Cross-Domain Synthesis of Medical Images Using Efficient Location-Sensitive Deep Network

Conference Paper · October 2015

DOI: 10.1007/978-3-319-24553-9_83

3 authors, including:

S. Kevin Zhou
Siemens
391 PUBLICATIONS **4,530** CITATIONS
SEE PROFILE

Raviteja Vemulapalli
Google, Seattle, United States
17 PUBLICATIONS **892** CITATIONS
SEE PROFILE

Cross-Domain Synthesis of Medical Images Using Efficient Location-Sensitive Deep Network

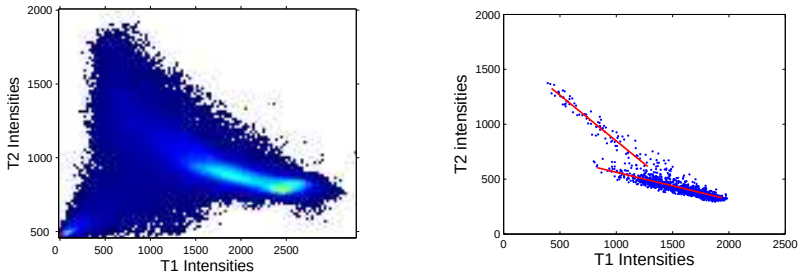
Abstract. Cross-modality image synthesis has recently gained significant interest in the medical imaging community. In this paper, we propose a novel architecture called location-sensitive deep network (LSDN) for synthesizing images across domains. Our network integrates intensity feature from image voxels and spatial information in a principled manner. Specifically, LSDN models hidden nodes as products of features and spatial responses. We then propose a novel method, called ShrinkConnect, for reducing the computations of LSDN without sacrificing synthesis accuracy. ShrinkConnect enforces simultaneous sparsity to find a compact set of functions that well approximates the responses of all hidden nodes. Experimental results demonstrate that LSDN+ShrinkConnect outperforms the state of the art in cross-domain synthesis of MRI brain scans by a significant margin. Our approach is also computationally efficient, e.g. $26\times$ faster than other sparse representation based methods.

1 Introduction

Cross-modality synthesis of medical images is important for many applications such as atlas construction [4], virtual enhancement [10], multimodal registration [3, 11, 12, 17] and segmentation [7, 12, 13]. Various physics-based and data-driven approaches have been proposed to improve the accuracy of this task. For example, a model-based approach for generating ultrasound scans from CT scans was proposed in [17]. In [5], an approach was proposed for explicitly estimating the intrinsic tissue parameters (that cause MR contrast) using a set of MRI scans. Both these approaches rely on the underlying physics of the data acquisition process and cannot be applied to other modalities.

A simple modality transformation between T1-MRI and T2-MRI was proposed in [9] for image registration. A supervised regression-based synthesis approach using decision forest was proposed in [1, 8]. Recently, coupled sparse representation was used in [3]. Similar sparse coding-based approaches have also been used in [12, 14, 16] for image synthesis applications. A supervised approach was proposed in [13] for MR contrast synthesis using a codebook of paired training patches. Similar approaches inspired by image analogies [6] have also been used in [7]. An iterative synthesis approach, called modality propagation (MP), was proposed in [18].

Spatial information is important for accurate image synthesis. However, existing approaches either do not use spatial information or use it as a hard constraint. For example, modality propagation [18] restricts nearest neighbor search to a small window around the voxel's location. In order to see the importance of incorporating the spatial information, we plot the intensity correspondences of registered MRI-T1 and MRI-T2 of the same subject in Fig. 1a. We can notice that the intensity transformation is not only non-linear but also far from unique, i.e. there are multiple feasible intensity values in



(a) (b)

Fig. 1: a) Intensity correspondences between T1 and T2 scans over an entire image. Brighter color indicates higher density regions. b) Intensity correspondences of a restricted region of $10 \times 10 \times 10$ voxels. Red lines indicate the main directions of variation. All images are registered using rigid transformations.

MRI-T2 domain for one intensity value in MRI-T1 domain. The non-uniqueness comes from a well-known fact that intensity values depend on the regions in which voxels reside. By restricting to a local neighborhood of, say $10 \times 10 \times 10$ voxels, the intensity transformation is much simpler as shown in Fig. 1b. In particular, it could be reasonably well described as a union of two linear subspaces represented by the two red lines. That is to say, the spatial information helps simplify the relations between modalites which in turn could enable more accurate prediction. Unfortunately, existing approaches do not have a systematic way to incorporate this type of information.

In this paper, we propose a novel architecture called location-sensitive deep network (LSDN) to integrate image intensity features and spatial information in a principled manner. Our network models the responses of hidden nodes as the product of feature responses and spatial responses. In LSDN formulation, spatial information is used as soft constraints whose parameters are learned. As a result, LSDN is able to capture the joint distribution of feature and spatial information. We also propose a network simplification method for speeding up LSDN prediction. Experimental results demonstrate that our approach achieves better synthesis quality compared to the state-of-the-art. It is also more computationally efficient because the algorithm only uses feed-forward operations instead of expensive sparse coding or nearest neighbor search.

Contributions: 1) We incorporate spatial location and image intensity feature for cross-domain image synthesis in a principled manner. To perform such an integration, we propose a novel deep network architecture called location-sensitive deep network. We derive the gradients necessary for training LSDN. 2) We propose a technique based on simultaneous sparsity for simplifying LSDN. 3) We provide experiments to demonstrate that LSDN outperforms state-of-art methods on brain MRI synthesis.

2 Location-Sensitive Deep Network

Our goal is to learn a deep network that uses an image of one domain (e.g., MRI-T1) to predict the corresponding image from another domain (e.g., MRI-T2). It is ineffective to train a network that operates on the entire image since the number of variables becomes too large for the learning algorithm to generalize well. Instead, our network operates on

small voxels. Let $\mathbf{s}$ and $\mathbf{x}$ denote the voxel's intensity feature and spatial coordinates from the input domain, respectively. Let $\psi(.)$ represent a mapping that is carried out by a multi-layer network. This function operates on $(\mathbf{s}, \mathbf{x})$ and gives out a scalar value approximating the corresponding intensity t in the output domain. The error function that we want to minimize can be written as:

$$E = \frac{1}{2N} \sum_{n=1}^{N} \|\psi(\mathbf{s}_n, \mathbf{x}_n) - t_n\|^2 \tag{1}$$

where N is the total number of voxels sampled from all training images. The minimization is with respect to network variables which will be explained in detail shortly. For the simplicity of notation, the subscript "n" would be omitted in our derivations. Motivated by the observation in Fig. 1, which shows that output intensity depends on voxel's location, we make our mapping dependent on both local feature and spatial coordinates. One straight forward way to do this is by concatenating $\mathbf{s}$ and $\mathbf{x}$ into a single vector. However, our experiments show that this strategy is not effective, possibly because the image feature and the spatial location are two heterogeneous sources of information.

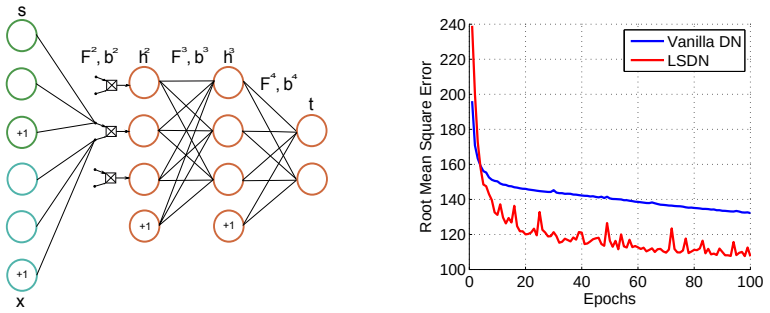
We introduce a location-sensitive deep network for effectively fusing image feature and spatial information in a principled manner. Fig. 2a shows the architecture of a LSDN, where $(\mathbf{F}^k, \mathbf{h}^k, \mathbf{b}^k)$ are the set of filters, hidden nodes, and biases at k-th layer, respectively. The network contains multiple feed-forward layers. In addition, the hidden nodes in the second layer of LSDN is modeled as products of feature and spatial functions:

$$\mathbf{h}^2 = \kappa(\mathbf{s}) \odot \varsigma(\mathbf{x}), \quad \kappa(\mathbf{s}) = \gamma(\mathbf{u}^2), \quad \mathbf{u}^2 = \mathbf{F}^2\mathbf{s} + \mathbf{b}^2 \tag{2}$$

$$\varsigma(\mathbf{x}) = 2 \times sigm\left(-\left[\frac{\|\mathbf{x} - \hat{\mathbf{x}}_1\|^2}{\sigma^2}, \ldots, \frac{\|\mathbf{x} - \hat{\mathbf{x}}_{p_2}\|^2}{\sigma^2}\right]^T\right) \tag{3}$$

Here, $\kappa(\mathbf{s})$ is a feature response computed by a linear filtering followed by a sigmoid function denoted as $\gamma(.)$. The spatial response function $\varsigma(\mathbf{x})$ is parameterized by $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_{p_2}]$, which are learned, and a constant σ. we use "$\odot$" to indicate the Hadamard product, and $sigm$ to denote the *sigmoid* function. The reason we choose ς as in (3) is because we want to enforce locality property within the network. Specifically, we associate each hidden node in the second layer with a latent coordinates $\hat{\mathbf{x}}_i$. As can be seen from (3), i-th hidden node of the second layer only turns on when the voxel is close enough to location $\hat{\mathbf{x}}_i$. The combination of on/off hidden nodes effectively creates multiple sub-networks, each tuned to a small spatial region in the image. This novel property is an important advantage of our network compared to other approaches. We recall from the observation in Fig. 1b that the input-output mapping becomes much simpler when restricted to a smaller spatial region. Therefore, LSDN has potential to yield more accurate prediction. Our experimental results in section 4 confirm this intuition.

We note that, for spatial coordinates $\mathbf{x}$ to convey useful information, training and test images are registered to a reference image using rigid transformations. This makes the same location in different images corresponding to roughly the same anatomical region. Alternatively, one could eliminate the need for registration by using relative coordinates with respect to some landmarks. This direction is open for future research.



(a) (b)

Fig. 2: a) Location-sensitive deep network. Green color and cyan color indicate feature $\mathbf{s}$ and spatial location $\mathbf{x}$, respectively. For better clarity, we only draw connections between input layer and one product node. b) The evolution of training errors over 100 epochs.

2.1 Training LSDN

We use stochastic gradient descent [2] to optimize the loss function in (1). The optimization is with respect to the network's parameters such as filters' coefficients, biases, and latent coordinates. Recall that the second-layer hidden nodes' responses are given in (2). We can write the responses of hidden nodes in higher layers as:

$$\mathbf{h}^k = \gamma(\mathbf{u}^k), \text{ where } \mathbf{u}^k = \mathbf{F}^k\, \mathbf{h}^{k-1} + \mathbf{b}^k, \ \forall k \in [3, K] \tag{4}$$

where K is the number of layers. First, the gradients of the bias terms $\mathbf{b}^k$, denoted as $\mathbf{d}^k$, can be computed recursively as in (5), where $\gamma'(.)$ denotes the derivative of the sigmoid function. This recursive relationship can be verified easily using the chain rule.

$$\mathbf{d}^k = (\mathbf{F}^{k+1})^T \mathbf{d}^{k+1} \odot \gamma'(\mathbf{u}^k), \ \text{ where } \ \mathbf{d}^k = \frac{\partial E}{\partial \mathbf{b}^k}, \ \forall k \in [3, K-1]. \tag{5}$$

The above expression only applies to the intermediate layers. The gradients of biases in the second layer and the last layer take slightly different forms:

$$\mathbf{d}^2 = {\mathbf{F}^3}^T\, \mathbf{d}^3 \odot \gamma'(\mathbf{u}^2) \odot \varsigma(\mathbf{x}), \ \text{ and } \ \mathbf{d}^K = (\psi(\mathbf{s}, \mathbf{x}) - t) \odot \gamma'(\mathbf{u}^K) \tag{6}$$

Once all $\mathbf{d}^k$ and $\mathbf{h}^k$ are computed, the other gradients could be easily derived from using the chain rule. For completion, the gradients of filters coefficients and latent coordinates are provided below. We use $[.]_i$ to denote the i-th element of a vector.

$$\frac{\partial E}{\partial \mathbf{F}^k} = \mathbf{d}^k\, {\mathbf{h}^{k-1}}^T, \ \forall k \in [2, K] \tag{7}$$

$$\frac{\partial E}{\partial \hat{\mathbf{x}}_i} = \frac{4}{\sigma^2} \left[{\mathbf{F}^3}^T\, \mathbf{d}^3 \odot \gamma(\mathbf{u}^2) \odot \varsigma(\mathbf{x}) \odot (1 - \varsigma(\mathbf{x})) \right]_i \times (\hat{\mathbf{x}}_i - \mathbf{x}) \tag{8}$$

The learning rate is one of the most important tuning parameters. We empirically found that 0.25 is a good learning rate for our experiments. We also slowly decrease the learning rate after each iteration by multiplying it by 0.99. Fig. 2b shows the evolution of

training error over 100 epochs which we obtained when training a LSDN to predict MRI-T2 intensity values from MRI-T1 intensity values. More details of this experiment will be explained in section 4. We can see that LSDN error goes significantly lower than that of the vanilla network despite they have the same parameter setting such as learning rate and number of hidden nodes. Similar patterns were observed for different learning rates and network sizes.

3 Network Simplification

Applying LSDN on every voxel during the synthesis process can be computationally expensive because medical images usually contains hundreds of thousands of voxels. In what follows, we propose a post-processing approach for simplifying the network in order to improve the speed of LSDN without losing much in its prediction accuracy. Our method is based on a central observation that, at each hidden layer of a network, there exists a smaller subset of functions that approximately span the same functional space of the entire layer. Let $\mathcal{I}^k$ denote the index set of such subset, we have:

$$h^k_{i,n} \approx \sum_{j \in \mathcal{I}^k} \alpha^k_{ij} h^k_{j,n}, \ \forall i \in [1, p_k], \ \forall n \in [1, N] \tag{9}$$

where p_k is the number of hidden nodes at k-th layer, $h^k_{i,n}$ is the response of i-th hidden node at k-layer for n-th training sample, and α^k_{ij} is an unknown coefficient of the linear approximation. In order to simplify the network, we want the set $\mathcal{I}^k$ to be smaller than the original layer's size p_k. We propose the following optimization to find $\mathcal{I}^k$ and α^k_{ij}:

$$\underset{\mathbf{A}^k}{\text{argmin}} \ \|\mathbf{H}^k - \mathbf{A}^k \mathbf{H}^k\|^2_F, \ \text{ subject to } \ \|\mathbf{A}^k\|_{col-0} \leq T^k \tag{10}$$

$$\mathbf{A}^k_{ij} = \begin{cases} \alpha^k_{ij}, & \text{if } j \in \mathcal{I}^k \\ 0, & \text{otherwise} \end{cases}, \qquad \mathbf{H}^k = \begin{pmatrix} h^k_{1,1} & \dots & h^k_{1,N} \\ \vdots & \ddots & \vdots \\ h^k_{p_k,1} & \dots & h^k_{p_k,N} \end{pmatrix} \tag{11}$$

The optimization in (10) enforces a small number of hidden nodes to linearly represent well all hidden nodes for all training samples. This is achieved by constraining the quasi-norm $\|\mathbf{A}^k\|_{col-0}$, which is the number of nonzero columns, to be less than T^k. Finally, the subset $\mathcal{I}^k$ is obtained from the indices of non-zero columns in $\mathbf{A}^k$. We use simultaneous orthogonal matching pursuit [15] to efficiently minimize the loss function in (10), which takes less than 2 seconds for each network in our experiments.

Once we find $\mathcal{I}^k$ and all the coefficients, the computation can be reduced by shrinking the connection at each layer (in short, ShrinkConnect) by discarding hidden nodes that are not in the set $\mathcal{I}^k$. In order to compensate for the discarded hidden nodes, the filters coefficients have to be updated as follows:

$$\mathbf{F}^{k+1} \leftarrow \mathbf{F}^{k+1}_{\mathcal{I}^k} + \boldsymbol{\Delta}^{k+1}, \ \text{ where } \ \boldsymbol{\Delta}^{k+1} = \mathbf{F}^k_{\setminus \mathcal{I}^k} \mathbf{A}^k \tag{12}$$

where $\mathbf{F}^k_{\mathcal{I}^k}$ and $\mathbf{F}^k_{\setminus \mathcal{I}^k}$ are the matrices formed by the columns of $\mathbf{F}^k$ whose indices are in and outside of $\mathcal{I}^k$, respectively. This update is derived using the relationship

in (9). Intuitively, what (12) does it adding the weights $\boldsymbol{\Delta}^{k+1}$ to the remaining filters so that the output of the simplified network is similar to that of the original network. In practice, we set the sparsity level T^k of all layers to a certain percentage of the original layer's size (e.g. from 10% to 90%) and pick the smallest network that does not degrade the prediction accuracy on training data by more than 2%. We re-train LSDN with 10 epochs after performing ShrinkConnect to refine the whole network. In most cases, the network could be reduced $4\times$ without losing much in prediction accuracy. We note that training a LSDN of the same size from the scratch or randomly removing hidden nodes yield worse results.

4 Experiments

We perform experiments on NAMIC brain dataset to demonstrate the effectiveness of our method. All images are registered, within domain and across domain, to a reference image using rigid transformations. The registration is necessary for voxels' locations to convey meaningful spatial information. We use leave-one-out cross-validation setting, in which 18 image pairs are used for training and one pair is left for testing. Our process of synthesizing a full image contains a training phase and a test phase.

Training phase: We are given a set of training pairs of images. Each pair has one image from a source domain (e.g. MRI-T1) and another image from a target domain (e.g. MRI-T2) of the same subject. We assume that our data are 3-dimensional volumes. Source images are cropped into small voxels of size $3 \times 3 \times 3$. The source voxels's intensities and their corresponding center's coordinates, denoted as $(\mathbf{s}, \mathbf{x})$, are used as input for training a LSDN network. The intensities at centers of target voxels are treated as the desirable outputs. We investigate two network configurations denoted LSDN-1 and LSDN-2 whose layers sizes are [27-200-20-1] and [27-400-40-1], respectively. The learning rate λ is set to 0.25, and the constant σ to 0.5.

Test phase: We apply LSDN over all voxels of a given source image in a sliding-window fashion. The predicted intensity values of all source-domain voxels are arranged according to the voxel centers' coordinates to create a synthesized target image. We note that computational complexity for applying LSDN to one voxel is $\mathcal{O}(p_s p_2 + p_x p_2 + \sum_{k=3}^{K} p_{k-1} p_k)$, where p_s and p_x are the dimensions of the intensity feature and the spatial location, respectively. This is slightly more expensive than that of a vanilla network, which is $\mathcal{O}(p_s p_2 + \sum_{k=3}^{K} p_{k-1} p_k)$. However, we will see that ShrinkConnect further reduces the computation of LSDN, making our network significantly faster than the vanilla deep network.

Results and discussion: We use signal-to-noise ratio (SNR) as the measure to evaluate different methods. Table 1 compares average SNR for different methods. One of the methods uses a vanilla deep network (VDN) with only intensity feature. Another approach, denoted as concatenation deep network (CDN), train a deep network on the concatenation of intensity feature and spatial coordinates. We also compare with recent methods in literature such as modality propagation (MP) [18] and coupled sparse representation [3]. The improvements in SNR is quite significant for LSDN compared to other approaches, especially for the case of T2$\rightarrow$T1 synthesis. It is interesting to see that the synthesis results from T2$\rightarrow$T1 is much better than from T1$\rightarrow$T2. We con-

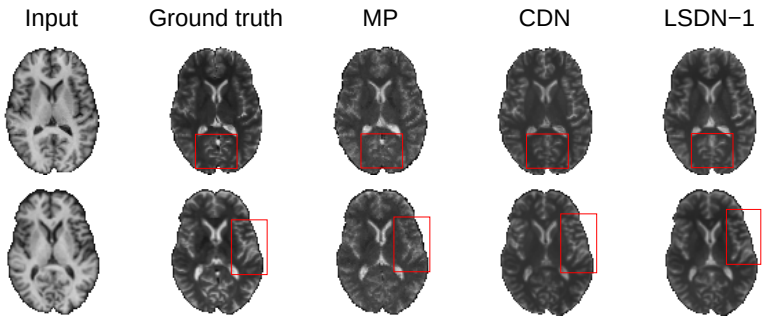


Fig. 3: Visual comparison of synthesized MRI-T1 volumes using different approaches. Red boxes indicates regions where there are significant differences among approaches.

Method	**SNR (T1→T2)** (dB)	**SNR (T2→T1)** (dB)	**Training** (hour)	**Synthesis** (second)
MP [18]	13.64 ± 0.67	15.13 ± 0.88	n/a	928
Coupled Sparse [3]	13.72 ± 0.64	15.24 ± 0.85	0.8	245
VDN	12.67 ± 0.6	14.19 ± 0.82	1.2	23.5
CDN	13.79 ± 0.68	15.36 ± 0.88	1.2	23.6
LSDN-Small	12.53 ± 0.75	13.85 ± 0.86	0.6	9.2
LSDN-1	**14.82 ± 0.72**	**17.09 ± 0.94**	**1.4**	**29.5**
LSDN-2	**14.93 ± 0.73**	**17.39 ± 0.91**	**2.5**	**68.0**
LSDN-1+ShrinkConnect	**14.79 ± 0.72**	**17.05 ± 0.91**	**1.4**	**9.2**
LSDN-2+ShrinkConnect	**14.80 ± 0.74**	**17.1 ± 0.86**	**2.5**	**21.5**

Table 1: Comparison of signal to noise ratios and speeds on NAMIC brain dataset.

jecture that more details of the brain are visible under T2 than under T1. From CDN results, we observe that the concatenation of intensity feature and spatial feature is not as effective as LSDN. ShrinkConnect reduces the LSDN-1 and LSDN-2 sizes respectively to [27-50-5-1] and [27-100-10-1] without losing much in prediction accuracy, as shown in the last two rows of Table 1. To validate if we could obtain the same accuracy without ShrinkConnect, we train a network of size [27-50-5-1] from scratch, denoted as LSDN-Small. We can easily notice the accuracy of LSDN-Small is significantly lower than that of LSDN+ShrinkConnect. This demonstrates the effectiveness of our network simplification technique.

The results also indicate that increasing network size helps improve the accuracy, at the cost of higher run-time computation. Table 1 provides training time of LSDN with 300 epochs. The average time it takes LSDN-1 to synthesize an image is 29.5 seconds compared to 23.5 seconds of VDN. With ShrinkConnect, the run time is reduced to 9.2 seconds per image, which is $26\times$ faster than coupled sparse method and $100\times$ faster than modality propagation. Fig 3 provides visual comparisons for different methods. Note we only show small images due to space limit but the resolution of images is much higher than that in Fig. 3. All of our algorithms were implemented in Matlab. Our codes were run on a 20-core machine with Intel X5650 processor (2.66 GHz).

5 Conclusions

We proposed LSDN as a way to incorporate both image intensity feature and spatial information into a deep network. We also proposed a novel network simplification technique for reducing computation of LSDN. Our approach outperforms the state of the art in both accuracy and computation on MR brain image synthesis. In the future, we plan to investigate the use of LSDN for other applications such as segmentation.

References

1. Alexander, D.C., Zikic, D., Zhang, J., Zhang, H., Criminisi, A.: Image Quality Transfer via Random Forest Regression: Applications in Diffusion MRI. In: MICCAI (2014)
2. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010, pp. 177–186. Springer (2010)
3. Cao, T., Zach, C., Modla, S., Powell, D., Czymmek, K., Niethammer, M.: Multi-modal Registration for Correlative Microscopy Using Image Analogies. MIA 18(6), 914–926 (2014)
4. Commowick, O., Warfield, S.K., Malandain, G.: Using Frankenstein's Creature Paradigm to Build a Patient Specific Atlas. In: MICCAI (2013)
5. Fischl, B., Salat, D.H., van der Kouwe, A.J.W., Makris, N., Ségonne, F., Quinn, B.T., Dale, A.M.: Sequence-independent Segmentation of Magnetic Resonance Images. NeuroImage 23, S69 – S84 (2004)
6. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image Analogies. In: Annual Conference on Computer Graphics and Interactive Techniques (2001)
7. Iglesias, J.E., Konukoglu, E., Zikic, D., Glocker, B., Leemput, K.V., Fischl, B.: Is Synthesizing MRI Contrast Useful for Inter-modality Analysis? In: MICCAI (2013)
8. Jog, A., Roy, S., Carass, A., Prince, J.L.: Magnetic Resonance Image Synthesis through Patch Regression. In: ISBI (2013)
9. Kroon, D., Slump, K.: MRI Modalitiy Transformation in Demon Registration. In: ISBI (2009)
10. Prakosa, A., Sermesant, M., Delingette, H., Marchesseau, S., Saloux, E., Allain, P., Villain, N., Ayache, N.: Generation of Synthetic but Visually Realistic Time Series of Cardiac Images Combining a Biophysical Model and Clinical Images. IEEE TMI 32(1), 99–109 (2013)
11. Roy, S., Carass, A., Jog, A., Prince, J.L., Lee, J.: MR to CT Registration of Brains Using Image Synthesis. In: SPIE Medical Imaging (2014)
12. Roy, S., Carass, A., Prince, J.L.: Magnetic Resonance Image Example-based Contrast Synthesis. IEEE Transactions on Medical Imaging 32(12), 2348–2363 (2013)
13. Roy, S., Carass, A., Shiee, N., Pham, D.L., Prince, J.L.: MR Contrast Synthesis for Lesion Segmentation. In: ISBI (2010)
14. Rueda, A., Malpica, N., Romero, E.: Single-image Super-resolution of Brain MR Images Using Overcomplete Dictionaries. Medical Image Analysis 17(1), 113–132 (2013)
15. Tropp, J.A., Gilbert, A.C., Strauss, M.J.: Simultaneous sparse approximation via greedy pursuit. In: IEEE ICASSP. vol. 5, pp. v–721 (2005)
16. Wang, S., Zhang, L., Liang, Y., Pan, Q.: Semi-coupled Dictionary Learning with Applications to Image Super-resolution and Photo-sketch Synthesis. In: CVPR (2012)
17. Wein, W., Brunke, S., Khamene, A., Callstrom, M.R., Navab, N.: Automatic CT-Ultrasound Registration for Diagnostic Imaging and Image-guided Intervention. MIA (2008)
18. Ye, D.H., Zikic, D., Glocker, B., Criminisi, A., Konukoglu, E.: Modality Propagation: Coherent Synthesis of Subject-specific Scans with Data-driven Regularization. In: MICCAI (2013)