DELHI TECHNOLOGICAL UNIVERSITY


# Computer Vision (EC353)
# Project

## To implement a Wideband/Narrowband Fusion-Based Motion Estimation Method


Submitted by:

Gourav Garg
2K17/MC/043

# Introduction

A new wideband/narrowband fusion based motion estimation method is proposed for maneuvering target. In the fusion scheme, a fast motion parameters estimation method based on cross-correlation of adjacent echoes (CCAE) is adopted as the wideband estimation method. The narrowband estimation method is the maximum likelihood estimation with Newton's method (MN method). The proposed method mainly includes three steps.

First, the velocity and acceleration of the target are estimated by CCAE. Second, the velocity and acceleration estimated by CCAE are adopted as the initial velocity and final acceleration of MN method, respectively. Finally, the high precision velocity and distance of the target are estimated by MN method. The proposed fusion method has two advantages.

Due to the large scope of the unambiguous velocity of CCAE, the velocity ambiguity problem of MN method is solved. Second, the three dimensional (3D) search in MN method is reduced to two dimensional (2D) search. The simulation results demonstrate that the proposed fusion method achieves similar estimation performances on distance and velocity with much lower computational cost, compared with the MN method.

## Scheme of Wideband/Narrowband Fusion Estimation

In the fusion scheme, narrowband and wideband LFM pulses are transmitted alternately. Assume that radar transmit 2M pulses, including M narrowband pulses and M wideband pulses. The object of the proposed fusion method is to estimate the range, velocity and acceleration of the target using one period of echo pulses. In this subsection, we first give the signal models and algorithms of CCAE and MN method respectively, then the proposed fusion method is presented.

1) The MN Method: The transmitted narrowband LFM signal can be written as:

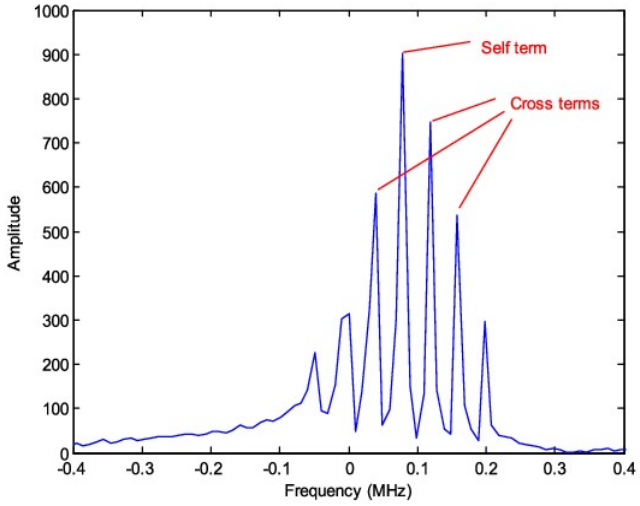
$$s^{tr}_n(t) = \text{rect}(t/T)\, s_n(t) \exp(j2\pi fct),$$

where $s_n(t) = \text{rect}(t/T) \exp(j\pi\gamma n\, t^2)$ denotes the complex envelope of the transmitted pulse, the subscript n indicates the signal is narrowband signal.

2) The CCAE Method: The transmitted wideband LFM signal can be written as:

$$s^{tr}_w(t) = \text{rect}(t/T)\, s_w(t) \exp(j2\pi fct),$$

where $s_w(t) = \text{rect}(t/T) \exp(j\pi\gamma w\, t^2)$ denotes the complex envelope of the transmitted pulse, the subscript w indicates the signal is wideband. $\gamma_w = B_w / T$ denotes the chirp rate of LFM signal and $B_w$ is the swept bandwidth of the wideband LFM signal.




3. The 1D spectrum of the cross-correlation result (SNR = 0 d

3) The Fusion Estimation Method: The CCAE method estimates the motion parameters of the target at the first wideband pulse, while the MN method estimates the motion parameters of the target at the intermediate position of the accumulated narrowband pulses. In other words, the time position of the estimated motion parameters by MN method is the M-th pulse whether M is odd or even. In MN method, if the search scope of velocity, which is generally set as the scope of unambiguous velocity, is less than the difference between the initial velocity and the true velocity, the peak of the ambiguity function can't be found. Therefore, it is necessary to make the two methods estimate the parameters at a consistent time.

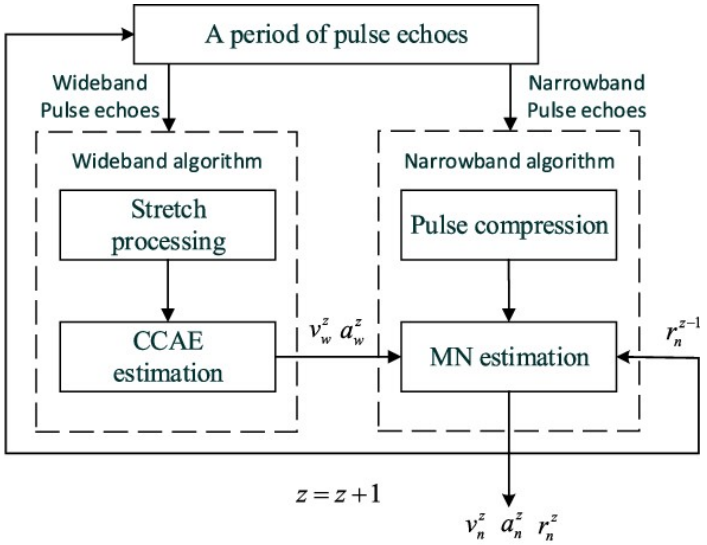



Fig. 4. The flowchart of the fusion method.

## On Solving the Velocity Ambiguity Problem

In order to obtain the scope of unambiguous velocity, the acceleration is set as zero. Then the velocity obtained by CCAE, can be expressed as:

$$v_w = cf \times f_r / 2\gamma_w$$

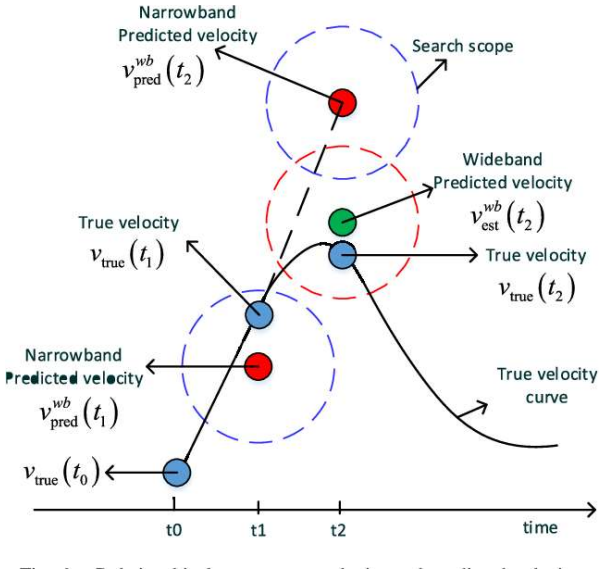
where $f_r$ is the pulse repetition frequency (PRF) and f is the estimated frequency of the Fourier transform (FT) result. Because the maximum value of the estimated frequency is equal to the sampling frequency, the scope of unambiguous velocity of wideband method can be written as

$$\bar{v}_w = c\, f_s \times f_r / 2\gamma_w$$

where $f_s$ is the sampling frequency. Similarly, the scope of unambiguous velocity of MN method can be written as

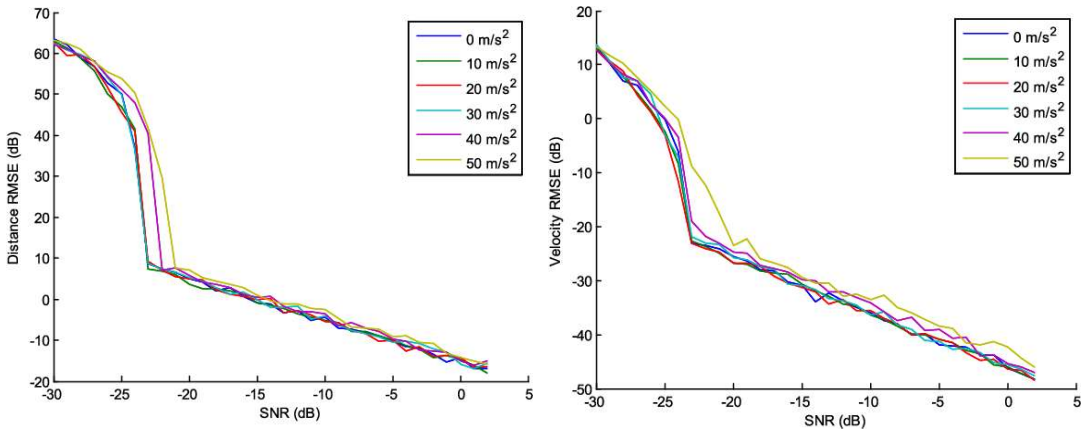
$$\bar{v}_n = c \times f_r / 2 f_c$$

where $f_c$ is the carrier frequency. Using the radar parameters the scope of unambiguous velocity can be calculated: $\bar{v}_n$ = 2.83 m/s and $\bar{v}_w$ = 25482 m/s.




## Influence of Initial Acceleration on Performance of Distance and Velocity

In the proposed fusion method, the estimated velocity by CCAE is used as the initial velocity of MN method, while the estimated acceleration by CCAE is used as the final acceleration of MN method. We first compare the acceleration and velocity performances between the wideband CCAE and narrowband MN methods. Then, for MN method, the influence of the initial acceleration on the performance of distance and velocity is analysed. Some preliminary experiments are conducted in this subsection.



## OpenCV (C++) Code :

```
#include <iostream>
#include "opencv2/core.hpp"
#include "opencv2/features2d.hpp"
#include "opencv2/xfeatures2d.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/xfeatures2d.hpp"
#include "opencv2/xfeatures2d/nonfree.hpp"
#include <opencv2/opencv.hpp>
#include <iostream>
#include <vector>
#include <stdio.h>
using namespace cv;
using namespace std;
using namespace cv::xfeatures2d;
int main( int argc, char* argv[] )
{


   double t = (double)getTickCount();
   cv::VideoCapture capture("/Users/kakarlakeerthy/Desktop/prj1/prj1/input.mov"); // input video present in input folder


   double fps = capture.get( cv::CAP_PROP_FPS );// obtaining  frames per second


   cv::Size size(
            (int)capture.get( cv::CAP_PROP_FRAME_WIDTH ),
            (int)capture.get( cv::CAP_PROP_FRAME_HEIGHT )
            );// getting width and height of video



   cv::Mat bgr_frame1,frame_gray1, img_keypoints,bgr_frame2, frame_gray2,gray2, img_keypoints2, bgr_frame3,frame_gray3,img_keypoints3;
```

```
std::vector<KeyPoint> keypoints1 , keypoints2,keypoints3;
cv::Size winSize = Size(11,11);
int maxLevel = 3;
double minEigThreshold = 1e-4;
cv::TermCriteria criteria= TermCriteria(cv::TermCriteria::COUNT | cv::TermCriteria::EPS,30,0.01);
Mat descriptors, err;
vector<uchar> status;
CV_OUT std::vector<Point2f> points2f1,points2f2, points2f3;
int length = int(capture.get(CAP_PROP_FRAME_COUNT));
const std::vector< int > &      keypointIndexes = std::vector< int >();
const std::vector< int > &      keypointIndexes2 = std::vector< int >();
cv::VideoWriter writer; // video writer object


capture >> bgr_frame1; // taking first frame from the video and storing in bgr_frame1
Ptr<Feature2D> f2d = SURF::create(7000); //for SURF, we set hessianThreshold = 7000 to get a limited number of keypoints
cv::cvtColor( bgr_frame1, frame_gray1, cv::COLOR_BGR2GRAY);// converting bgr_frame1 to gray image and storing in frame_gray1
f2d->detect(frame_gray1, keypoints1);// detecting keypoints for first frame using SURF algorithm


KeyPoint::convert(keypoints1,points2f1,keypointIndexes);// converting keypoints of first frame to Points2f type


Mat imgLines = Mat::zeros( frame_gray1.size(), CV_8UC3 );// creating an empty image with size equals to frame_gray1 size.
// char c = cv::waitKey(fps);
capture >> bgr_frame2;// get second frame from the video
cv::cvtColor( bgr_frame2, frame_gray2, cv::COLOR_BGR2GRAY);//converting bgr_frame2 to gray image and storing in frame_gray2
f2d->detect(frame_gray2, keypoints2);// detecting keypoints for second frame using SURF algorithm


KeyPoint::convert(keypoints2,points2f2,keypointIndexes2);//converting keypoints of 2nd frame to Points2f type
length = length -1;
writer.open("/Users/kakarlakeerthy/Desktop/Proj1/output.mp4", CV_FOURCC('M','J','P','G'), fps, size );// output video path and type
```

```
   for(;;) // infinite loop


   {
      length = length -1;
      if (length == 2 )
         break;
      calcOpticalFlowPyrLK(frame_gray1,frame_gray2,
points2f1,points2f2,status,err,winSize,maxLevel,criteria, minEigThreshold); //calculating optical
flow for two frames


      frame_gray2.copyTo(frame_gray1); // copying content of frame_gray2 to frame_gray1


      for(size_t i=0; i<points2f2.size(); i++)//
      {
         if(status[i])
         {
            if (i == 0)
            {
               line(imgLines,points2f1[i],points2f2[i],Scalar(0,0,255),4);
            }
            if ( i == 1)
            {
               line(imgLines,points2f1[i],points2f2[i],Scalar(0,255,0),4);
            }
            if ( i ==2 )
            {
               line(imgLines,points2f1[i],points2f2[i],Scalar(255,0,0),4);
            }
            if ( i==3)
            {
               line(imgLines,points2f1[i],points2f2[i],Scalar(255,255,255),4);
            }


            points2f1[i].x = points2f2[i].x;
            points2f1[i].y = points2f2[i].y;
```

```
            }


        }


        bgr_frame2 = bgr_frame2 + imgLines; // adding the original image and the image with lines



        imshow("eee",bgr_frame2); // displaying the image after addition


        writer << bgr_frame2; // saving bgr_frame2 to writer

        //    if( c == 27)  break;


        char c = cv::waitKey(fps);

        capture >> bgr_frame2;   // capturing a new frame from the input video

        cv::cvtColor( bgr_frame2, frame_gray2, cv::COLOR_BGR2GRAY); //converting bgr_frame2 to gray image and storing in frame_gray2

        f2d->detect(frame_gray2, keypoints2);// detecting keypoints for frame_gray2 using SURF algorithm

        KeyPoint::convert(keypoints2,points2f2,keypointIndexes2);//converting keypoints2 to Points2f type

        //   if( c == 27)  break;

    }

        t = ((double)getTickCount() - t)/getTickFrequency();

        std::cout << "Times passed in seconds: " << t << std::endl;



    return 0;


}
```

Input:


Output: