000    0.518    0.518 pickle.py:1383(load)
     1    0.117    0.117    0.518    0.518 pickle.py:851(load)
     1    0.037    0.037    0.464    0.464 __init__.py:101(<module>)
     2    0.000    0.000    0.442    0.221 backend_macosx.py:79(_draw)
 340/2    0.003    0.000    0.416    0.208 artist.py:47(draw_wrapper)
     2    0.002    0.001    0.416    0.208 figure.py:1438(draw)
   8/2    0.000    0.000    0.408    0.204 image.py:123(_draw_list_compositing_images)
     6    0.000    0.000    0.407    0.068 _base.py:2528(draw)
     1    0.008    0.008    0.400    0.400 pyplot.py:19(<module>)
    15    0.066    0.004    0.335    0.022 __init__.py:1(<module>)
     1    0.002    0.002    0.309    0.309 register.py:1(<module>)
     1    0.007    0.007    0.306    0.306 metric.py:1(<module>)
     1    0.004    0.004    0.286    0.286 __init__.py:127(<module>)
     2    0.010    0.005    0.274    0.137 __init__.py:57(<module>)
     1    0.005    0.005    0.273    0.273 filters.py:31(<module>)
262144    0.157    0.000    0.267    0.000 pickle.py:935(load_binint1)
     1    0.002    0.002    0.263    0.263 _ni_docstrings.py:1(<module>)
     1    0.005    0.005    0.255    0.255 __init__.py:172(<module>)
     1    0.022    0.022    0.250    0.250 interpolate.py:2(<module>)
    12    0.000    0.000    0.209    0.017 axis.py:1182(draw)
     3    0.017    0.006    0.201    0.067 __init__.py:7(<module>)
     6    0.000    0.000    0.167    0.028 image.py:569(draw)
     1    0.013    0.013    0.166    0.166 __init__.py:106(<module>)
526858    0.163    0.000    0.163    0.000 {method 'read' of 'file' objects}
     6    0.004    0.001    0.160    0.027 image.py
     6    0.015    0.003    0.155    0.026 image.py
   295    0.003    0.000    0.153    0.001 {__impo
     1    0.008    0.008    0.148    0.148 colorbar
     1    0.003    0.003    0.119    0.119 add_newc
    24    0.001    0.000    0.109    0.005 axis.py:
   216    0.001    0.000    0.094    0.000 axis.py:
    78    0.002    0.000    0.093    0.001 axis.py:
     1    0.000    0.000    0.093    0.093 pyplot.p
    13    0.000    0.000    0.089    0.007 six.py:9
```





Option 3: Acceleration for image registration operations

- Task
  - 1. Review the code we release, and understand how it works.
  - 2. Select one function to accelerate by FPGA.
  - 3. Run simulations in Verilog and analyze the acceleration.

Grade Metric

- Algorithm accuracy run with PC: 30%
- FPGA acceleration design (and network pruning): 40%
- Acceleration results with simulations: 30%