Back to the Future: Time Reversal Imaging

18-551 Final Report

Fall 2009, Group 10

David DeBaun (ddebaun@andrew.cmu.edu)
Margaret Sheng (msheng@andrew.cmu.edu)
Alan Zhu (axz@andrew.cmu.edu)

Table of Contents

The Problem

The problem we are addressing in this project is that the conventional methods for locating a target in a cluttered environment have low accuracy rates. To be precise, matched field processing (MFP) which is Green's function integrated, is too expensive to run and is extremely sensitive to accuracy of the environmental data [2]. The answer is then to use the TRAIC (Time Reversal Imaging by Adaptive Interference Cancelling) [2] algorithm that boosts accuracy of target returns and attenuates non-target returns.

Novelty

The novelty of this project is that this is the first time the time reversal algorithm has been used, attempted, or implemented with DSP hardware. There are no previous relevant 18-551 projects and it also contributes to popular new research area, continuing the work of Yuanwei Jin and Professor Jose Moura at Carnegie Mellon University.

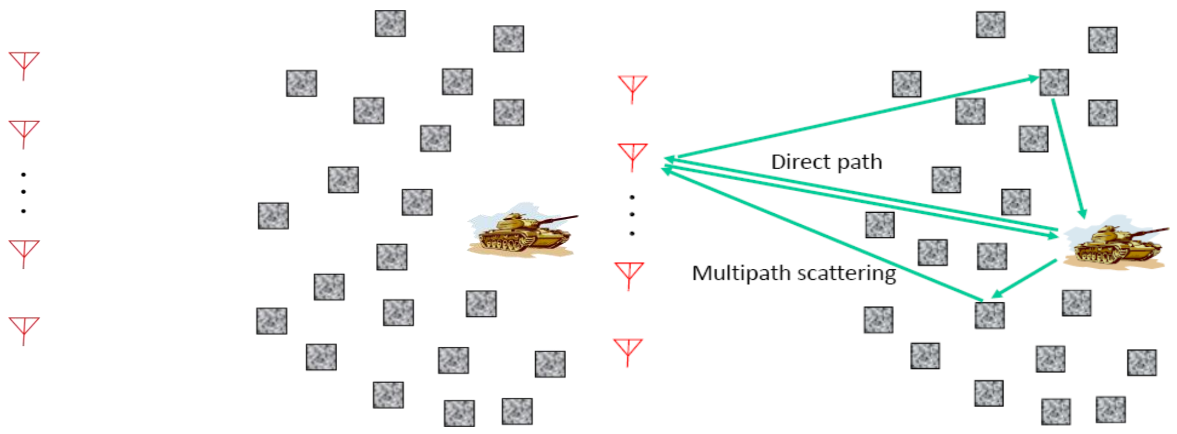
Solution

The TRAIC algorithm proposed by Moura and Jin in their paper utilizes multipath scattering to increase the accuracy in target imaging. The steps as proposed by the algorithm include: clutter channel probing; time reversal waveform reshaping; time reversal target focusing; and the final step, image formation using beamforming and triangulation.

What We Did

We actually follow the algorithm below exactly except for two major caveats. The first is that the data has already been collected for us from a previous experimental setup. Thus the initial step of obtaining the environment response as well as the environment plus target response has already been done for us in our implementation. The second is that this information has already been Fourier Transformed, which allows us to implement the entire algorithm from a mathematic standpoint in the frequency domain, doing phase conjugation in place of actual signal re-transmittance. In our algorithm below we note the frequency domain mathematical equations used in our implementation.

Setup



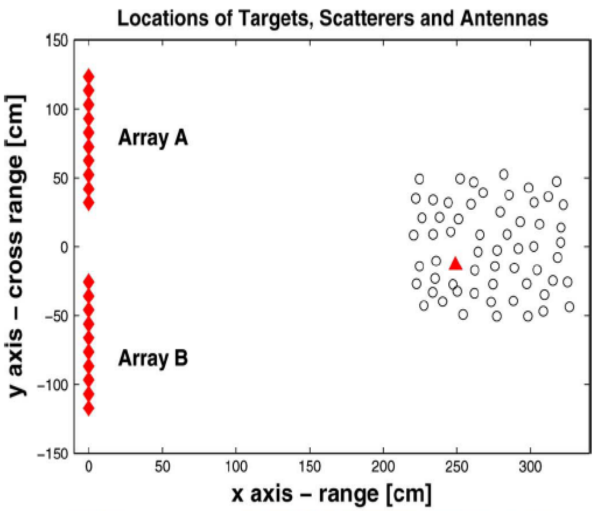
[Fig. 1] A target present in an area rich with scatter components. Note that when our detection antenna sends a signal into the environment, it receives both the direct path reflection and the reflection off the scatters resulting in multipath. Ref. [2]

First, we need to define the size our data.

We have two arrays of antennas, each array having ten antennas. We have each antenna transmit a series of bursts of narrow band pulses where each burst is a sequence of pulses stepped in frequency from pulse to pulse by a fixed step size. The bandwidth is from 4GHz to 6GHz, stepped by 80 MHz for a total of 26 pulses for each antenna. Thus we are coarsely sampling the wideband signal spectrum of our bandwidth at these frequencies.

With reference to figure 4, we send the pulses from each antenna in array A to some region that we are interested in, and receive a reflected signal at array B. Thus for each pulse sent from each antenna in A, we have a received signal at each antenna in array B. Doing this for each of the ten antennas in array A, we have 26x10 pulses sent by array A in total, and 26x10x10 received waves by array B in total, since each antenna in B receives something from each antenna in A.

Since the received signals at array B represent the response we receive from the environment (with or without target), we have a matrix of dimensions 10x10x26 complex numbers, since for each received signal we have amplitude and phase.



[Fig. 2] Overhead view of the physical setup generating the data sets. Ref. [3]

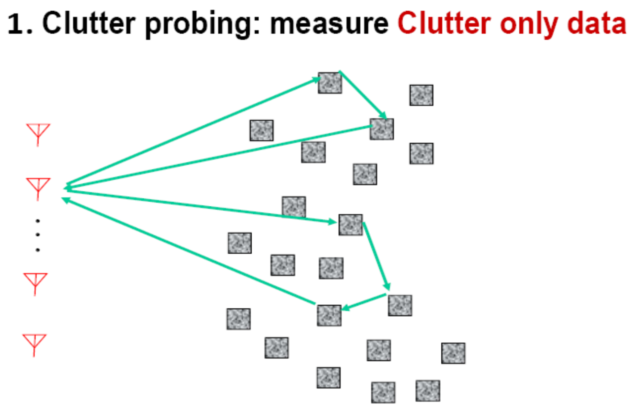
Programmatically, since we represent a complex number with two doubles, this equates to a matrix of dimensions 10x10x26 where each entry is two doubles.

We chose such a bandwidth based on the experimental conditions that produced our experimental data. The magnitude of the data set for the original experiment at 200 frequencies (10 MHz step size) would be relatively large (200x10x10) although of course more accurate because of increased frequency resolution. Because the computational and memory costs will contribute to a substantial increase in processing time and complexity in implementation we use the aforementioned step size of 80MHz and thus for our test sets, we will simply use only a total of 26 frequencies.

The Algorithm

The complete algorithm we implement is called Time Reversal Adaptive Interference Cancellation with Time Reversal Beam Forming, TRAIC+TRBF. The breakdown of the algorithm is described by the following steps:

1. Get the clutter. As mentioned beforehand we have two instances of data that we are concerned about. One is the frequency response of the environment alone, and one is the frequency response of the environment with the target in it. We define the environment as the established initial area we want to monitor. When there is a target in this high-scattering environment we want to be able to detect that target and image the area with relative accuracy. Based on the research we have done and the feasibility of our project we assume that the area is relatively static.



[Fig. 3] The first step in the algorithm: getting the environment response. Ref. [2]

   Our first step is to get the frequency response of the clutter environment when the target is not present. Termed 'clutter probing,' we do this to implement the direct subtraction later on.

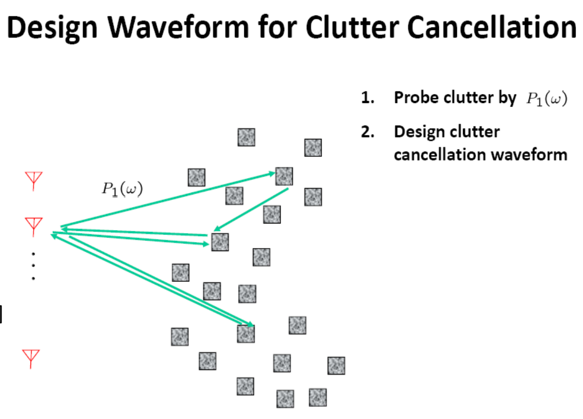
   As mentioned early, we send pulses at 20 frequencies from array A to array B. For each frequency we have on pulse sent from each antenna in array A, and some received response at each antenna in array B for each antenna in array A. This gives us a 10x10 matrix for each frequency. Over 20 frequencies, we use a 10x10x20 matrix to characterize our total response.

   - Clutter only data matrix (from array A to array B)

$$[\mathbf{K}_c(\omega)]_{ij} = k_c(\omega; B_i \leftarrow A_j) \qquad i,j = 1,\cdots,P$$

2. Use clutter response to design a filter that cancels the energy from the clutter. Note that the figure below references a *S(w_q)* which represents the noise component that is however not within the scope of our project. The formula for the optimal reshaping filter

[Fig. 4] Step 2: Design waveform to cancel clutter components. Ref. [2]


Design Waveform for Clutter Cancellation
1. Probe clutter by $P_1(\omega)$
2. Design clutter cancellation waveform
$P_1(\omega)$

follows (See paper reference [3] for complete mathematical proof) : Since we are working in the frequency domain, mostly all of our operations will consist of matrix multiplications.

Received signal at B $\mathbf{r}_p(\omega_q) = \mathbf{K}_c(\omega_q) S(\omega_q) \mathbf{e}_p$

Reshaped signal transmitted from B $\mathbf{x}_p(\omega_q) = \mathbf{W}(\omega_q)[\mathbf{K}_c^*(\omega_q) S^*(\omega_q) \mathbf{e}_p]$

Received signal at A $\mathbf{y}_p(\omega_q) = \mathbf{K}_c^T(\omega_q) \mathbf{W}(\omega_q) \mathbf{K}_c^*(\omega_q) S^*(\omega_q) \mathbf{e}_p$

Total energy of signal from clutter $\|\mathbf{y}\|_F^2 = \sum_{q=0}^{Q-1} \|\mathbf{K}_c^T(\omega_q)\, \mathbf{W}(\omega_q)\, \mathbf{K}_c^*(\omega_q)\|_F^2 |S(\omega_q)|^2$

[Fig. 5] Step 2: Equation for total energy of signal from clutter. Ref. [2]

Furthermore, we can represent the physical re-transmittance of a signal by multiplying the response of the environment we are sending the signal to with the conjugate of the signal being sent. Note that in figure 6, the transmitted signal from B has been conjugated, noting that it has been time-reversed and that the received signal from A is the received signal from B multiplied by our K-matrix, which represents the clutter response, or in other words mathematically represents the re-transmittance of our time-reversed signal back to A.

We design the optimal reshaping filter by taking the equation for the total energy of the signal from the clutter, and solving for the matrix that minimizes that total energy. To do this we apply three constraints to our energy equation, those being unit norm, symmetry, and constant volume as shown above in figure 5. The end result is that our optimal reshaping filter is the pseudo-inverse (denoted by the cross in figure 7) of the product of the transpose of the clutter response with the conjugate of the clutter response. Again, the entire proof can be found in the paper noted by reference [3].

Design waveform reshaping filter to minimize total clutter return

$$\mathbf{W}_{\text{opt}}(\omega_q) = \text{arg min} \|\mathbf{K}_c^T(\omega_q)\mathbf{W}(\omega_q)\mathbf{K}_c^*(\omega_q)\|_F^2$$

- Unit norm $\|\mathbf{W}(\omega_q)\|_F^2 = 1$
- Symmetry $\mathbf{W}(\omega_q) = \mathbf{W}^H(\omega_q)$
- Constant volume $|\det[\mathbf{W}(\omega_q)]| \le \prod_i w_{ii}$ (= constant)

[Fig. 6] Applying constrictions to solve for the optimal cancellation filter. Ref. [2]

So in our first step we received the K_c matrix at antenna array B, the clutter frequency response as a result from probing the clutter. We time-reversed this signal, K_c^* in order to re-transmit it. Now that we have the reshaping filter, we multiply the filter by our time-reversed signal K_c^* to get the final reshaped signal that will be sent from antenna array B back to antenna array A (remember that these steps so far are done without the target present in the media so we can generally compute this far before the target actually comes into play):

$$[\mathbf{x}_0(\omega_q) \cdots \mathbf{x}_{P-1}(\omega_q)] = k_q \left[\mathbf{K}_c^*(\omega_q)\mathbf{K}_c^T(\omega_q)\right]^{-1} \mathbf{K}_c^*(\omega_q).$$

$$[\mathbf{x}_0(\omega_q) \cdots \mathbf{x}_{P-1}(\omega_q)] = k_q \mathbf{K}_c^{-T}(\omega_q)$$

[Fig. 7] Representations for the signal to be retransmitted from antenna array B. Ref. [3]

3. When we receive this new signal sent from antenna array B to antenna array A, there are two possibilities. One is that the target is present in the medium and the other is if the target is not present in the medium. Should the target not be present in the medium, then the response of the environment will remain K_c and the received signal at A should be the product of our sent signal from B, namely X_i with respect to figure 8, with the response K_c which will yield the identity matrix. Note that the optimal filter is actually just the inverse transpose of the clutter response hence our expected result with no target is presence during the re-transmittance.

$$[\mathbf{y}_0(\omega_q)\cdots\mathbf{y}_{P-1}(\omega_q)] = k_q\mathbf{I}$$

[Fig. 8] Received re-transmitted signal without target in field. Ref. [3]

But if the target is present in the medium then we will get a response z_p' not corresponding to the identity matrix but as follows:

[Fig. 9] Received re-transmitted signal when target is in the field. Ref. [3]

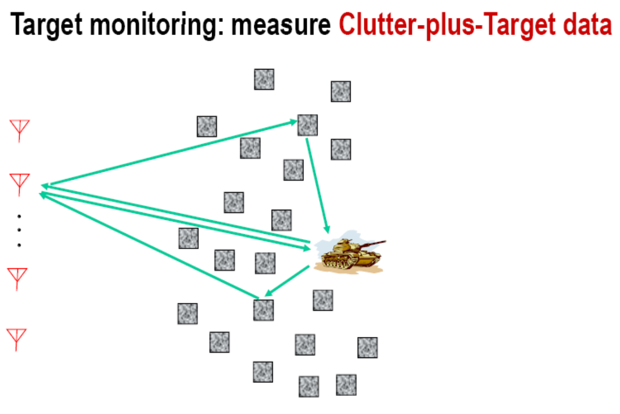
$$\begin{aligned}\mathbf{z}_p'(\omega_q) &= (\mathbf{K}_t(\omega_q) + \mathbf{K}_c(\omega_q))^T\mathbf{x}_p(\omega_q)\\ &= \mathbf{z}_p^t(\omega_q) + \mathbf{z}_p^c(\omega_q)\\ &= k_q\left(\mathbf{K}_t^T(\omega_q)\mathbf{K}_c^{-T}(\omega_q) + \mathbf{I}\right)\end{aligned}$$

[Fig. 10] Isolating target and clutter response by direct subtraction. Ref. [3]

$$\mathbf{K}_t(\omega_q) = \mathbf{K}_{c+t}(\omega_q) - \mathbf{K}_c(\omega_q).$$

Note that in figure 9 we use the equation shown in figure 10 to separate the target and clutter component responses to the retransmitted signal X_i into responses z_p^t and z_p^c, respectively.

Until now we have not defined when we obtain the response of the target plus the environment, namely K_{c+t}, but this can be easily attained by either consistent probing until a change appears in the response, or probing as soon as the received retransmitted signal no longer returns our expected identity matrix value.



▪ Clutter-plus-target data matrix

$$[\mathbf{K}_{c+t}(\omega)]_{ij} = k_{c+t}(\omega; B_i \leftarrow A_j)$$

[Fig. 11] Depiction of the response from probing when target is within environment. Ref. [3]

Also note that if the target is moving, we can assume that we can send and receive signals at a much faster rate than the target's movement so even if the target is moving we can update the target plus environment response and thus the target response much faster than the rate the target moves at.

4. In the case where the target is presence and we receive the response shown by figure 9, we can then subtract out the clutter component (namely the identity matrix) and we are left with the target component.

$$\mathbf{Z}(\omega_q) = \left[\mathbf{z}_0^t(\omega_q), \mathbf{z}_1^t(\omega_q), \ldots, \mathbf{z}_{P-1}^t(\omega_q)\right] \\ = \bar{k}_q \mathbf{K}_t^T(\omega_q)\mathbf{K}_c^{-T}(\omega_q)$$

[Fig. 12] Clutter component subtracted out from figure 9 to yield isolated target response. Ref. [3]

5. Now that we have used the reshaped time reversed signal to minimize the energy from the clutter response, we have successfully isolated a rough estimate of the target response. We thus retransmit the signal again using time reversal for refocusing. Now our signal should focus directly on the target and have little or no energy directed towards the scatters, however again we separate the environment response to our retransmitted signal so that we can subtract out additional clutter responses.

   In our received signal z_p (before said time-reversal and 2nd retransmittal), the transposed matrices have now taken on their hermitian values equivalent to phase conjugation. Note that we again multiply the sum of the target and clutter response to mathematically simulate the retransmittance. So as a result, back at antenna array B, we receive a final signal that has a clutter and target component.

[Fig. 13] Second retransmitted signal for additional focusing. Ref. [3]

$$\begin{aligned}\mathbf{p}'_p(\omega_q) &= [\mathbf{K}_t(\omega_q) + \mathbf{K}_c(\omega_q)][\mathbf{z}_p(\omega_q)]^* \\ &= [\mathbf{K}_t(\omega_q) + \mathbf{K}_c(\omega_q)] k_q \mathbf{K}_t^H(\omega_q)\mathbf{K}_c^{-H}(\omega_q) \\ &= k_q \mathbf{K}_t(\omega_q)\mathbf{K}_t^H(\omega_q)\mathbf{K}_c^{-H}(\omega_q) \\ &\quad + k_q \mathbf{K}_c(\omega_q)\mathbf{K}_t^H(\omega_q)\mathbf{K}_c^{-H}(\omega_q) \\ &= \mathbf{p}_p^t(\omega_q) + \mathbf{p}_p^c(\omega_q)\end{aligned}$$

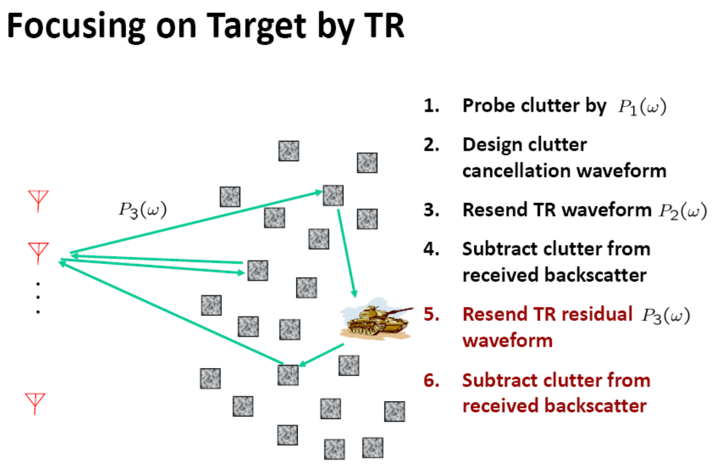
6. And now we just isolate the target component:

$$\begin{aligned}\mathbf{p}_p^t(\omega_q) &= \mathbf{p}'_p(\omega_q) - \mathbf{p}_p^c(\omega_q) \\ &= k_q \mathbf{K}_t(\omega_q)\mathbf{K}_t^H(\omega_q)\mathbf{K}_c^{-H}(\omega_q)\end{aligned}$$
$$\mathbf{M}^B(\omega_q) = \left[\mathbf{p}_0^t(\omega_q), \mathbf{p}_1^t(\omega_q), \ldots, \mathbf{p}_{P-1}^t(\omega_q)\right]$$

$$\begin{aligned}\mathbf{M}^A(\omega_q) &= k_q \mathbf{K}_t^T(\omega_q)\mathbf{K}_t^*(\omega_q)\mathbf{K}_c^{-*}(\omega_q) \\ &= k_q \mathbf{K}_t(\omega_q)\mathbf{K}_t^H(\omega_q)\mathbf{K}_c^{-H}(\omega_q)\end{aligned}$$

[Fig. 14] Final target-isolated signal received at antenna array A and at antenna array B, namely M^A and M^B, respectively. Ref. [3]

Note that the steps described assume that step 3 (our first clutter-cancelling TR signal) originates at antenna array B. We thus also specify the final result if we assume that step 3 originates from antenna array A. Note that the two results are the same except the transpositions are flipped.


Focusing on Target by TR
$P_3(\omega)$
1. Probe clutter by $P_1(\omega)$
2. Design clutter cancellation waveform
3. Resend TR waveform $P_2(\omega)$
4. Subtract clutter from received backscatter
5. Resend TR residual $P_3(\omega)$ waveform
6. Subtract clutter from received backscatter

[Fig. 15] Second time reversal for additional refocusing. Ref. [3]

7. With the responses measured at each antenna array we then calculate the final image as a spatial distribution of the total energy at each pixel using the following formula:

$$I(\mathbf{x}) = \sum_{q=0}^{Q-1} |Y^A(\mathbf{x};\omega_q)Y^B(\mathbf{x};\omega_q)|^2$$

[Fig. 16] Formula for relating each pixel and the corresponding energy. Ref. [3]

$$\mathbf{w}_{rB}(\mathbf{x};\omega_q) = \frac{\mathbf{g}_B(\mathbf{x};\omega_q)}{||\mathbf{g}_B(\mathbf{x};\omega_q)||}$$
$$\mathbf{w}_{tB}(\mathbf{x};\omega_q) = \frac{\mathbf{K}_c^{-1}(\omega_q)\mathbf{g}_B(\mathbf{x};\omega_q)}{||\mathbf{K}_c^{-1}(\omega_q)\mathbf{g}_B(\mathbf{x};\omega_q)||}$$
$$\mathbf{w}_{rA}(\mathbf{x};\omega_q) = \frac{\mathbf{g}_A(\mathbf{x};\omega_q)}{||\mathbf{g}_A(\mathbf{x};\omega_q)||}$$
$$\mathbf{w}_{tA}(\mathbf{x};\omega_q) = \frac{\mathbf{K}_c^{-T}(\omega_q)\mathbf{g}_A(\mathbf{x};\omega_q)}{||\mathbf{K}_c^{-T}(\omega_q)\mathbf{g}_A(\mathbf{x};\omega_q)||}.$$

[Fig. 17] Weight vectors used for calculating the beamformers. Ref. [3]

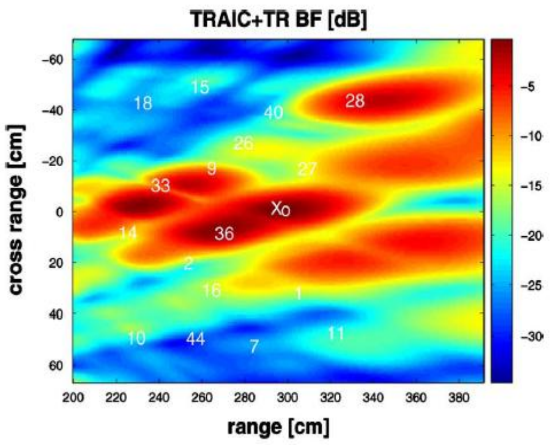
This represents the combination of two beam-formers Y^A and Y^B by triangulation. Note that Y^A and Y^B come straight from multiplying our response by weight vectors of unit norm calculated from the green's function and clutter response. Note that for each antenna array we have a receive and transmit beam for each frequency. Forming the image represents the most computationally heavy part of our algorithm.

$$Y^B(\mathbf{x};\omega_q) = \mathbf{w}_{rB}^H(\mathbf{x};\omega_q)\mathbf{M}^B(\omega_q)\mathbf{w}_{tB}(\mathbf{x};\omega_q)$$
$$Y^A(\mathbf{x};\omega_q) = \mathbf{w}_{rA}^H(\mathbf{x};\omega_q)\mathbf{M}^A(\omega_q)\mathbf{w}_{tA}(\mathbf{x};\omega_q).$$

[Fig. 18] Complete equations for each beamformer. Ref. [3]

Thus if we contain our samples to the same environment, we only need to calculate the weight vectors once, and our computation time will be significantly decreased.

We can then plot an image of our region where each pixel has an associated energy. Given our algorithm, we should expect the peak energy to represent roughly the location of the target. Note that our results are far from perfect, as shown



[Fig. 19] Final imaged area of interest based on energy of returns correlated with green's function calculation. Ref. [3]

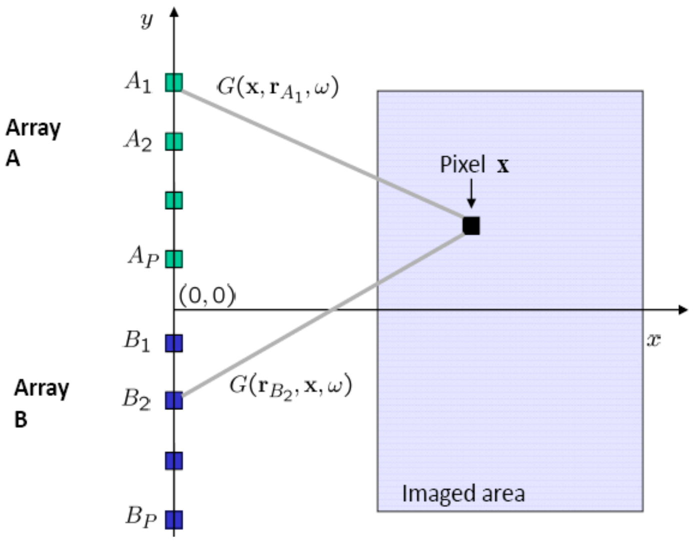
So what have we used time-reversal for in all of this?

How to minimize backscatter from clutter while focusing on target?

- **Solution: TR twice**
  - **Adaptive interference cancellation** (1st TR)
    reshape transmission waveforms to minimize backscatter from clutter (focus on clutter)
  - **Target focusing** (2nd TR)
    back propagate the time-reversed wave field (focus on target)

Green's function:

Two-point Green's function G(r_1, r_2, *w*) tells us (assuming free space conditions) the response at point r_1 to an impulse from point r_2 with frequency *w*. So if we send a known signal from an antenna A_1, we have a function that approximates what we should receive at *x*, namely G(*x*, A_1, *w*). We can also calculate the Green's function between a point in antenna B. If there is a point of reflection (scatter or target) at pixel x, Green's function gives us a relation between what is sent by A and received by B.



[Fig. 20] Complete equations for each beamformer. Ref. [3]

$$G(\mathbf{r}, \mathbf{r}'; \omega_q) = \frac{1}{4j} H_0^{(2)}(k_q|\mathbf{r} - \mathbf{r}'|)$$

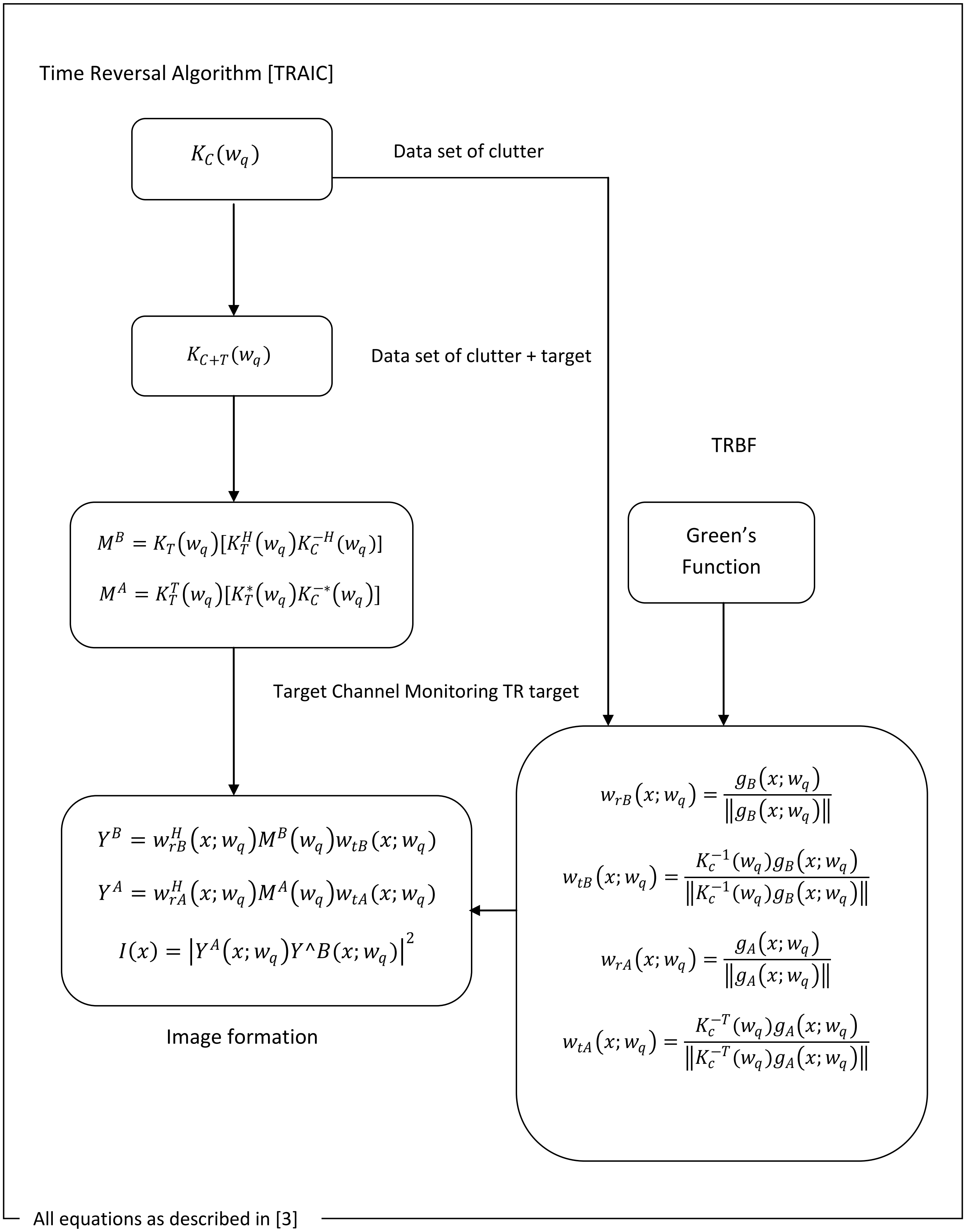
Where H_0 is the zeroth-order Hankel function.

We assume that Green's function also satisfies the reciprocity relation: $G(\mathbf{r}, \mathbf{r}'; \omega_q) = G(\mathbf{r}', \mathbf{r}; \omega_q)$.

Far-field approximation: $G(\mathbf{r}, \mathbf{r}'; \omega_q) \approx e^{-jk_q|\mathbf{r}-\mathbf{r}'|}$

Since we are computing Green's function offline on the PC, we can use the initial and more accurate equation to calculate green's function for the entire imaged area.

Algorithm Visual Overview


Time Reversal Algorithm [TRAIC]
$K_C(w_q)$
Data set of clutter
$K_{C+T}(w_q)$
Data set of clutter + target
$M^B = K_T(w_q)[K_T^H(w_q)K_C^{-H}(w_q)]$
$M^A = K_T^T(w_q)[K_T^*(w_q)K_C^{-*}(w_q)]$
Target Channel Monitoring TR target
$Y^B = w_{rB}^H(x;w_q)M^B(w_q)w_{tB}(x;w_q)$
$Y^A = w_{rA}^H(x;w_q)M^A(w_q)w_{tA}(x;w_q)$
$I(x) = \left|Y^A(x;w_q)Y\wedge B(x;w_q)\right|^2$
Image formation
TRBF
Green's Function
$w_{rB}(x;w_q) = \frac{g_B(x;w_q)}{\|g_B(x;w_q)\|}$
$w_{tB}(x;w_q) = \frac{K_c^{-1}(w_q)g_B(x;w_q)}{\|K_c^{-1}(w_q)g_B(x;w_q)\|}$
$w_{rA}(x;w_q) = \frac{g_A(x;w_q)}{\|g_A(x;w_q)\|}$
$w_{tA}(x;w_q) = \frac{K_c^{-T}(w_q)g_A(x;w_q)}{\|K_c^{-T}(w_q)g_A(x;w_q)\|}$

All equations as described in [3]

Results

The plots below represent the final output of our algorithm. What we have is 3 pairs of images, one pair of images for each of our data sets (data from 3 different experimental setups). Each image represents a plot of the desired area to be imaged, where ‘range’ denotes the x-axis and ‘cross range’ denotes the y-axis. For each pixel the corresponding color to that pixel represents the magnitude of the energy response from that physical point (x, y) in the imaged area.

Looking at the dB color scale, blue represents low energy in the response, and red, strong energy in the response. What we mean by “energy” of the response is the reflectivity at that point, so the higher the energy at a point, the stronger the reflectivity and ideally, the more likely that there is some object at that point (since we do clutter suppression we would ideally hope that only the target location has strong reflectivity).

For each plot note that there is an ‘x’ and ‘o’ marked. The ‘x’ represents the target’s actual location, where ‘o’ represents the maximum energy peak of the plot, the point where the reflected energy is the highest. The black and bold numbers marked on the plot represent the approximate location of the scatters. In the experimental setup, these scatters were simulated by long copper pipes.

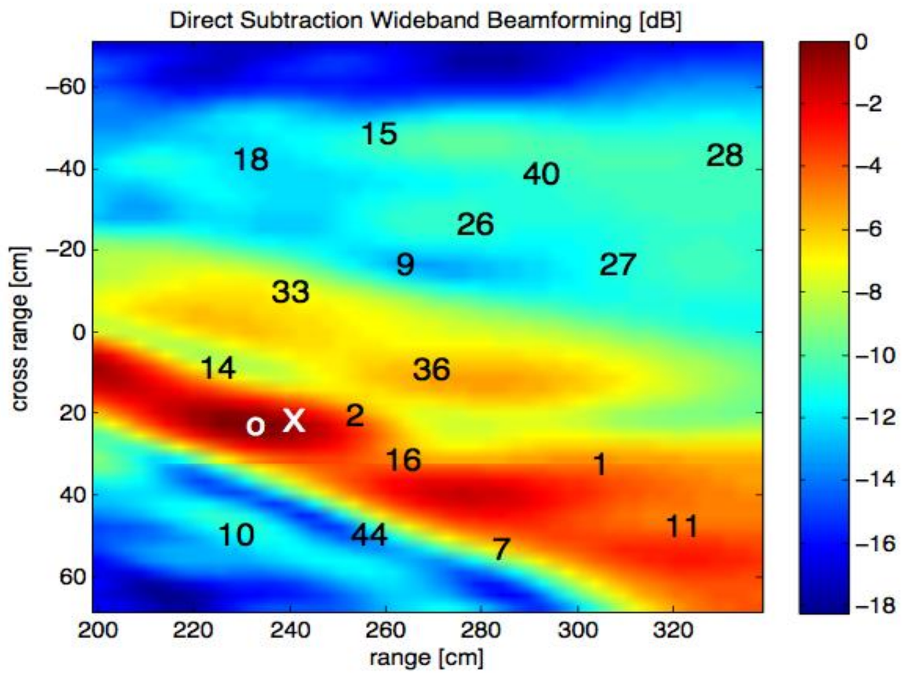
Each data set is comprised of two images, one showing the results from conventional detection and imaging, and one showing the results from our time-reversal based algorithm. The plot labeled “direction subtraction” represents a plot generated by calculating the difference between the clutter response and the clutter plus target response and doing beam forming without any additional steps. This is the conventional and simplest of detection and imaging schemes, where as our algorithm finds the difference, and then execute the algorithm as we described above, before doing beam forming to create the image. The plot labeled “TRAIC-TR” represents the plot generated from our algorithm.

Data Set 1

Looking at the first data set, it has the least number of scatters out of all the setups, so we would probably expect the least performance gain from TR in this case out of all the cases since TR improvements increase with an increased number of scatters, that is multipath becomes an advantage. As one can see from the plots, the actual accuracy in detection is very close, but TR does a noticeably better job in general at suppressing the energy reflected from the scatters, as most of the stronger returns in the TR plot are localized around the target location. In the direct subtraction plot, the area of strong returns stretches along the bottom of the image and covers a larger area than in the TR plot. The strong returns not reflected by the target position are mostly due to secondary scattering, in which we have no method to predict.

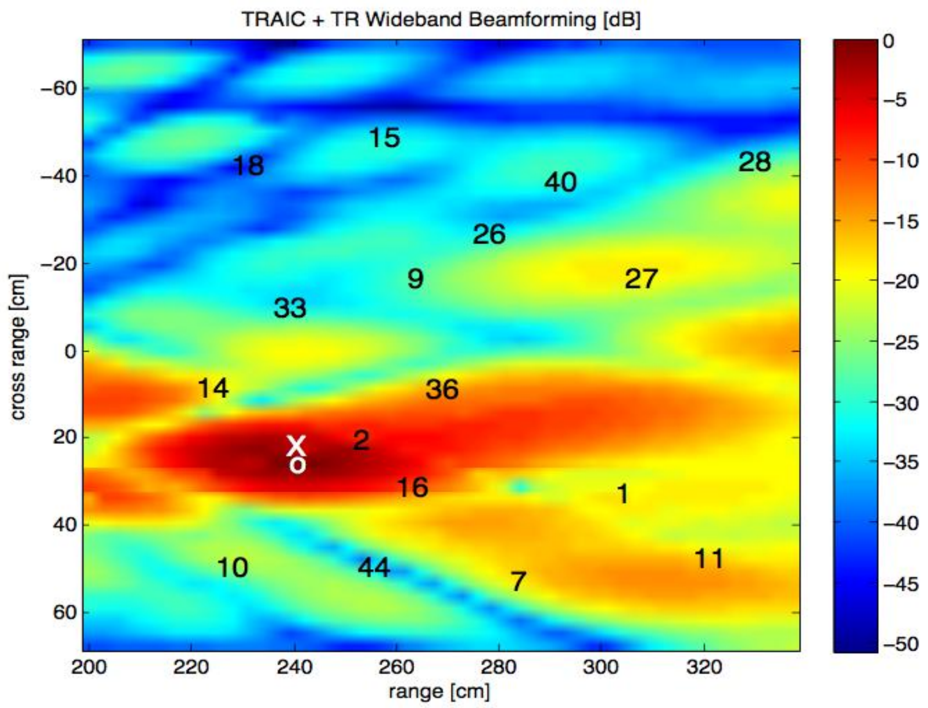
Data Set 1

Figure 1



Data Set 1

Figure 2



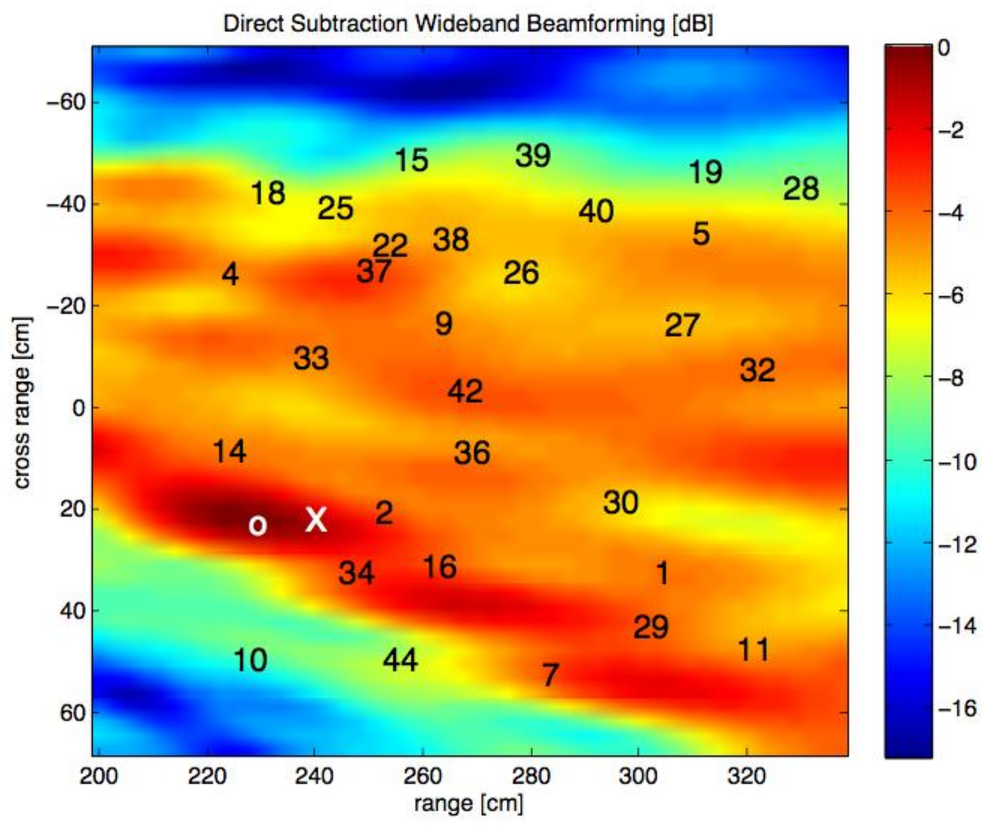
Data Set 2

In this setting we increase the number of scatters and change the scatter and target positions. Notice that direct subtraction, despite being not significantly far off from detecting the target, returns significant energy from almost all points of the image. TR imaging manages to get a more accurate location on the target (smaller distance between ‘x’ and ‘o’) as well as significantly suppressing most of the environment response.

As mentioned before, there is the factor of secondary scattering coupled with the fact that our target response, modeled from the directly subtracting the clutter response from the target plus clutter response, is of course imperfect. So we see a couple of slightly strong returns in unexpected areas of the TR plot, but overall it does an improved job of suppressing the energy returned from the environment and successfully localizing the target to a small enough region.

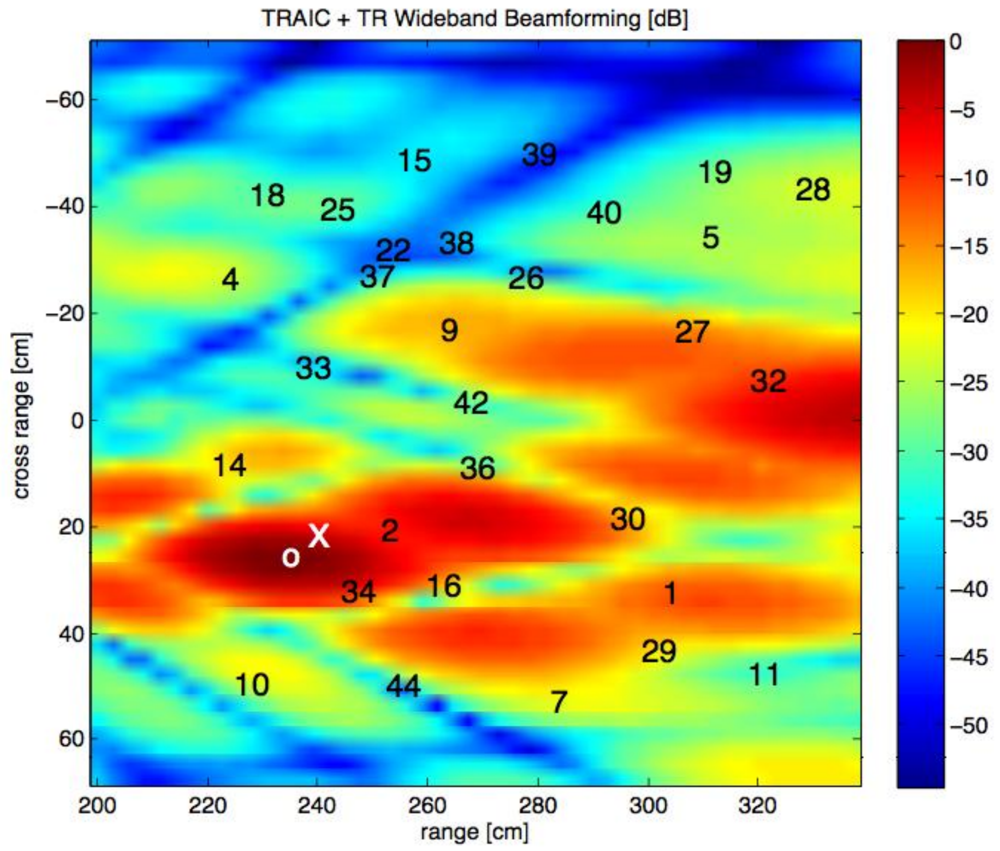
Data Set 2

Figure 3



Data Set 2

Figure 4



Data Set 3

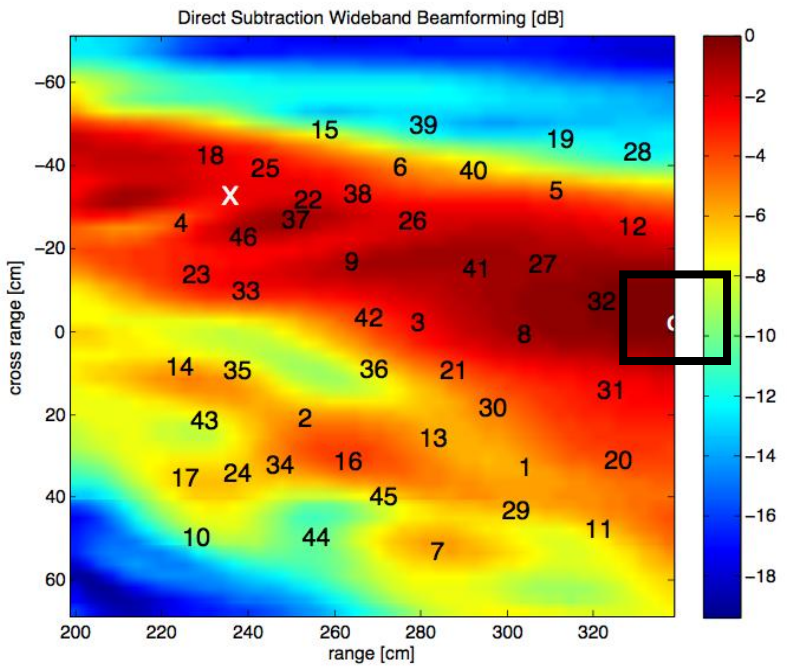
Data set 3 uses the exact same setup as data set 2 except more scatters have been added. Thus we would expect direct subtraction to perform the worst here and our TR algorithm to show the best improvements under these conditions.

Our result shows that direct subtraction fails quite miserably in this case, displaying intense energy returns all over the top region of the imaged area. As a result, the energy peak is almost off the map, having a distance from the actual target pint almost equal to the entire range of the imaged area. Examining more closely, the stronger returns don’t even localize around the target but rather along the scatters behind the target.

Looking at the TR result, the mass energy returns direct subtraction had have mostly been mitigated. There is still a region of substantial energy not localized around the target, but the strongest returns form a nice circular region about the target location, and the target location is fairly close to the energy peak. So in this case we note that the TR algorithm has given us a fairly large improvement over conventional imaging methods, thus the more scatters, the better our gains.

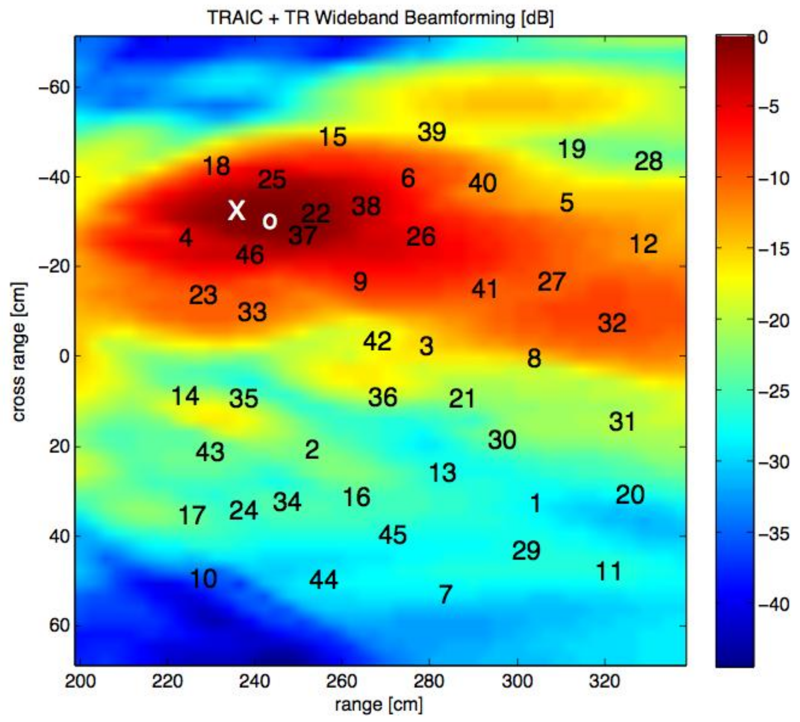
Data Set 3

Figure 5



Data Set 3

Figure 6



What are the DSK and PC doing?

The PC loads the clutter data and clutter plus target data via MATLAB. Any necessary operations on the clutter data only (inverse, conjugate, hermitian) are calculated in MATLAB, and then the clutter plus target data, clutter data, and modified clutter data is all written to an output file. In particular we calculate $K_C^{-*}(w_q)$ and $K_C^{-H}(w_q)$, and send over these two along with $K_C(w_q)$ and $K_T(w_{q)})$. This seems reasonable since the clutter data is known far before the clutter plus target data is obtained so we can use the PC to quickly calculate the inverses we need.

The PC also uses MATLAB to calculate Green's function. Since Green's function depends only on the imaged area, this represents a beforehand calculation as well, so we do it quickly on the PC. The main idea is for the DSK to achieve near real-time performance, thus we limit its actions to strictly the operations containing clutter plus target data.

The PC side code reads all the data in from the MATLAB output file and transfers it to the DSK for processing. In our code, a total of four request transfers are used to transfer the data over to the DSK.

The DSK takes the clutter only data, clutter plus target data, and the modified clutter only data to generate the final response after mathematically implementing our time reversal algorithm. The entirety of the DSK operations can be expressed by the following equations.

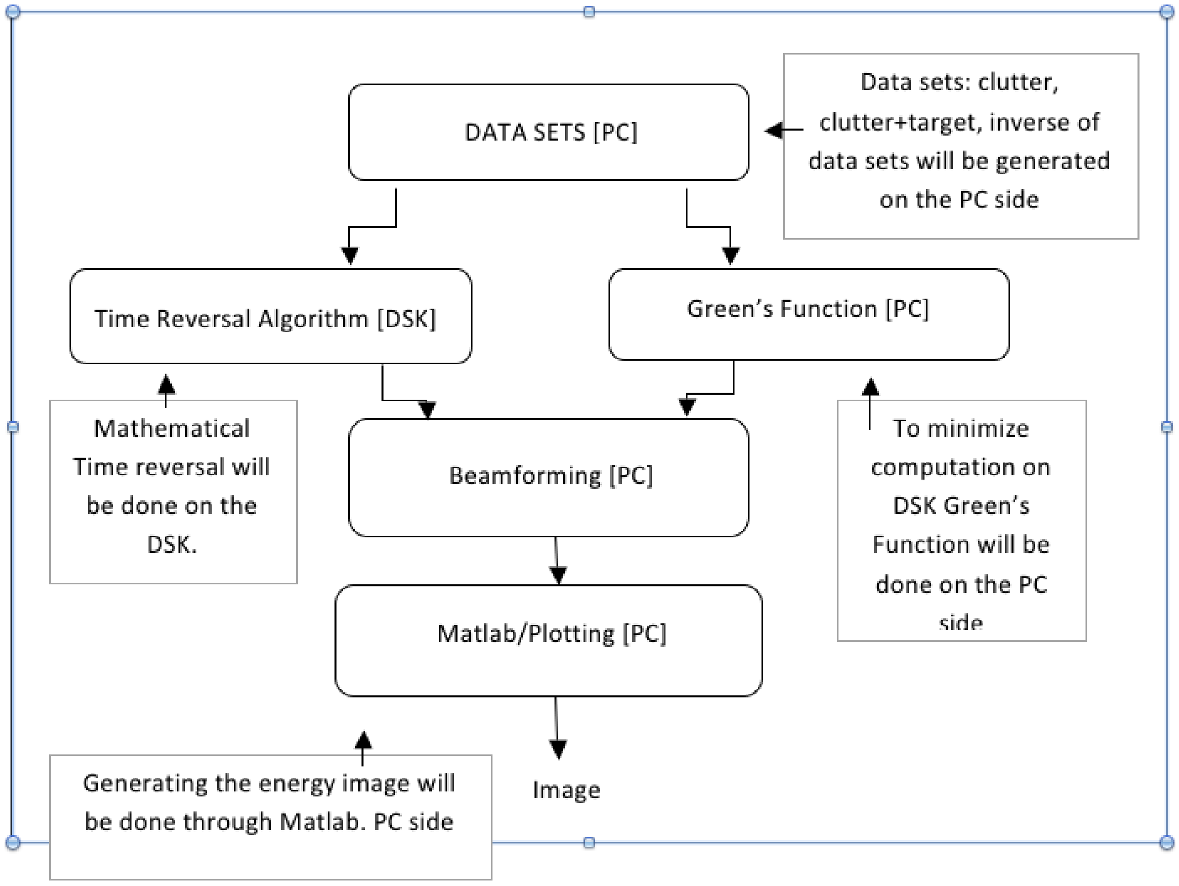
$$M^B = K_T(w_q)[K_T^H(w_q)K_C^{-H}(w_q)]$$

$$M^A = K_t^T(w_q)[K_T^*(w_q)K_C^{-*}(w_q)]$$

Where M^B represents the clutter focused target response matrix measure with respect to array B (for array A being the transmitter and array B being the receiver) and M^A corresponds to the clutter target response matrix measured at array A (array B being the transmitter and array A being the receiver).

We then use only two request transfers to move the data back to the PC since we only have two results to transfer. The PC side code writes the data to an output file to be opened and read in by MATLAB for plotting. To do the plotting, beamforming is required, which uses weighted vectors computed from Green's Function. Ideally this should be done on the DSK as well so all the PC has to do is take the result from the DSK and plot it, but we did not have sufficient time to implement beamforming on the DSK. See the timing section for more details on projected performance using C as compared to our current MATLAB beamforming code.

Task Division between PC & DSK


DATA SETS [PC]
Data sets: clutter, clutter+target, inverse of data sets will be generated on the PC side
Time Reversal Algorithm [DSK]
Green's Function [PC]
Mathematical Time reversal will be done on the DSK.
Beamforming [PC]
To minimize computation on DSK Green's Function will be done on the PC side
Matlab/Plotting [PC]
Generating the energy image will be done through Matlab. PC side
Image

What Code We Used

The most critical code we used was the TI C67 DSP library we found after much searching on the TI website (to which the download link is now broken). The library does not support complex transpose but does support complex matrix multiplication, which greatly improved the performance of our algorithm compared to regular C code since the assembly code was already hand tuned. All other code was self-written, most of it concerning transfers either between MATLAB and the PC or between the PC and DSK. Time constraints did not allow us to find proper complex matrix transpose code so we were forced to implement a straight-forward intuitive method that performs much worse than even the complex matrix multiplication as noted in the timing section.

Since we did not implement beamforming on the dsk there weren't extremely heavy performance constraints. The limiting factor from our code's performance is probably the complex matrix transpose since our implementation is very basic. Most of our transfer and read/write code was optimized for maximum performance since the memory requirement wasn't significantly high. Though we could have easily processed more frequencies on the DSK with time to spare, our choice of the load to put on the DSK was directly limited by the MATLAB code which already took several seconds to run for ultra-low resolutions and frequency numbers.

In the timing section below we evaluate the possible performance for c-based beamforming and estimate the efficiency the algorithm could gain from implementing the MATLAB beamforming (most computationally heavy and longest computation time as of now) in C and on the DSK.

Timing Analysis and Performance

General benchmark….

Complex matrix multiplication in MATLAB: 0.0005987 seconds, 598.7 µs
(multiplied two complex matrices of size 10x10)

Matrix multiplication in C: 0.000031 seconds, 31.0 µs
(multiplied two complex matrices of size 10x10, 1000 times and divided by 1000)

DSK multiplication in C:
8*r1*c1*c2'+ 18*(r1*c2)+40
8*10*10*10+ 18*(10*10)+40= 9840 cycles * 4.4ns = 43.2960 µs

In MATLAB …

Read in data, declare and initialize parameters for imaging = 0.039006 sec
[Calculate inv herm and inv conj and write four matrices to file = 0.020960 sec]
Compute inv herm and inv conj = 0.0119325 sec
Write to file (unwrap, interleave, etc) = 0.0090275 sec
PC side C code….

Read in four matrices on PC side C code = 0.001328 sec (read them in 1000 times, divided by 1000)

DSK side C code…

DSK to request data transfer from PC four times (the 4 matrices) = 12982947 cycles * 4.4ns sec/cycle = 0.0571249 seconds
4 DMA transfers at top of loop = 769 cycles * 4.4ns = 3.3836 µs
(target plus clutter response - clutter response) for one pair of matrices = 31 cycles * 4.4ns = 0.1364 µs
Compute conjugate of target response = 42 cycles * 4.4ns = 1.848e-7 s = 0.1848 µs
Compute transpose of target response (one matrix) = 13 cycles * 4.4ns = 57.2000 ns
Compute transpose of target response conj (one matrix) = 13 cycles * 4.4ns = 57.2000 ns
2 DMA transfers at bottom of loop = 880628 cycles * 4.4ns = 0.003874763 seconds

PC side C code…

Write 2 result matrices from PC side C code to be read into MATLAB = 0.0006639000 seconds

In MATLAB….

Read 2 results matrices back into MATLAB, reshape matrices, reconstruct complex numbers = 0.012500 seconds

Beamforming time = 7.1409070 seconds
Plotting time = 0.7109390 seconds

Beamforming:

Time to multiply three matrices 1x tmpB = 0.00550317915 sec
This time * 26 * 50 * 50 = 357.706644750 sec (for three loops) (this doesn't seem right)

Time to multiply three matrices 1x tmpA = 4.3176150e-5 sec = 43.17615 µs
This time * 26 * 50 * 50 = 2.8064497500 sec (for three loops)

Time measured for MATLAB to do M mult in loop = 7.508441924 seconds

Size(vB') = 1 10
Size(fix2)= 10 10
size(vKb/sqrt(vKb'*vKb)) = 10 1

0.003200 to perform 1000 of the three matrix mults
3.2 µs /matrix mult
0.003200 * 26 * 50 * 50 * 10^-3 = 0.20800 sec

Speed of complex matrix multiplication was investigated to learn about the computational costs of time reversal and beamforming. Benchmark times for multiplication of two 10x10 complex matrices were measured to be 598.7 µs, 31 µs, and 43.296 µs for MATLAB, PC-side C code, and DSK-side C code, respectively (see table below).

Multiplication of Two Complex Matrices of Size 10x10		
MATLAB	598.7	µs
PC-Side C Code	31	µs
DSK-Side C Code	43.296	µs

The profile times below were measured for all sections of the MATLAB code, PC-side C code, and DSK-side C code.

MATLAB				
Read in data, declare and initialize imaging parameters			0.039006	s
Compute inverse hermatian, inverse conjugate			0.0119325	s
Write to file (unwrap, interleave, format)			0.0090275	s
PC side C Code				
Read in four matrices on PC side C code			0.001328	s
DSK side C Code				
DSK to request data transfer from PC four times	12982947	cycles	0.0571249	s
Four DMA transfers for pre-calculation paging	769	cycles	3.3836	µs
Direct Subtraction Step	31	cycles	0.1364	µs
Compute conjugate of target response	42	cycles	0.1848	µs
Compute transpose of target response (one matrix)	13	cycles	57.2	ns
Compute transpose of target response conjugate (one matrix)	13	cycles	57.2	ns

Two DMA transfers for post-calculation paging	880628	cycles	0.003874763	s
PC side C Code				
Write 2 result matrices from PC side C code to be read in MATLAB			0.0006639	s
MATLAB				
Read two results matrices back into MATLAB, reshape matrices, reconstruct complex numbers			0.0125	s
Beamforming time			7.140907	s
Plotting time			0.710939	s

Beamforming is believed to be the most computational intensive part of the project. It features complex matrix multiplication inside three nested loops. In MATLAB, this matrix multiplication step was profiled to take 43.17615 µs. Repeated through 26*50*50 = 65000 iterations, we multiply to get an estimated total computation time of 43.17615 µs * 65000 = 2.80644975 seconds. Profiling the three nested loops in MATLAB as a whole, we get 7.508441924 s. The difference 7.508441924 s - 2.80644975 s = 4.701992174 s is indicative of the extra time MATLAB takes to perform matrix multiplication inside of loops. Comparatively, the same multiplication step in C was profiled to take 3.2 µs. At 65000 iterations, this step takes an estimated 3.2 µs * 65000 = 0.20800 s. The difference in estimates between the MATLAB and C implementations of this matrix multiplication step is 2.80644975 s - 0.20800 s = 2.5984497 s. The C code implementation took roughly (0.20800 / 2.80644975) * 100 = 7.41% of the runtime of the MATLAB implementation.

Full Semester Schedule & Task Division

Date	Goal	Who
Oct 12th	Understand the TRAIC Algorithm	All
Oct 19th	Decide what should be done on PC and DSK	ALL
Oct 26th	Meeting with Yuanwei to decide PC and DSK parts, Start coding	Alan, Dave
Nov 2nd	Begin DSK & PC side coding	ALL
Nov 9th	Finish Coding for DSK	Alan
Nov 16th	Begin profiling, finish PC and DSK side code	ALL
Nov 30th	Finish profiling and preparing for final presentation	ALL

Future Improvements and General Recommendations

This project had a lot of educational overhead, and because the research has not fully developed, the results are not definite or completely convincing. That said, the project still has great relevancy since the DSK is perfectly suited for matrix operations. Since the matrices contained complex data, finding libraries and online code was nearly impossible. It's not as elegant or surefire as other classical projects but it was good to step outside the realm of normal DSP applications. Furthermore, we were not able to implement the entire algorithm, although there was a high possibility that the entire algorithm could very well run in real-time on the DSK.

References

[1] J. Moura, Y. Jin "Time Reversal Imaging by Adaptive Interference Canceling," ***IEEE Transactions on Signal Processing,*** **vol. 56, no. 1, Jan. 2008.**

[2] IMAGES WERE TAKEN FROM "*Time Reversal Slides*.PDF" put together by Yuanwei Jin.

[3] J. Zhu, Y. Jiang "Time Reversal Method for Target Detection Using Electromagnetic Waves in a Cluttered Environment," ***IEEE*** **, Oct. 2004.**

[4] IMAGES WERE TAKEN FROM "*Time Reversal Adaptive Interference Canceller (TRAIC) for EM Wave Target Detection in Cluttered Environment.ppt.*" put together by Jimmy Zhu & Yi Jiang.

[5] TMS320C67x DSP Library: TI library with hand-tuned assembly code for complex matrix multiplication on C67x DSPs.
http://focus.ti.com/dsp/docs/dspplatformscontentaut.tsp?familyId=327&sectionId=2&tabId=499
(download link no longer working, but downloaded zip file available upon request)