FAST IMPLEMENTATION OF TWO-DIMENSIONAL APES AND CAPON SPECTRAL ESTIMATORS

Erik G. Larsson and Petre Stoica

Dept. of Systems and Control, Uppsala University
P.O. Box 27, SE-751 03 Uppsala, Sweden.
Email: egl@syscon.uu.se

ABSTRACT

The matched-filterbank spectral estimators APES and CAPON have recently received considerable attention in a number of applications. Unfortunately, their computational complexity tends to limit their usage in several cases – a problem that has previously been addressed by different authors. In this paper, we introduce a novel method to the computation of the 1D and 2D APES and CAPON spectra, which is considerably faster than all existing techniques. Numerical examples are provided to demonstrate the application of APES to synthetic aperture radar (SAR) imaging, and to illustrate the reduction in computational complexity provided by our implementation.

1. THE APES AND CAPON ESTIMATORS

Consider the problem of estimating the amplitude spectrum of a complex-valued two-dimensional signal $\{\boldsymbol{X}_{n,\bar{n}}\}_{n=0,\bar{n}=0}^{N-1,\bar{N}-1}$ where $N \times \bar{N}$ is the dimension of the data matrix. We require $N \geq 1$ and $\bar{N} > 1$, where the special case $N = 1$ will be referred to as the 1D case.

The APES and CAPON estimators [7, 1], which we next describe, are two powerful techniques to solve this problem. We start by introducing the user parameters M and $\bar{M}$, which are usually referred to as *filter lengths*. Increasing M and $\bar{M}$ typically increases the spectral resolution at the cost of reducing the statistical stability of the spectral estimates. Let $L \triangleq N - M + 1$ and $\bar{L} \triangleq \bar{N} - \bar{M} + 1$ (in the 1D case we have $M = N = L = 1$), and arrange the samples $\boldsymbol{X}_{n,\bar{n}}$ in a matrix as

$$\boldsymbol{Y} \triangleq \begin{bmatrix} \boldsymbol{y}_{0,0} & \cdots & \boldsymbol{y}_{L-1,0} & \boldsymbol{y}_{0,1} & \cdots & \boldsymbol{y}_{L-1,\bar{L}-1} \end{bmatrix} \quad (1)$$

where

$$\boldsymbol{y}_{l,\bar{l}} \triangleq \mathrm{vec}\left(\begin{bmatrix} \boldsymbol{X}_{l,\bar{l}} & \cdots & \boldsymbol{X}_{l,\bar{l}+\bar{M}-1} \\ \vdots & \ddots & \vdots \\ \boldsymbol{X}_{l+M-1,\bar{l}} & \cdots & \boldsymbol{X}_{l+M-1,\bar{l}+\bar{M}-1} \end{bmatrix} \right) \quad (2)$$

and where $\mathrm{vec}(\cdot)$ denotes the operation of stacking the columns of a matrix on top of each other.

THIS WORK WAS PARTLY SUPPORTED BY THE SWEDISH FOUNDATION FOR STRATEGIC RESEARCH (SSF).

Define the *forward sample covariance matrix*

$$\hat{\boldsymbol{R}}_f \triangleq \frac{1}{L\bar{L}} \sum_{l=0}^{L-1} \sum_{\bar{l}=0}^{\bar{L}-1} \boldsymbol{y}_{l,\bar{l}} \boldsymbol{y}_{l,\bar{l}}^* = \frac{1}{L\bar{L}} \boldsymbol{Y}\boldsymbol{Y}^* \in \mathbb{C}^{M\bar{M} \times M\bar{M}} \quad (3)$$

and the *forward-backward sample covariance matrix*

$$\boldsymbol{R} \triangleq \frac{1}{2}(\hat{\boldsymbol{R}}_f + \boldsymbol{\Psi}_{M\bar{M}} \hat{\boldsymbol{R}}_f^T \boldsymbol{\Psi}_{M\bar{M}}) \quad (4)$$

where $\boldsymbol{\Psi}_P$ denotes a $P \times P$ matrix with ones on its anti-diagonal and zeros everywhere else, $(\cdot)^T$ denotes the transpose and $(\cdot)^*$ denotes the conjugate transpose.[1]

Introduce, for arbitrary integers P and $\bar{P}$, the 2D Fourier vector

$$\begin{aligned} \boldsymbol{a}_{P,\bar{P}}(\omega,\bar{\omega}) \triangleq & \begin{bmatrix} 1 & e^{i\omega} & \cdots & e^{i(P-1)\omega} \end{bmatrix}^T \\ & \otimes \begin{bmatrix} 1 & e^{i\bar{\omega}} & \cdots & e^{i(\bar{P}-1)\bar{\omega}} \end{bmatrix}^T \end{aligned} \quad (5)$$

where $\otimes$ denotes the Kronecker product. Define the 2D Fourier transforms

$$\begin{aligned} \bar{\boldsymbol{g}}(\omega,\bar{\omega}) &\triangleq \frac{1}{L\bar{L}} \boldsymbol{Y}\boldsymbol{a}_{L,\bar{L}}(-\omega,-\bar{\omega}) = \frac{1}{L\bar{L}} \sum_{l=0}^{L-1} \sum_{\bar{l}=0}^{\bar{L}-1} \boldsymbol{y}_{l,\bar{l}} e^{-i(\omega l + \bar{\omega}\bar{l})} \\ \tilde{\boldsymbol{g}}(\omega,\bar{\omega}) &\triangleq \frac{1}{L\bar{L}} \tilde{\boldsymbol{Y}}\boldsymbol{a}_{L,\bar{L}}(-\omega,-\bar{\omega}) \end{aligned} \quad (6)$$

where $\boldsymbol{Y}$ is defined in (1) and (2), and $\tilde{\boldsymbol{Y}} \triangleq \boldsymbol{\Psi}_{M\bar{M}} \boldsymbol{Y}^c \boldsymbol{\Psi}_{L\bar{L}}$. Here $(\cdot)^c$ denotes the complex conjugate.

The *amplitude spectrum* CAPON (ASC) is defined by

$$\phi_{ASC}(\omega,\bar{\omega}) \triangleq \frac{\boldsymbol{a}_{M,\bar{M}}^*(\omega,\bar{\omega}) \hat{\boldsymbol{R}}^{-1} \bar{\boldsymbol{g}}(\omega,\bar{\omega})}{\boldsymbol{a}_{M,\bar{M}}^*(\omega,\bar{\omega}) \hat{\boldsymbol{R}}^{-1} \boldsymbol{a}_{M,\bar{M}}(\omega,\bar{\omega})} \quad (7)$$

and the *power spectrum* CAPON (PSC) is given by

$$\phi_{PSC}(\omega,\bar{\omega}) \triangleq \frac{1}{\boldsymbol{a}_{M,\bar{M}}^*(\omega,\bar{\omega}) \hat{\boldsymbol{R}}^{-1} \boldsymbol{a}_{M,\bar{M}}(\omega,\bar{\omega})} \quad (8)$$

The APES estimator is defined by

$$\phi_{APES}(\omega,\bar{\omega}) \triangleq \frac{\boldsymbol{a}_{M,\bar{M}}^*(\omega,\bar{\omega}) \hat{\boldsymbol{Q}}^{-1}(\omega,\bar{\omega}) \bar{\boldsymbol{g}}(\omega,\bar{\omega})}{\boldsymbol{a}_{M,\bar{M}}^*(\omega,\bar{\omega}) \hat{\boldsymbol{Q}}^{-1}(\omega,\bar{\omega}) \boldsymbol{a}_{M,\bar{M}}(\omega,\bar{\omega})} \quad (9)$$

where

$$\hat{\boldsymbol{Q}}(\omega,\bar{\omega}) \triangleq \hat{\boldsymbol{R}} - \frac{1}{2} \begin{bmatrix} \bar{\boldsymbol{g}}(\omega,\bar{\omega}) & \tilde{\boldsymbol{g}}(\omega,\bar{\omega}) \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{g}}^*(\omega,\bar{\omega}) \\ \tilde{\boldsymbol{g}}^*(\omega,\bar{\omega}) \end{bmatrix} \quad (10)$$

[1] In this work we treat only the forward-backward estimators.

2. KEY RESULTS

The starting point for deriving the fast implementations is the following observation.

Lemma 1 *Let $\hat{\boldsymbol{R}}^{-1/2}$ denote the inverse of the Cholesky factor of $\hat{\boldsymbol{R}}$ and let $\{\boldsymbol{r}_m\}_{m=1}^{M\bar{M}}$ be the columns of $\hat{\boldsymbol{R}}^{-1/2}$. Consider the following trigonometric polynomials:*

$$p_{ij}(\omega,\bar{\omega}) \triangleq \sum_{m=1}^{M\bar{M}} p_i^{(m)*}(\omega,\bar{\omega}) p_j^{(m)}(\omega,\bar{\omega}) = \sum_{m=1}^{M\bar{M}} p_{ij}^{(m)}(\omega,\bar{\omega}) \tag{11}$$

where

$$\begin{aligned} p_1^{(m)}(\omega,\bar{\omega}) &\triangleq \boldsymbol{r}_m^* \boldsymbol{a}_{M,\bar{M}}(\omega,\bar{\omega}) \\ p_2^{(m)}(\omega,\bar{\omega}) &\triangleq \frac{1}{L\bar{L}} \boldsymbol{r}_m^* \boldsymbol{Y} \boldsymbol{a}_{L,\bar{L}}(-\omega,-\bar{\omega}) \\ p_3^{(m)}(\omega,\bar{\omega}) &\triangleq \frac{1}{L\bar{L}} \boldsymbol{r}_m^* \tilde{\boldsymbol{Y}} \boldsymbol{a}_{L,\bar{L}}(-\omega,-\bar{\omega}) \\ p_{ij}^{(m)}(\omega,\bar{\omega}) &\triangleq p_i^{(m)*}(\omega,\bar{\omega}) p_j^{(m)}(\omega,\bar{\omega}) \end{aligned} \tag{12}$$

and let

$$\boldsymbol{\Sigma}(\omega,\bar{\omega}) \triangleq \frac{1}{2}\begin{bmatrix} p_{22}(\omega,\bar{\omega})-2 & p_{23}(\omega,\bar{\omega}) \\ p_{23}^*(\omega,\bar{\omega}) & p_{22}(\omega,\bar{\omega})-2 \end{bmatrix}$$

Then $p_{13}^*(\omega,\bar{\omega}) = p_{12}^*(\omega,\bar{\omega}) e^{-i((N-1)\omega+(\bar{N}-1)\bar{\omega})}$, and furthermore, the ASC, PSC and APES spectra can be expressed as

$$\phi_{ASC}(\omega,\bar{\omega}) = \frac{p_{12}(\omega,\bar{\omega})}{p_{11}(\omega,\bar{\omega})}, \quad \phi_{PSC}(\omega,\bar{\omega}) = \frac{1}{p_{11}(\omega,\bar{\omega})} \tag{13}$$

and $\phi_{APES}(\omega,\bar{\omega}) =$

$$\frac{p_{12}(\omega,\bar{\omega}) - \frac{1}{2}\begin{bmatrix} p_{12}(\omega,\bar{\omega}) & p_{13}(\omega,\bar{\omega}) \end{bmatrix} \boldsymbol{\Sigma}^{-1}(\omega,\bar{\omega}) \begin{bmatrix} p_{22}(\omega,\bar{\omega}) \\ p_{23}^*(\omega,\bar{\omega}) \end{bmatrix}}{p_{11}(\omega,\bar{\omega}) - \frac{1}{2}\begin{bmatrix} p_{12}(\omega,\bar{\omega}) & p_{13}(\omega,\bar{\omega}) \end{bmatrix} \boldsymbol{\Sigma}^{-1}(\omega,\bar{\omega}) \begin{bmatrix} p_{12}^*(\omega,\bar{\omega}) \\ p_{13}^*(\omega,\bar{\omega}) \end{bmatrix}} \tag{14}$$

Proof: See [6].

The technique proposed by Liu et al. in [8] for the computation of 1D and 2D APES and ASC evaluates the polynomials in Lemma 1 by using the fact that once $\hat{\boldsymbol{R}}^{-1/2}$, $\hat{\boldsymbol{R}}^{-1/2}\boldsymbol{Y}$ and $\hat{\boldsymbol{R}}^{-1/2}\tilde{\boldsymbol{Y}}$ were computed, the trigonometric polynomial vectors $\hat{\boldsymbol{R}}^{-1/2}\boldsymbol{a}_{M,\bar{M}}(\omega,\bar{\omega})$, $\hat{\boldsymbol{R}}^{-1/2}\boldsymbol{Y}\boldsymbol{a}_{L,\bar{L}}(-\omega,-\bar{\omega})$ and $\hat{\boldsymbol{R}}^{-1/2}\tilde{\boldsymbol{Y}}\boldsymbol{a}_{L,\bar{L}}(-\omega,-\bar{\omega})$ can be evaluated using $M\bar{M}$ 2D FFTs of size $K\times\bar{K}$. In [2], Ekman et al. use the same technique for the evaluation of ASC, with the exception that the Cholesky factor $\hat{\boldsymbol{R}}^{-1/2}$ is approximated with a structured matrix obtained by fitting an autoregressive model to the data. This approximation technique provides efficient means for computing $\hat{\boldsymbol{R}}^{-1/2}$, which is sometimes claimed to be a bottleneck in the computation of the spectrum (at least for the 1D case). However, it has turned out that this technique is not applicable to the computation of the APES spectrum.

The key observation that leads to our fast implementation of APES, ASC and PSC is as follows: instead of computing the values of the $M\bar{M}$ polynomials $p_i^{(m)}(\omega,\bar{\omega})$ at $K\times\bar{K}$ points, we compute the coefficients of the polynomials $p_{ij}(\omega,\bar{\omega})$ in (11) and upon completion of this computation, each polynomial is evaluated at $K\times\bar{K}$ points using a single 2D FFT. The computation of the polynomial coefficients is performed by accumulating the contributions from $p_{ij}^{(m)}(\omega,\bar{\omega})$ in the sum in (11).

To exploit this idea optimally, we combine it with some relevant results on fast computation, which we state next.

Lemma 2 *Let $\boldsymbol{u}$ be an arbitrary vector of length $M\bar{M}$. Then the products $\boldsymbol{u}^*\boldsymbol{Y}$ and $\boldsymbol{u}^*\tilde{\boldsymbol{Y}}$, where $\boldsymbol{Y}$ and $\tilde{\boldsymbol{Y}}$ are defined above, can be computed in $O\big(N\bar{N}\log(N\bar{N})\big)$ floating point operations. The same is true for products of the type $\boldsymbol{Y}\boldsymbol{u}$ and $\tilde{\boldsymbol{Y}}\boldsymbol{u}$, where $\boldsymbol{u}$ is a vector of length $L\bar{L}$.*

Proof: See [6].

The operation count in the lemma should be compared with direct evaluation of the vector-matrix product $\boldsymbol{u}^*\boldsymbol{Y}$, which would require $O(M\bar{M}N\bar{N})$ operations. Note that one, but not the only, useful consequence of Lemma 2 is that $\hat{\boldsymbol{R}}$ can be computed in $O\big(M\bar{M}N\bar{N}\log(N\bar{N})\big)$ operations. To see this, note that row k of $\hat{\boldsymbol{R}}$ equals $\frac{1}{L\bar{L}}\boldsymbol{u}_k^*\boldsymbol{Y}^*$ where $\boldsymbol{u}_k^*$ is the kth row of $\boldsymbol{Y}$, and apply Lemma 2 $M\bar{M}$ times (once for each row). This should be compared with direct evaluation of (3) etc., which requires $O(M^2\bar{M}^2N\bar{N})$ operations.

Lemma 3 *Let $p(\omega,\bar{\omega}) = p_{n_p-1,\bar{n}_p-1}e^{i\left((n_p-1)\omega+(\bar{n}_p-1)\bar{\omega}\right)} + \cdots + p_{n_p-1,0}e^{i(n_p-1)\omega} + \cdots + p_{0,\bar{n}_p-1}e^{i(\bar{n}_p-1)\bar{\omega}} + \cdots + p_{0,0}$, and similarly $q(\omega,\bar{\omega})$ be two arbitrary 2D trigonometric polynomials. Then the $(n_p+n_q-1)(\bar{n}_p+\bar{n}_q-1)$ coefficients of the polynomial $r(\omega,\bar{\omega}) = p^*(\omega,\bar{\omega})q(\omega,\bar{\omega})$ as well as the coefficients of the polynomials $\tilde{r}(\omega,\bar{\omega}) = p^*(\omega,\bar{\omega})q(-\omega,-\bar{\omega})$ and $\hat{r}(\omega,\bar{\omega}) = p^*(-\omega,-\bar{\omega})q(-\omega,-\bar{\omega})$ can be computed in $O\Big((n_p+n_q)(\bar{n}_p+\bar{n}_q)\log\big((n_p+n_q)(\bar{n}_p+\bar{n}_q)\big)\Big)$ operations.*

Proof: See [6].

As a remark, note that direct computation of the polynomial coefficients would require $O\big((n_p+n_q)^2(\bar{n}_p+\bar{n}_q)^2\big)$ operations.

Lemma 4 *By using the Generalized Schur algorithm of [5] together with a factorization result in [6, App. C], the columns $\{\boldsymbol{r}_k\}_{k=1}^{M\bar{M}}$ of $\hat{\boldsymbol{R}}^{-1/2}$ can be computed* from the data *in $O\big(MN\bar{N}\log(N\bar{N}) + M^2\bar{M}^2N\big)$ floating point operations ($O\big(\bar{N}\log(\bar{N}) + \bar{M}^2\big)$ in the 1D case). Under the further assumption (which will be seen to be valid in our application) that each $\boldsymbol{r}_k$ is used directly after its computation, i.e. the whole Cholesky factor is not stored, running the algorithm in [6, App. C] requires $O(M\bar{M}N)$ bits of memory ($O(\bar{M})$ in the 1D case).*

Proof: See [6].

The operation count in the lemma should be compared to computation of $\hat{\boldsymbol{R}}$ via (3) etc. followed by standard Cholesky factorization and inversion [3] which together require $O(M^2\bar{M}^2N\bar{N})$ operations ($O(\bar{M}^2\bar{N})$ in the 1D case). The method of Lemma 4 is usually significantly faster than inversion and classical Cholesky factorization, provided that

$M, \bar{M}$ are relatively large (which in effect is the case of interest in practice). As a further remark, note that storage of the whole Cholesky factor would require $O(M^2\bar{M}^2)$ bits of memory which is typically much more than the storage requirement of the algorithm in [6]. The low memory requirement of the algorithm is of value for both possible hardware implementations and off-line data analysis applications.

Finally, we stress that the lemma gives the inverse of the exact Cholesky factor, which should not be confused with the approximations in [4, 2] (the latter are also relatively computationally efficient but they operate on a structured Toeplitz approximation of $\hat{\boldsymbol{R}}$).

3. FAST IMPLEMENTATION OF APES, ASC AND PSC

1. Decide whether to use classical Cholesky factorization and inversion or the fast technique of Lemma 4 to compute $\hat{\boldsymbol{R}}^{-1/2}$. If the fast factorization method is chosen, perform the initialization described in [6, App. C]. Otherwise, first compute $\hat{\boldsymbol{R}}$ as indicated in the remark after Lemma 2 and thereafter compute $\hat{\boldsymbol{R}}^{-1/2}$ by a direct method [3].
2. Perform the following steps for $m = 1, \ldots, M\bar{M}$:
   (a) Obtain the mth column $\boldsymbol{r}_m$ of $\hat{\boldsymbol{R}}^{-1/2}$. If the classical Cholesky factorization is used, $\boldsymbol{r}_m$ is already precomputed in Step 1 above – otherwise perform one iteration of the generalized Schur algorithm described in [6, App. C].
   (b) Compute $\boldsymbol{r}_m^*\boldsymbol{Y}$ and $\boldsymbol{r}_m^*\tilde{\boldsymbol{Y}}$ by using Lemma 2.
   (c) Compute the coefficients of the polynomials $p_i^{(m)}(\omega, \bar{\omega})$, $i = 1, 2, 3$, above by using the results of the substeps (a)-(b) above.
   (d) Compute the coefficients of the polynomials $p_{ij}^{(m)}(\omega, \bar{\omega})$ above by using Lemma 3. These coefficients are summed up to obtain the coefficients of the polynomials $p_{ij}(\omega, \bar{\omega})$ in (11).
3. Evaluate the polynomials $p_{ij}(\omega, \bar{\omega})$ at $K \times \bar{K}$ points by using a single 2D FFT (per polynomial) and compute the ASC, PSC or APES spectrum according to (13) or (14).

4. NUMERICAL EXAMPLES

Example 1: Complexity comparison for the 1D case ($N = M = L = 1$). Random data with varying length $\bar{N}$ and the filter length $\bar{M} = \bar{N}/2$ are used, and the spectrum is evaluated on a frequency grid with $\bar{K} = 4096$ points. The comparison is carried out by measuring the number of floating point operations used by a Matlab implementation. Figure 1 shows the results. We compare our implementation of APES with the technique suggested by Liu et al. in [8]. It can be observed that our algorithm requires around 5%-10% of the floating point operations needed by the method of [8]. We stress that our algorithm, as well as the method of [8], computes the exact spectrum. In Figure 1 we also show a comparison of our implementation of ASC with the (approximate) technique suggested by Ekman et al. in [2]. It is clear that also in this case our technique is considerably faster, despite the fact that [2] computes only a (fast) approximation of ASC.

Example 2: Complexity comparison for the 2D case. A (randomly generated) square data matrix of varying size $N = \bar{N}$ is considered. The filter lengths are $M = N/2$ and $\bar{M} = \bar{N}/2$ for $N = \bar{N} \leq 64$, and $M = \bar{M} = 32$ for $N = \bar{N} \geq 64$. The spectrum is evaluated on a grid with $K = \bar{K} = 1024$ points. Figure 2 shows the results. We compare the computation of the APES spectrum using the technique of Liu et al. [8] with the algorithm proposed in this paper. For a data matrix of size 64×64, our technique is about 50 times faster.

In [4], Jakobsson et al. propose an algorithm for the fast computation of the PSC spectrum. Note that the computation of PSC is simpler than that of APES or ASC and hence the former can presumably be organized in a more efficient manner than the latter (the fast implementation of the latter has been the main objective of this paper). Furthermore, in contrast to our technique, the algorithm of [4] does not compute the exact PSC as defined herein, but an approximation of it based on a structured (block-Toeplitz) version of the sample covariance matrix. Despite these facts, we observe from Figure 2 that for sufficiently large data sets (which may be the case of interest in practice) the effort for computing the PSC using our method is somewhat lower than that needed by the fast PSC algorithm of [4].

The figures indicate that the new implementation improves significantly over the available techniques. Most likely the program codes for the implementation proposed in this paper as well as for the implementations in [8, 2] can be "polished" to reduce the constant factor in the corresponding operation counts. The important point however is that the difference in computational complexity between these implementations increases with increasing data sample lengths, and with increasing $\frac{K}{N}$ and $\frac{\bar{K}}{\bar{N}}$. In particular, our implementation is practically insensitive to the density of the frequency grid $(K, \bar{K})$, unlike those in [8, 2].

Exampe 3: Application of APES to SAR imaging. We consider a 100×100 matrix of phase history data of an object (at 0° azimuth angle) generated by XPATCH, a high frequency electromagnetic scattering prediction code for complex 3D-objects. A photo of the object under consideration (taken at 45° azimuth angle) is shown in Figure 3. Figures 4 and 5 show the SAR image obtained by applying APES with $M \times \bar{M}$ equal to 8×8 and 30×30, respectively. In all figures, the spectrum was evaluated on a 1024×1024 grid. From Figures 4 and 5, it is clear that the increase in filter lengths improves the quality of the SAR image significantly. The latter observation is particularly important in providing the motivation for this paper, as the computation of the image in Figure 5 using the existing techniques for the evaluation of the APES spectrum (such as that in [8]) would have been computationally extremely expensive.

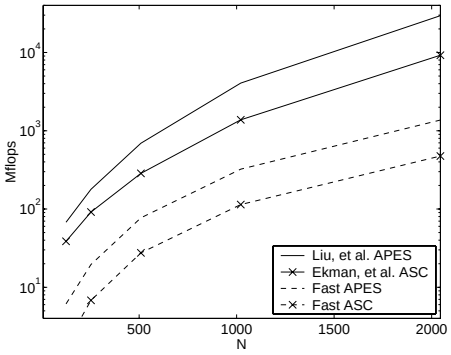

Figure 1: Complexity comparison in the 1D case. In Figure 1 and 2, "Fast APES, ASC and PSC" refers to our new computational method.

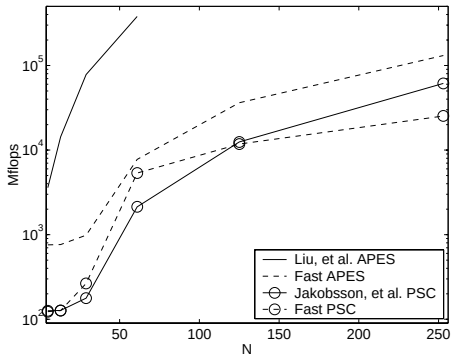

Figure 2: Complexity comparison in the 2D case. The missing points correspond to computations which would have taken unreasonably long time to perform on our workstation.

Figure 3: Photograph of the object (taken at 45° azimuth angle).

Figure 4: SAR image obtained via APES with $M = \bar{M} = 8$.

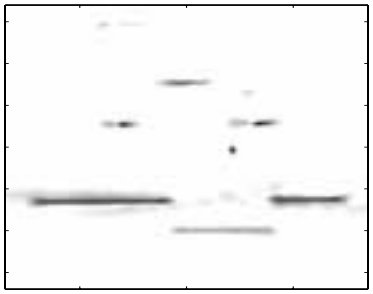

Figure 5: SAR image obtained via APES with $M = \bar{M} = 30$.

5. ACKNOWLEDGEMENT

The authors are grateful to A. Jakobsson and T. Ekman for providing the Matlab code of their 2D PSC and 1D ASC implementations, which was used for comparison purposes, and to Prof. J. Li for providing the SAR data.

6. REFERENCES

[1] J. Capon, "Maximum-likelihood spectral estimation," in *Nonlinear Methods of Spectral Analysis* (S. Haykin, ed.), Springer-Verlag, 1983.

[2] T. Ekman, A. Jakobsson, and P. Stoica, "On efficient implementation of the CAPON algorithm," in *Proc. of European Signal Processing Conference (EUSIPCO)*, (Tampere, Finland), 2000.

[3] G. H. Golub and C. F. van Loan, *Matrix Computations*. Maryland, USA: The Johns Hopkins University Press, 1989.

[4] A. Jakobsson, S. L. Marple, Jr., and P. Stoica, "Two-dimensional CAPON spectral analysis," *IEEE Transactions on Signal Processing*, vol. 48, pp. 2651–2661, Sept. 2000.

[5] T. Kailath and A. Sayed, *Fast Reliable Algorithms for Matrices with Structure*. Philadelphia, USA: SIAM, 1999.

[6] E. G. Larsson and P. Stoica, "Efficient implementation of two-dimensional CAPON and APES for spectral estimation," *Multidimensional Systems and Signal Processing*, 2000. Submitted.

[7] J. Li and P. Stoica, "An adaptive filtering approach to spectral estimation and SAR imaging," *IEEE Transactions on Signal Processing*, vol. 44, pp. 1469–1484, June 1996.

[8] Z.-S. Liu, H. Li, and J. Li, "Efficient implementation of CAPON and APES for spectral estimation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, pp. 1314–1319, Oct. 1998.