Fast Bilateral Symmetry Detection Using Inverted Gradient Hash Maps

R. Gonzalez(✉) and L. Lincoln

School of ICT, Griffith University, Parklands Drive, Southport, QLD, Australia
r.gonzalez@griffith.edu.au

Abstract. This paper presents a fast and novel algorithm for bilateral symmetry detection based on inverted gradient hash maps (IGHMs). A hash map is an associative array that stores image gradient magnitudes and orientations in the form of an inverted index. This mapping of image gradients to their locations permits points of interest to be located very rapidly without needing to search through the image. Unlike many symmetry operators it is able to detect large-scale symmetry. The method is described and experimentally evaluated against existing methods for bilateral symmetry detection.

Keywords: Symmetry detection · Reflective · Bilateral · Mirror

1 Introduction

The detection of symmetry has an important role in visual perception. It is also a fundamental process within many image-processing applications. It has a variety of uses such as an attentional operator in computer vision, as a detector of man made and natural objects such as human faces and also shape representation and characterization. While different kinds of symmetry can be detected, the most common are radial (rotational) and bilateral (reflective or mirror) symmetries.

Perhaps the best-known symmetry detector is Reisfeld's generalized symmetry transform [1]. This can detect bright and dark, isotropic or radial symmetries. It's main drawback is its computation complexity, having an order of NK^2, where N is the number of pixels in an image and K is the kernel size A variety of other bilateral and radial symmetry detectors have been proposed in the literature such as those utilizing the Hough transform [1–3] which have an order of KBN, where B is the number of angular steps. Faster algorithms have been proposed for detecting only radial symmetry such as that of Loy [5] that has an order of KN. While Reisfeld's algorithm requires 259 Mflops for a 30×30 kernel and a 521×512 pixel image, the Hough based methods require around 34 Mflops, and Loy's approach requires between 8–19 Mflops [6]. The main limitation with many of these methods is that they require multiple passes through the image to consider symmetry at multiple scales. Detecting bilateral symmetry across an entire image requires a kernel that is the same size as the image. In this case the required computing time becomes very large.

Typical symmetry detection operates by either searching for matching gradients within a kernel using the image's intrinsic Cartesian space or by using a one to many

P.C. Vinh et al. (Eds.): ICTCC 2016, LNICST 168, pp. 72–81, 2016.
DOI: 10.1007/978-3-319-46909-6_8

voting scheme using an alternate parameterized 'Hough' accumulator space. In contrast, the method proposed in this paper operates purely in gradient space and avoids searching by using an inverted index. Until recently, inverted indices have only been used in for speeding up retrieval in databases [7]. Yet as described in [8] the concept of an inverted gradient space representation presents certain advantages for algorithms that exploit image gradients. Most importantly, by eliminating the need to search for matching gradients within a kernel, it can reduce processing time by up to two orders of magnitude.

In the following sections the inverted gradient space representation using hash maps is first described. Then fast bilateral symmetry detection using an IGHM is presented in Sect. 2.3. Experimental results for the performance of this algorithm relative to existing methods are presented in Sect. 3 and conclusions in Sect. 4.

2 Method Description

2.1 Inverted Gradient Hash Maps

A typical image $\boldsymbol{f}$, is a mapping of image coordinates *(x,y)* to pixel intensity values *i*:

$$\boldsymbol{f}: (x, y) \rightarrow i \text{ where } [x,y] \in \mathbf{N}^2 \tag{1}$$

The derivative of the image intensity at a given coordinate (x,y) gives rise to the local gradient and is defined by the magnitude and orientation $\{m, \theta\}$ at that coordinate:

$$m_{x,y} = \sqrt{\left(\frac{\partial}{\partial x} i_{x,y}\right)^2 + \left(\frac{\partial}{\partial y} i_{x,y}\right)^2} \text{ and } \theta_{x,y} = \arctan\left(\frac{\partial}{\partial x} i_{x,y}, \frac{\partial}{\partial y} i_{x,y}\right) \tag{2}$$

A mapping of image coordinates to their corresponding gradients is known as the gradient image **g**:

$$\boldsymbol{g}: (x, y) \rightarrow \{m, \theta\} \tag{3}$$

A reverse mapping from the image gradients to image coordinates is known as an inverted gradient image $\boldsymbol{h}$.

$$\boldsymbol{h}: \{m, \theta\} \rightarrow (x, y) \tag{4}$$

While there is a single gradient for each image coordinate, there may be many image coordinates that have the same gradient. Thus, the inverted gradient image cannot be simply stored as a two dimensional array. Instead the inverted gradient image is best stored as a hash map where collisions in $\{m, \theta\}$ are resolved via chaining. This hash map is simply a two dimensional array of lists. These lists are indexed by the gradient magnitude and orientation and store the coordinates of the pixels having the indicated local gradient as depicted in Fig. 1.

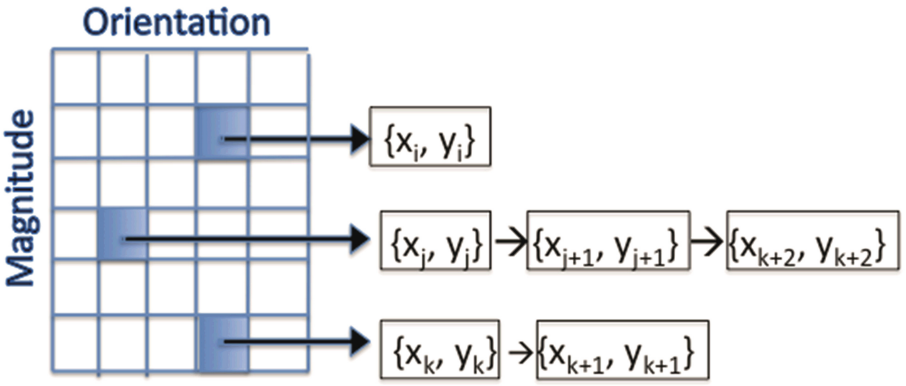

Fig. 1. Inverse Gradient Hash Map Data Structure

The size of the two-dimensional hash map is determined by the desired angular and scalar resolution used for the values (m, θ). Typically $m \rightarrow 0..255$ and $\theta \rightarrow 0..360$ in one degree increments.

2.2 Symmetry Detection

In Reisfeld's transform a full search of the image is required to find potentially contributing pairs of pixels within a given range defined by the kernel size. Each pixel is checked multiple times. This is avoided using an inverse gradient hash map. A single pass is made through the image visiting each pixel only once in order to populate the hash map. Once the hash map is formed all of the relevant pairs of gradients can be found via a simple lookup and their contribution assessed in a single pass.

The algorithm proceeds by simply looking up all the pairs of pixels having similar gradient magnitudes and opposing orientations (i.e. are rotated by approximately 180°). A margin of up to $\pm 45°$ is used to include gradients that partially contribute to symmetry. This lookup results in two lists containing the coordinates of all the pixels matching the gradient criteria. The contribution of each pair of pixels in these two lists to the symmetry at the given orientation can then be calculated. Thus symmetry in specific directions can be readily determined or if desired, for all directions.

In the following pseudo code description of the basic algorithm, the outer two loops iterate over the entire hash table using indices *a* and *m*. Ideally magnitudes below some noise floor *thres* should be ignored. The next two inner loops using indices *b* and *n* only iterate respectively over the range of $(a + 45 + 180)$ and $(a - 45 + 180)$ degrees and also $(m - 5)$ to $(m + 5)$ to allow for noise in the gradient magnitude. For each pair of indices *(a, m)* and *(b, n)* two lists containing the relevant points are referenced using two pointer variables ptrA and ptrB. For each pair of coordinate points, *i* and *j* in these lists, the contribution to the symmetry is calculated (as further described below) and accumulated if it is within the target range of influence.

```
function symmetry(IGHM, output, maxdist, thres)
Begin
  //-- visit all gradients in hash map --
  Foreach a = gradient orientations from 0…360
    Foreach m = gradient magnitudes > thres
      //-- get contributing gradient points --
      For (b = a-45; b < a+45; b++)
        For (n = m-5; n < m+5; n++)
        Begin
          Var c = (b + 180) % 360
          var ptrA = &(IGHM[a][m])
          Var ptrB = &(IGHM[c][n])
          //-- walk coordinate lists --
          For (i=0; i< ptrA->len; i++)
            For (j=0; j< ptrB->len; j++)
            Begin
              Var Dij = Distance(ptrA[i], ptrB[j])
              Var Cij = Calc(a,b,ptrA[i], ptrB[j])
              Var x = (ptrA[i].x + ptrB[j].x) / 2
              Var y = (ptrA[i].y + ptrB[j].y) / 2
              If (Dij < maxdist) Output[x][y] += Cij
            End
        End
End
```

The contribution C_{ij} of each pair of pixels p_i and p_j is a function of the magnitude of the nominated gradient m_i and m_j, the deviation of the gradient orientation to the suggested axis of symmetry and optionally, the distance between contributing pixels D_{ij}. This contribution is assigned to the halfway point q_{ij} between the contributing pixel pair. A two-dimensional accumulator C is used to store the contribution of each pair at the halfway points.

Given a pair of contributing pixels p_i and p_j, as depicted in Fig. 2, the first step is to determine the orientation of the suggested axis of symmetry φ_{ij}. This is calculated as the normal to the vector connecting p_i and p_j, thus:

$$\phi_{ij} = \arctan 2\left(\frac{p_i(y) - p_j(y)}{p_j(x) - p_i(x)}\right) \tag{5}$$

Next the angular difference between the orientation of the suggested axis of symmetry φ_{ij} and the orientation of the gradients θ_i and θ_j at each point is found, using a formulation that avoids costly transcendental functions.

$$\Delta_i = \left|\pi - \left|\left|\phi_{ij} - \theta_i\right| - \pi\right|\right| \; and \; \Delta_j = \left|\pi - \left|\left|\phi_{ij} - \theta_j\right| - \pi\right|\right| \tag{6}$$

The contribution of the gradients p_i and p_j becomes

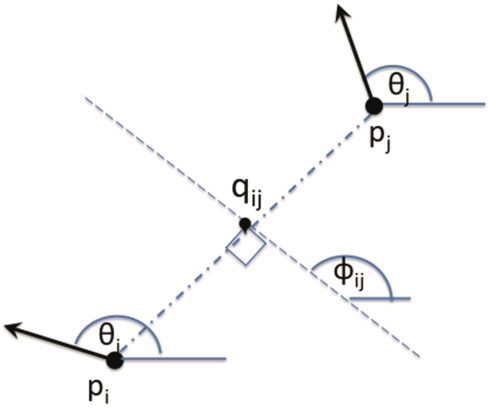

Fig. 2. The geometry associated with points contributing to symmetry.

$$C_{ij} = \frac{m_i \times m_j}{1 + \left|\Delta_i - \Delta_j\right|} \tag{7}$$

The Euclidean distance between contributing pixels D_{ij} can be used to limit the range of influence that gradients can have exert by ignoring the contributions of points beyond this range limit.

The symmetry can be calculated to consider all gradients or only those that pertain to bright objects on a dark background (bright symmetry) or dark object on a bright background (dark symmetry). To do this the orientation of the gradient relative to the halfway point needs to be determined. First the angle A_i from each point p_i to the halfway point q_{ij} is found and similarly for A_j.

$$A_i = \arctan\left(\frac{q_{ij}(y) - p_i(y)}{p_i(x) - q_{ij}(x)}\right) \; and \; A_j = \arctan\left(\frac{q_{ij}(y) - p_j(y)}{p_j(x) - q_{ij}(x)}\right) \tag{8}$$

These values are next compared to the gradient orientation θ_i and θ_j at p_i and p_j to see whether the gradients are aligned towards or away from A_i and A_j.

$$\varphi_i = \left|\theta_i - A_i\right| \; and \; \varphi_j = \left|\theta_j - A_j\right| \tag{9}$$

Depending on the value of ψ_i and ψ_j the contribution C_{ij} for that pair is retained or set to zero. For bright symmetry the following relationship holds:

$$C_{ij} = \begin{cases} 0 & \left(\varphi_i < 90 \vee \varphi_i > 270\right) \wedge \left(\varphi_j < 90 \vee \varphi_j > 270\right) \\ C_{ij} & otherwise \end{cases} \tag{10}$$

For dark symmetry the following relationship is used

$$C_{ij} = \begin{cases} 0 & \left(\varphi_i > 90 \wedge \varphi_i < 270\right) \wedge \left(\varphi_j > 90 \wedge \varphi_j < 270\right) \\ C_{ij} & otherwise \end{cases} \tag{11}$$

2.3 Complexity Analysis

From the foregoing discussion one can observe that the computational complexity of this method, given the three sets of two loops in the basic algorithm is in the order of:

$$\begin{aligned} &(A * M) * (A/4 * 10) * (La * Lb) \\ &= k * A^2 * M * La * Lb \end{aligned} \tag{12}$$

Here A and M are respectively the number of orientation and magnitude bins used in the hash map and are a function of the angular and scalar resolution used for the values (m, θ). The constant k for a typical implementation equals 2.5, but could be reduced if desired. La and Lb are the lengths of the two coordinate lists and are related to the load factor of the hash map. The load factor will be a function A and M relative to the image size in pixels N. Typically the average load factor L_f is defined as:

$$L_f = N/(A * M) \tag{13}$$

Since on average the lengths of the two lists La and Lb will be equal to the load factor the following substitutions can be made to simplify the calculation of (12):

$$\begin{aligned} A^2 * M * L_f * L_f &= A^2 * M * N^2/(A * M)^2 \\ &= N^2/M \end{aligned} \tag{14}$$

Although the worst-case complexity of this algorithm will never be greater than N^2, in practice this upper bound will never be approached unless the hash map only a single bin for the gradient magnitude, which defeats the purpose of using a two dimensional hash map. The complexity using a 360 × 256 cell hash map is unlikely to ever exceed $N^{1.8}$ as can be seen from Table 1 for the different size images:

Table 1. Computational Complexity for various image sizes.

Image pixels	Complexity
256 × 256	$N^{1.5}$
512 × 512	$N^{1.56}$
1024 × 1024	$N^{1.6}$
2048 × 2048	$N^{1.65}$
4096 × 4096	$N^{1.67}$
8192 × 8192	$N^{1.69}$
1,000,000 × 1,000,000	$N^{1.8}$

In addition to the symmetry calculation the cost of generating the hash map needs to be considered. As this is a linear operation with a complexity of N it has fairly small impact on the total complexity.

3 Experimental Results

Symmetry detection experiments were run using a variety of images. For ease of comparison, a version of the "Cards, keys and hand" image from Reisfeld's paper was used with the results shown in Fig. 3 for Reisfeld's method in the centre and the proposed one on the right.

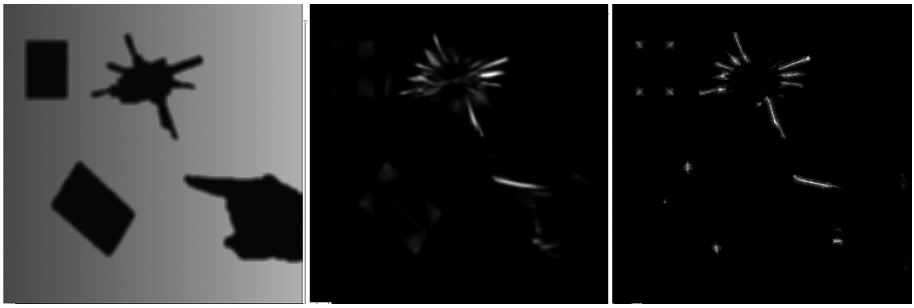

Fig. 3. Bilateral symmetry with the original image on the left, symmetry detected Reisfeld's method using a kernel size of 16 in the middle and IGHM based method with a D_{ij} of 16.

While symmetry detection algorithms typically work well for simple, small-scale symmetry such as that in Fig. 3, the real challenge is for detecting image wide symmetry. For this experiment the well-known Lena image was first mirrored horizontally and then vertically around a central axis as shown in Fig. 4. In addition, other images of natural and man made objects as depicted in Fig. 5 were also evaluated. In both Figs. 4 and 5 the mirrored images are shown on the left, with the result of the proposed symmetry operator in the centre. The axis of symmetry is found by applying a line detector after non-maximal suppression to the resulting image.

To compare the ability of the proposed method to detect image wide symmetry relative to Reisfeld's method, his operator was applied with a kernel size of 256 to the mirrored Lena image used in Fig. 4. As the results in Fig. 6 show, the Reisfeld operator (on the right side) completely fails to identify the symmetry in the image.

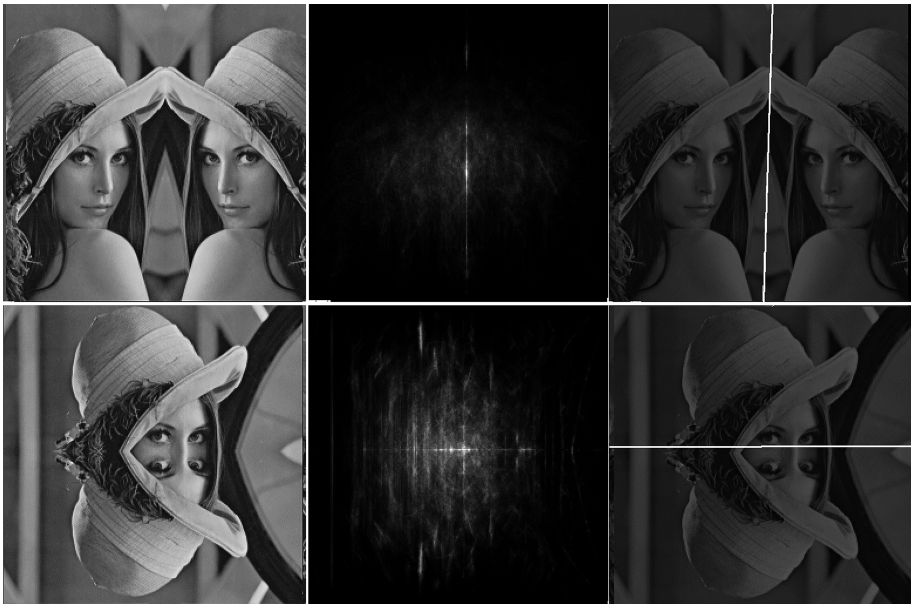

Fig. 4. Symmetry of mirrored images with the originals on the left, detected symmetry in the centre and the resulting axes of symmetry superimposed over the original on the right.

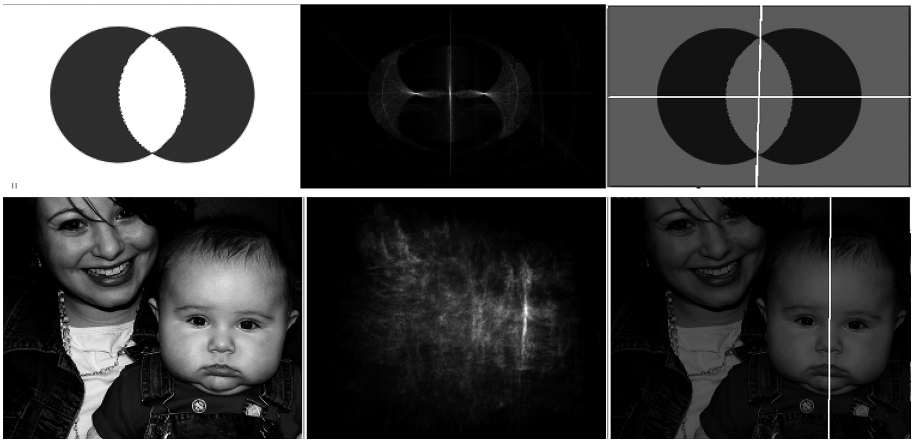

Fig. 5. Bilateral symmetry with the original images on the left, output of the symmetry detector in the centre and the resulting axes of symmetry superimposed over the original on the right.

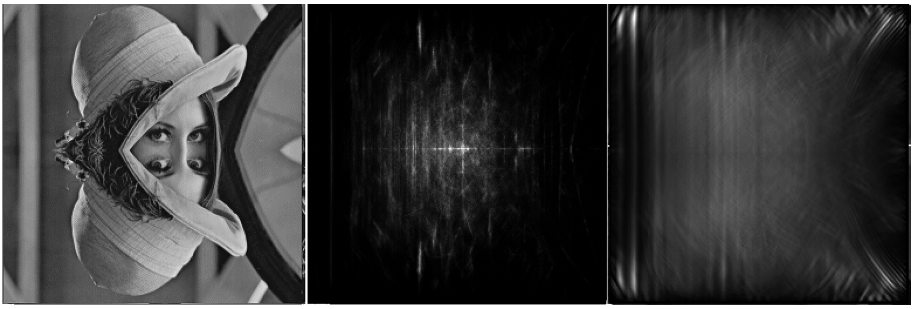

Fig. 6. Comparison of large field symmetry detection. The original is on the left, the proposed method in the centre and Reisfeld's on the right side.

The time performance of the proposed method was evaluated on a MacBookPro with a 2.6 GHz Intel i7 with 16 GB of memory. The running time of Reisfeld's general symmetry transform was measured for kernel sizes of 8, 16 and 256 as well as the running time for the proposed method with corresponding maximum gradient influence distance constraints (Table 2).

Table 2. Empirical bilateral symmetry detection computation time.

Image	Reisfeld-8	Reisfeld-16	Reisfeld-256	IGMH-16	IGHM-256
Card, keys and hand	1,173 ms	4,322 ms	100,222 ms	152 ms	624 ms
Lena 256 × 256	1,207 ms	4,535 ms	104,097 ms	811 ms	2,527 ms
Elvis 256 × 256	1,218 ms	4,716 ms	105,931 ms	898 ms	1,750 ms

These results were compared to those reported in the literature for two Hough based methods by Li et al. [2] and Yip [4]. The results reported by Li et al. were obtained on a 2.2 GHz Pentium using Matlab and took from 10 to 20 s for a 300 × 300 image. Li et al. further reported that a native C language implementation, using simple 64 × 64 pixel images and subsampling was able to run in under one second. R.K.K. Yip reported taking between 20–50 s results using a 500 MHz Pentium 2 for simple 256 × 256 images of polygons, and about 50 min for more complex synthetic images. In comparison the proposed IGMH based method processes simple 256 × 256 images in about 150 ms. While it is half an order of magnitude faster when using small kernel sizes on complex images, the processing time is orders of magnitude less when considering image wide symmetry, as is predicted from the complexity analysis.

Like the proposed method, Patraucean's Hough voting based approach [3] is able to detect image wide mirror symmetries however no performance figures are reported other than that the validation time alone, for each symmetry candidate on an 800 × 600 image requires 2 s on an 2.53 GHz, Intel i5 based computer. Since typically between five to ten candidates need to be validated, the total time for the validation alone amounts to 10 to 20 s. This however does not include the time required for the selection of the symmetry candidates themselves. In contrast, for similar sized images the proposed IGHM based method completes the task in about 10 s.

4 Conclusions

Detection of symmetry is a fundamental task in image processing but it has traditionally been computationally expensive. While efficient algorithms have been developed for calculating radial symmetry, algorithms for the detection of bilateral symmetry are still relatively slow. This paper has presented a fast and novel approach to finding bilateral symmetry based on inverted gradient hash maps. Not only is it significantly faster than other methods for detecting bilateral symmetry but it can successfully detect such symmetry on large scales. Future work will consider methods for automatic selection of D_{ij} the maximum distance between contributing pixels.

References

1. Reisfeld, D., Wolfson, H., Yeshurun, Y.: Context-free attentional operators: the generalized symmetry transform. Int. J. Comput. Vision **14**(2), 119–130 (1995)
2. Li, W.H., Zhang, A.M., Kleeman, L.: Fast global reflectional symmetry detection for robotic grasping and visual tracking. In: Proceedings of Australasian Conference on Robotics and Automation, December 2005
3. Patraucean, V., von Gioi, R.G., Ovsjanikov, M.: Detection of mirror-symmetric image patches. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 211–216, 23–28 June 2013
4. Yip, R.K.K.: A Hough transform technique for the detection of reflectional symmetry and skew-symmetry. In: Pattern Recognition Letters, Vol. 21, Issue 2, pp. 117–130, Feb 2000
5. Loy, G., Zelinsky, A.: A fast radial symmetry transform for detecting points of interest. IEEE Trans. Pattern Anal. Mach. Intell. **25**(8), 959–973 (2003)
6. Loy, G.: Computer Vision to See People: a basis for enhanced human computer interaction. Ph.D. Thesis, Australian National University, January 2003
7. Zobel, J., Moffat, A.: Inverted files for text search engines. ACM Comput. Surv. **38**(2), 6 (2006)
8. Gonzalez, R.: Fast line and circle detection using inverted gradient hash maps. In: The 40th International Conference on Acoustics, Speech and Signal Processing, Proceedings of (ICASSP2014), Brisbane, Australia, 19–24 April 2015. IEEE (2015)