FAST AND RELIABLE TDOA ASSIGNMENT IN MULTI-SOURCE REVERBERANT ENVIRONMENTS

Martin Kreißig and Bin Yang

Institute of Signal Processing and System Theory, University of Stuttgart
email: {martin.kreissig, bin.yang}@iss.uni-stuttgart.de

ABSTRACT

The localization of acoustic sources based on Time Difference of Arrivals (TDOA) is very vulnerable in reverberant environments. In this paper, we propose a method to synthesize fully and partially consistent TDOA combinations. They fulfill the zero cyclic sum condition along all loops, which is a necessary condition for assigning TDOAs to an acoustic source. Our method is based on an efficient search of all sets of maximally connected compatible fundamental loops in a compatibility-conflict graph. We both prove the correctness of our algorithm and show some experimental results.

Index Terms— acoustic source localization, TDOA assignment, synthesis of consistent graph, compatibility-conflict graph

1. INTRODUCTION

Acoustic source localization is widely performed by using time difference measures obtained from the cross-correlation of two microphone signals. This is necessary in applications where the time of emission is unknown. More precisely, the TDOAs are obtained from the peaks in the Generalized Cross-Correlation (GCC) that is quite robust in moderate reverberant environments [1, 2]. Especially the GCC-PHAT variant [3] allows TDOA peaks to be detected at low SNR due to its pre-whitening filter.

Nevertheless, this filter also increases the noisy part of the spectrum and hence adds erroneous candidates to our TDOA estimates. Moreover, the periodicity of the speech signal, reflections and measurement errors increase the number of spurious TDOAs. To overcome this ambiguity, many approaches have been proposed that apply either a special cost function to improve the TDOA estimation [4] or apply even more complex algorithms like SRP-PHAT [2] or BSS [5, 6]. Even though these approaches reach quite good detection rates, they are restricted to less speakers than microphones (BSS, ICA) and limited by their high computational complexity for real-time applications.

In this paper, we present a new improved algorithm that examines all TDOA candidates of all microphone pairs and finds TDOA sets that stem from the same source and same propagation paths. These sets can be full (containing all sensor pairs) or partial. This algorithm can be downloaded from [7].

This paper is structured as follows: Sec. 2 addresses the ambiguity problem of TDOA and describes different disambiguation approaches. Sec. 3 gives an overview of our synthesis approach. In Sec. 4 we show how to reformulate the last step of our synthesis approach as a new compatibility-conflict graph problem and present a new algorithm to find its solutions. Finally, we show some measurement results in Sec. 5.

2. TDOA DISAMBIGUATION

2.1. Ambiguity of TDOA assignment

The TDOA candidates $\hat{\tau}$ are computed as positions of peaks in the GCC-PHAT function $R_{ij}(\tau)$ for the microphone pair (i, j). As discussed in [8] and [9], the peak detection is not unique due to multiple sound sources, reflection paths and periodicities in the source signals. In order to obtain all direct path TDOAs, one has to take several TDOA candidates per microphone pair. With M microphones there are at maximum $N = \binom{M}{2}$ such pairs with K_n TDOA candidates each $(n = 1, \ldots, N)$.

For a successfull localization, one has to pick for each source the correct direct path TDOAs at each micropohone pair and combine them. These TDOA sets are hard to find as there are $\prod_{n=1}^{N} K_n$ possible sets.

2.2. Disambiguation techniques

In [9] a speed estimate criterion was proposed to decide whether a given TDOA set stems from direct path Time of Arrival (TOA). The criterion is promising as the results in Sec. 5 show.

The disambiguation considered in this paper was initiated in [8]. It exploits the property that the sum of TDOAs from the same source and propagation paths along a loop is zero: $\tau_{ij} + \tau_{jk} + \ldots + \tau_{li} = 0$. A set of TDOA estimates from different microphone pairs which fulfills this condition along all loops is called consistent. In [8] the synthesis of consistent graphs is performed by combining consistent triples (loops of length 3). This approach, however, is restricted to the existence of consistent triples which is not always the case in practice.

This restriction is relaxed in [10] where a set of fundamental loop (FL) is used to represent all loops in a TDOA graph. A FL whose TDOAs satisfy the zero cyclic sum condition is called a consistent fundamental loop (cFL). However, these cFLs are combined together by a synthesis algorithm resulting in fully consistent graphs. If a microphone pair measures only spurious TDOAs, it can happen that no consistent graphs are found at all by the algorithm in [10] though partially consistent TDOA sets (i.e. with some missing pairs) do exist.

In this paper we propose an improved algorithm to combine cFLs to fully or partially consistent TDOA sets.

3. SYNTHESIS OF CONSISTENT GRAPHS

The microphone set is represented by a graph $G(V, E)$ with the vertex set $V = \{m_1, \ldots, m_M\}$ representing the microphones and the edge set $E = \{e_1, \ldots, e_N\}$ representing the microphone pairs. The TDOA candidates define the search space $W = W_1 \times \cdots \times W_N$

where $W_n = \{\tau_{n,1}, \ldots, \tau_{n,K_n}\}$ are the TDOAs of the n^{th} microphone pair.

The synthesis of consistent graphs is performed in several steps as shown in Fig. 1. They are discussed in the subsequent sections.

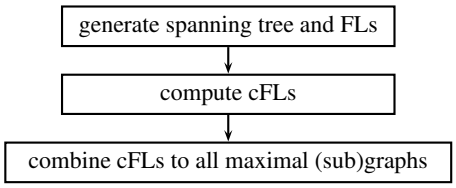

Fig. 1. Overview of the synthesis of all maximal consistent subgraphs.

3.1. Generation of cFLs

We determine the cFLs according to the proposed method in [11, 10, 12]. The spanning tree of the graph is generated first. Then each of the unused edges that define the complementary tree closes a FL when attached to the spanning tree. For a connected graph with M vertices and N edges, there is a total number of $N - M + 1$ FLs. For our purpose, we choose the Breadth-First Search (BFS) spanning tree [13] as it produces shorter FLs.

For each FL which typically consists of a small number (≥ 3) of edges we check all TDOA combinations. The FLs are represented as column vectors $\underline{l}_i \in \{-1, 0, 1\}^N$, where "1" represents an equally directed edge and loop and a "-1" counterwise. Thus the zero-cyclic sum condition can be written as $\underline{l}_i^T \underline{w} = 0$ for $i = 1, \ldots, N-M+1$. Due to quantization errors, noise and smoothing effects of overlapping peaks in the GCC-PHAT, we check the approximate consistency

$$|\underline{l}_i^T \underline{w}| \leq \epsilon \cdot \sqrt{||\underline{l}_i||_0}, \tag{1}$$

where ϵ is a threshold in the unit of sampling interval like the TDOA values τ_{ij} and $||\cdot||_0$ denotes the l_0-norm that enumerates the number of non-zero elements, i.e. the number of edges in the FL $\underline{l}_i$. The number of cFLs per topological FL $\underline{l}_i$ is assumed to be $\tilde{K}_i$. Next, we combine those cFLs that have the same TDOA values at the common edges.

3.2. Combination of cFLs

When two consistent subgraphs have the same TDOAs on common edges, they are called compatible and we can combine them together to a larger graph containing both subgraphs. If the TDOAs on common edges are different, the two subgraphs are in conflict and we are not allowed to combine them. The same applies when the two subgraphs have no common edges at all. This bottom-up synthesis, i.e. a successive combination of compatible consistent subgraphs, starting with the cFLs from Sec. 3.1, always returns a larger graph that is consistent as well. In Sec. 4 we propose a combination algorithm that finds all, maximal combinations of compatible cFLs and thus all consistent TDOA graphs based on the particular set of FLs.

4. FINDING ALL POSSIBLE COMPATIBLE COMBINATIONS OF CFLS

4.1. cc-graph

The problem of combining all compatible cFLs is not trivial, as all combinations have to be taken into account. Moreover, no redundant solutions, i.e. combinations that are subsets of other combinations, are allowed. Therefore, we represent the cFLs as vertices in a *compatibility-conflict graph (cc-graph)*. Each pair of vertices in a cc-graph has three possible relations. They can be compatible or in conflict or unrelated because the underlying two cFLs are compatible (same TDOAs for common edges) or in conflict (different TDOAs for common edges) or have no common edges at all.

These three states lead to the cc-graph with compatible, conflicting and free connections. An example is presented in Fig. 2 where the vertex set $V = \{v_k | k = 1, \ldots, 6\}$ represents the different cFLs. Two compatible cFLs are connected by a solid line and two conflicting cFLs by a dashed line. cFLs with no common edges are not connected in the cc-graph. The cc-graph in Fig. 2 has 4 sets of maximally connected compatible vertices: $\{v_1, v_2, v_3\}, \{v_1, v_2, v_4\}, \{v_2, v_3, v_5, v_6\}$ and $\{v_2, v_4, v_6\}$.

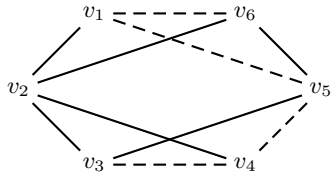

Fig. 2. A compatibility-conflict graph: compatible vertices are marked by a solid edge and conflicting vertices by a dashed edge

The adjacency matrix of the cc-graph is given by

$$\mathbf{A}_{cc} = \begin{bmatrix} 0 & 1 & 0 & 0 & -1 & -1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 \\ -1 & 0 & 1 & -1 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2}$$

where a "1" indicates a compatible neighbourhood and "-1" a conflicting one.

Based on the cc-graph we want to find all maximally connected combinations of cFLs. The proposed algorithm is similar to that of Bron and Kerbosch [14] for finding all sets of completely connected vertices. While Bron and Kerbosch only consider graphs with two different neighbourhood states (connected or not connected), our cc-graph has three different neighbourhood states and is more complex.

4.2. Algorithm

We call our algorithm to find all sets of maximally connected compatible vertices in a cc-graph characterized by A_{cc} the Compatibility-Conflict Graph (CCG) algorithm. First we introduce some notations. Let V be the set of all vertices in the cc-graph. Let $\tilde{V}$ be the set of currently considered vertices. The algorithm is initialized to $\tilde{V} = V$ at the beginning and changes during the algorithm because already visited vertices will be excluded from $\tilde{V}$. Let l denote the current solution, the set of connected compatible vertices. It is initialized to the empty set $\emptyset$ at the beginning. The notation

$$N_{\tilde{V}}(l) = \{v \in \tilde{V} | \exists u \in l : [\mathbf{A}_{cc}]_{vu} = 1 \wedge \nexists w \in l : [\mathbf{A}_{cc}]_{vw} = -1\} \tag{3}$$

denotes the set of vertices from $\tilde{V}$ that are compatible to at least one vertex in the solution l and have no conflict to any vertices in l. $[\mathbf{A}_{cc}]_{ij}$ indicates the element at the i^{th} row and j^{th} column of $\mathbf{A}_{cc}$. Similarly, we define

$$\bar{N}_{\tilde{V}}(l) = \{v \in \tilde{V} | \exists u \in l : [\mathbf{A}_{cc}]_{vu} = -1\} \tag{4}$$

as the set of vertices from $\tilde{V}$ that have conflict to at least one vertex in l. In other words, $N_{\tilde{V}}(l)$ and $\bar{N}_{\tilde{V}}(l)$ represent the compatible and conflicting neighbours of l in $\tilde{V}$, respectively. During the initialization of the algorithm, we use the convention $N_{\tilde{V}}(\emptyset) = \tilde{V}$ and $\bar{N}_{\tilde{V}}(\emptyset) = \emptyset$. Let X denote the set of vertices which have already been visited in previous iterations and thus shall be skipped. It is initiated to $\emptyset$ and is extended successively by the visited vertices. The CCG algorithm is started by the call `CombineAll` $(\mathbf{A}_{cc}, V, \emptyset, \emptyset)$ where the recursive routine `CombineAll` $(\mathbf{A}_{cc}, \tilde{V}, l, X)$ is summarized below.

```
1: CombineAll( A_cc, Ṽ, l, X )
2: determine N_Ṽ(l) and N̄_Ṽ(l)
3: if N_Ṽ(l) = ∅        % no compatible neighbours
   then
4:    save l             % save solution
5: else
6:    Ṽ = Ṽ \ N̄_Ṽ(l)          % remove conflicting neighbours
7:    for n ∈ N_Ṽ(l) \ X    % compatible neighb. not visited yet
      do
8:       CombineAll(A_cc, Ṽ \ n, l ∪ n, X)
9:       X = X ∪ n           % mark n as visited
10:   end for
11: end if
```

As an example Tab. 1 shows the complete procedure of CCG for the cc-graph in Fig. 2. It finds four solutions marked by the box in the corresponding column. In the first iteration with $\tilde{V} = V$ and $l = \emptyset$, the neighbour set $N_{\tilde{V}}(l) = V$ is considered as the set of root vertices indicated by the level 0 in Tab. 1. Then in each iteration, $N_{\tilde{V}}(l)$ and $\bar{N}_{\tilde{V}}(l)$ are determined and $\bar{N}_{\tilde{V}}(l)$ is removed from $\tilde{V}$. For each compatible neighbour n, we call `CombineAll` recursively by removing n from $\tilde{V}$ and adding it to l. Once there are no compatible neighbours left we store the solution l. When n is completely processed, we add it to the set of visited vertices X.

4.3. Proof

In this section, we prove the correctness of the CCG algorithm. As previously mentioned, we claim to find all sets of maximally connected compatible vertices.

Lemma 1. *Each vertex must be examined once as the root vertex.*

Proof. We prove Lemma 1 by contradiction. Given the cc-graph

$$\mathbf{A}_{cc} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{5}$$

All maximally connected compatible vertex sets are $\{1, 2\}$ and $\{3\}$. If only one root vertex is chosen, we obtain either $\{1, 2\}$ or $\{3\}$, not both. □

Lemma 2. *Given a root vertex n_0 from $\tilde{V}$,* `CombineAll` $(\tilde{V} \setminus n_0, n_0, \emptyset)$ *finds all solutions containing n_0.*

Proof. `CombineAll` searches the whole set of vertices $\tilde{V}$ for compatible neighbours and removes those that have a conflict with n_0. This is correct because, by definition, we only look for compatible combinations with n_0. Then we consider each solution pair $l = \{n_0, n\}$, $n \in N_{\tilde{V}}(n_0)$, and apply the same combination procedure recursively to l. Due to the fact that each combination of l with an $n \in N_{\tilde{V}}(l)$ initiates a new search branch and we do not remove any compatible neighbours, we are sure that all combinations including l and n are found. □

Lemma 3. `CombineAll` *saves a solution, only when it is maximal.*

Proof. `CombineAll` stops at two different stages: Either when there are no compatible neighbours of the current solution any more (line 3) or when the remaining neighbours belong to X (line 7). The former case implies that the solution l is maximal as there are no neighbours left. In the latter case, the current vertex has already been combined with a vertex $x \in X$ previously. From Lemma 2 we know that `CombineAll` has found all combinations containing the vertex x. Thus we can skip a further search and return without saving. □

Theorem 1. `CombineAll` *finds all maximal solutions.*

Proof. Based on Lemma 2 and Lemma 3, we know that `CombineAll` finds all maximal solutions starting from a root vertex n_0. Lemma 1 states that all vertices have to be examined. Hence the algorithm finds all sets of maximally connected compatible vertices. □

As a result, two useful properties of the algorithm follow immediately. First, each solution found by the algorithm is unique. The algorithm will never find the same solution more than one time. The reason is the use of the set X of excluded vertices. It indicates that the same set of compatible vertices has already been found previously starting from another root vertex. Secondly, it will never happen that one solution l_1 is a subset of another solution l_2. In Fig. 2 and Tab. 1 the algorithm returns the solution $\{v_2, v_3, v_5, v_6\}$, but never its subsets like $\{v_2, v_3, v_5\}, \{v_2, v_3, v_6\}, \ldots$ as additional solutions. The reason is that the algorithm always finds sets of maximally connected compatible vertices.

5. MEASUREMENTS

The synthesis of consistent graphs by using the CCG algorithm was evaluated on real data measured in the LMS audio laboratory at the University Erlangen-Nuremberg a room of size $5.5 \times 6.8 \times 2.6\ m^3$ with $T_{60} \approx 0.8$s. Three sources were modeled by three loudspeakers playing back noise signals to avoid speech pauses.

5 microphones were positioned in the corners and the centroid of a tetrahedra with radius of circumsphere of 70cm and recorded the signals at $f_s = 48$kHz. The loudspeakers were located at a distance of 1.5 to 4.5m from the centroid. Then GCC-PHAT extracted $K_{\max} \in \{5, 10\}$ TDOA candidates per microphone pair on a block length of 8192 samples (170ms). For each of the $K_{\max}$ maxima of GCC-PHAT, a quadratic interpolation is done to obtain TDOA estimates with an accuracy of fractional sampling interval. The synthesis algorithm in Sec. 3 and 4 was applied to extract consistent TDOA (sub)graphs. Note that $M = 5$ microphones lead to maximally $N = 10$ microphone pairs (edges). Hence, the synthesized graphs can have $\bar{N} = 3$ to 10 edges. Finally, the localization by squared-range-difference based LS estimate (SRD-LS) [15] is performed together with the speed estimate criterion of [9] with the reasonable estimated speed of sound range of $[340, 344]$m/s. Both the number of found consistent graphs and the localization results are averaged over 50 blocks (8.5s).

First we present the efficiency of the algorithm. We evaluated the localization on a standard dual-core PC without and with the graph synthesis. The computation time was measured in Matlab. Without the graph synthesis, i.e. we do localization for all $K^N_{\max} \in$

level	n	l	X	$\tilde{V}$	$N_{\tilde{V}}(l)$	save l	$\bar{N}_{\tilde{V}}(l)$	$\tilde{V} \setminus \bar{N}_{\tilde{V}}(l)$	$N_{\tilde{V}}(l) \setminus X$
0		$\emptyset$	$\emptyset$	1,2,3,4,5,6	1,2,3,4,5,6		$\emptyset$	1,2,3,4,5,6	1,2,3,4,5,6
i_a	1	1	$\emptyset$	2,3,4,5,6	2		5,6	2,3,4	2
ii_a	2	1,2	$\emptyset$	3,4	3,4		$\emptyset$	3,4	3,4
iii_a	3	1,2,3	$\emptyset$	4	$\emptyset$	yes			
iii_b	4	1,2,4	3	3	$\emptyset$	yes			
i_b	2	2	1	1,3,4,5,6	1,3,4,6		$\emptyset$	1,3,4,5,6	3,4,6
ii_a	3	2,3	1	1,4,5,6	1,5,6		4	1,5,6	5,6
iii_a	5	2,3,5	1	1,6	6		1	6	6
iv_a	6	2,3,5,6	1	$\emptyset$	$\emptyset$	yes			
iii_b	6	2,3,6	1,5	1,5	5		1	5	$\emptyset$
ii_b	4	2,4	1,3	1,3,5,6	1,6		3,5	1,6	6
iii_a	6	2,4,6	1,3	1	$\emptyset$	yes			
ii_c	6	2,6	1,3,4	1,3,4,5	3,4,5		1	3,4,5	5
iii_a	5	2,5,6	1,3,4	3,4	3		4	3	$\emptyset$
i_c	3	3	1,2	1,2,4,5,6	2,5		4	1,2,5,6	5
ii_a	5	3,5	1,2	1,2,6	2,6		1	2,6	6
iii_a	6	3,5,6	1,2	2	2		$\emptyset$	2	$\emptyset$
i_d	4	4	1,2,3	1,2,3,5,6	2		3,5	1,2,6	$\emptyset$
i_e	5	5	1,2,3,4	1,2,3,4,6	3,6		1,4	2,3,6	6
ii_a	6	5,6	1,2,3,4	2,3	2,3		$\emptyset$	2,3	$\emptyset$
i_f	6	6	1,2,3,4,5	1,2,3,4,5	2,5		1	2,3,4,5	$\emptyset$

Table 1. Details of the algorithm for the cc-graph in Fig. 2

$[5^{10}, 10^{10}]$ possible TDOA combination, we measured 0.957sec for $K_{\max} = 5$ and 13.952sec for $K_{\max} = 10$ in average for one block of 170ms. With the additional synthesis of consistent graphs and a dramatically reduced number of TDOA sets, we measured for one block 0.081sec for $K_{\max} = 5$ and 0.135sec for $K_{\max} = 10$ including the CCG algorithm. This is a factor of 100.

Tab. 2 shows the number of synthesized consistent graphs averaged over all blocks. Due to space limitations, we only present the fully consistent graphs ($\bar{N} = 10$) and the partially consistent graphs ($\bar{N} = 9$).

	$K_{\max} = 5$		$K_{\max} = 10$	
	$\bar{N} = 10$	$\bar{N} = 9$	$\bar{N} = 10$	$\bar{N} = 9$
$\epsilon = 1$	1.2	1.4	1.8	2.8
$\epsilon = 2$	1.2	1.6	3.5	16.7

Table 2. Average number of consistent TDOA graphs

Obviously, a larger value of $K_{\max}$ results in a larger number of TDOA candidates per microphone pair and a larger number of consistent graphs found. According to [8], $K_{\max}$ should not be chosen too small because the direct path TDOAs do not necessarily appear as the largest maxima in GCC-PHAT. Also a larger value of the threshold ϵ in (1), i.e. a more tolerant check of the zero sum condition, leads to a larger number of cFLs and a larger number of consistent graphs. Since it is likely that some microphone pairs do not find the correct TDOAs, the number of partially consistent graphs (e.g. $\bar{N} = 9$) can be larger than that of fully consistent graphs ($\bar{N} = 10$).

Finally, we present the localization errors. The position error in Tab. 3 represents the Euclidean distance in cm from the estimated position to the closest loudspeaker. It is averaged over all sources and all blocks. The loudspeakers have a membrane of 15cm diameter.

The small localization errors of fully consistent graphs show that the estimated positions are more realiable the more TDOAs we use. Due to the loudspeaker size, the correct positions of the sources cannot be optimally defined and hence the position errors are in a reasonable range. If TDOAs are missing, i.e. partially consistent graphs are synthesized, we obtain less correct position estimations. As a consequence of the relaxation of the cyclic sum condition, higher position errors are observed from larger ϵ values. Thus, one has to choose $K_{\max}$ and ϵ carefully in order to keep all consistent graphs (fully and partially) of the corresponding sources and to obtain small localization errors.

	$K_{\max} = 5$		$K_{\max} = 10$	
	$\bar{N} = 10$	$\bar{N} = 9$	$\bar{N} = 10$	$\bar{N} = 9$
$\epsilon = 1$	8.5	8.0	8.2	18.3
$\epsilon = 2$	8.5	21.5	12.4	162.2

Table 3. Average position error in cm over all blocks

6. CONCLUSIONS

This paper presents a new algorithm to synthesize fully and partially consistent graphs, given the TDOA estimates from a microphone array. As experiments show, it allows a fast and reliable TDOA assignment for some localization in a multi-source reverberant environment.

REFERENCES

[1] Carter, G. Clifford, "Coherence and Time Delay Estimation," *Proceedings of the IEEE*, vol. 75, no. 2, pp. 236–255, 1987.

[2] Brandstein, Michael and Ward, Darren, *Microphone Arrays: Signal Processing Techniques and Applications*, Springer Verlag, 2001.

[3] Knapp, Charles H. and Carter, G. Clifford, "The Generalized Correlation Method for Estimation of Time Delay," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 320–327, 1976.

[4] Nejad, Mohamad Hesam Mahmodi and Mahmoodi, Davood and Zohroudi, Salehe, "Multiple Speaker Localization in a Smart Room," in *Int. Conf. on Multimedia and Signal Processing*, May 2011, pp. 319–323.

[5] Lombard, Anthony and Buchner, Herbert and Kellermann, Walter, "Multidimensional Localization of Multiple Sound Sources using Blind Adaptive MIMO System Identification," in *Proc. IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, 2006, pp. 7–12.

[6] Loesch, Benedikt and Uhlich, Stefan and Yang, Bin, "Multidimensional Localization of Multiple Sound Sources using Frequency Domain ICA and an Extended State Coherence Transform," in *IEEE Workshop on Statistical Signal Processing*, Aug. 2009, pp. 677–680.

[7] "http://www.iss.uni-stuttgart.de/download," .

[8] Scheuing, Jan and Yang, Bin, "Disambiguation of TDOA Estimation for Multiple Sources in Reverberant Environments," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 16, no. 8, pp. 1479–1489, Nov. 2008.

[9] Hu, Jwu-Sheng and Yang, Chia-Hsin, "Estimation of Sound Source Number and Directions under a Multisource Reverberant Environment," *EURASIP Journal on Advances in Signal Processing*, vol. 2010, no. 1, pp. 1–14, 2010.

[10] Kreißig, Martin and Yang, Bin, "An Efficient Algorithm for the Synthesis of Fully Consistent Graphs," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2012, pp. 2653 – 2656.

[11] Yang, Bin and Kreißig, Martin, "An Introduction to Consistent Graphs and Their Signal Processing Applications," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2011, pp. 2740–2743.

[12] Kreißig, Martin and Yang, Bin, "Efficient Synthesis of Consistent Graphs," in *Proc. EURASIP European Signal Processing Conf.*, 2010, pp. 1364–1368.

[13] Russell, Stuart and Norvig, Peter, *Artificial Intelligence: A modern approach*, Pearson Education, 2003.

[14] Bron, Coen and Kerbosch, Joep, "Algorithm 457: Finding All Cliques of an Undirected Graph," *Commun. ACM*, vol. 16, no. 9, pp. 575–577, Sept. 1973.

[15] Beck, Amir and Stoica, Petre and Li, Jian, "Exact and Approximate Solutions of Source Localization Problems," *IEEE Trans. on Signal Processing*, vol. 56, no. 5, pp. 1770–1778, 2008.