Referências Bibliográficas

[01] AUGUSTO, C. E. L.. **Uma Infra-Estrutura para a Execução Distribuída de Componentes de Software**. PhD thesis, PUC-Rio, Brasil, 2008. 3.1

[02] BARTON, D.. **Radar system analysis and modeling**. Artech House, 2005. 2.2.3, A.1

[03] BELL LABORATORIES. **Abm research and development at bell laboratories: Project history**. Technical report, 1975. 2.3.3

[04] CARVALHO, B.; OTHERS. **Desdobramentos tecnológicos no desenvolvimento do radar saber m60**. 2008. 2.3.2, 4.1, 4.2

[05] CHUNG, C.. **Simulation Modeling Handbook: A Practical Approach**. CRC Press, 2004. 1

[06] COUNCILL, B.; HEINEMAN, G. T.. **Definition of a Software Component and its Elements**. 2001. 3

[07] ELTA SYSTEMS. **Cat. No. 6210**. Israel, 2007. 2.3.2

[08] FASSINO, J. S. J. L. J. M. G.. **Think: A software framework for component-based operating system kernels**. 2002. 6

[09] FORD, B.; BACK, G.; BENSON, G.; LEPREAU, J.; LIN, A. ; SHIVERS, O.. **The flux OSKit: A substrate for kernel and language research**. In: PROCEEDINGS OF THE SIXTEENTH ACM SYMPOSIUM ON OPERATING SYSTEM PRINCIPLES (16TH SOSP'97), OPERATING SYSTEM REVIEW (OSR), p. 38–51, Saint-Malo, France, Oct. 1997. ACM SIGOPS. Published as Proceedings of the Sixteenth ACM Symposium on Operating System Principles (16th SOSP'97), Operating System Review (OSR), volume 31, number 5. 6

[10] HELANDER, J. F. A.. **Mmlite: A highly componentized system architecture**. 1998. 6

[11] HORN, C.. **The orbix architecture**. Technical report, tech. rep., IONA Technologies, 1993. 4.2

[12] IERUSALIMSCHY, R.. **Programming in Lua**. Lua.org, 2003. 3.1.1

[13] KOPP, C.. **Active and semiactive radar missile guidance**, 2005. 2.3.4, 2.3.4

[14] LIMA, A. G. M.; MEDELLA, A.; RITA, V. A. S.; CRUVINEL, E. ; BUENO, M.. **Desenvolvimento de um radar de identificação de alvos amigo-inimigo**. 2008. 4.1

[15] LOCKHEED MARTIN. **AN/FPS-117 - Long Range Solid-State Radar**. USA, 2011. 2.3.3

[16] LOCKHEED MARTIN. **Cat. B103 7/03**. USA. 2.3.2

[17] MAGOUTIS, K.; BRUSTOLONI, J. C.; GABBER, E.; NG, W. T. ; SILBERSCHATZ, A.. **Building appliances out of components using pebble**, Sept. 26 2000. 6

[18] MAHAFSA, B. R.. **Radar signal analysis and processing using Matlab**. 2008. 1

[19] MICROSOFT CORPORATION. **Com - component object model**, 1993. 3, 6

[20] MICROSOFT CORPORATION. **Microsoft .net platform**. 3

[21] M.I. CORBA. **The mico project page: http://www.mico.org, http://www.fpx.de**. 4.2, B.1

[22] NLNETLABS. **Nsd: Name server daemon**, 2011. B.1

[23] NYQUIST, H.. **Thermal agitation of electric charge in conductors**. Physical Review, 32(1):110–113, 1928. 2.2.1

[24] OMG - OBJECT MANAGEMENT GROUP. **Common Object Request Broker Architecture: Core Specification**. USA, 2001. 3, 3.1.1, 6, B.1

[25] PAUL K. DAVIS, R. H. A.. **Improving the composability of Department of Defense models and simulations**. RAND Corporation, 2004. 1

[26] RAYTHEON. **Patriot Air and Missile Defense System**, 2011. 2.3.3

[27] ROHLING, H.. **Radar cfar thresholding in clutter and multiple target situations**. Aerospace and Electronic Systems, IEEE Transactions on, (04):608–621, 1983. 2.2.3

[28] SAWICKI, D. S.. **Police Traffic Speed Radar Handbook**. USA, 2011. 2.3.1, 5.1.1

[29] SCHLEHER, D.. **Automatic detection and radar data processing**. Artech House on Demand, 1980. 2.2.3

[30] SKOLNIK, M. I.. **Introduction to Radar Systems**. McGraw Hill, 2001. 1, 2.2.4

[31] SUN MICROSYSTEMS. **Javabeans 1.01 specification**, 1997. 3, 6

[32] SUN MICROSYSTEMS. **Enterprise javabeans technology v3.0**, 2006. 3

[33] SZYPERSKI, C.. **Component Software: BeyondObject-Oriented Programming**. 2d edition, 2002. 3

[34] TECGRAF. **Scs: Software component system**, 2008. 3.1, 4.2, B.1

[35] THALES - AEROSPACE DIVISION. **Active Electronically Scanned Array (AESA) Radar**, 2006. 2.3.3

[36] U.S. AIR FORCE. **Pave paws radar system**, 2010. 2.3.3

[37] WINTER, M.; GENLER, T.; DUCASSE, S.; WUYTS, R.; AREVALO, G.; MULLER, P.; STICH, C. ; SCHONHAGE, B.. **Components for embedded software - the pecos approach**, Oct. 22 2002. 6

[38] VAN OMMERING, R.; VAN DER LINDEN, F.; JEFF, K. ; MAGEE, J.. **The koala component model for consumer electronics software**. Computer, 33(3):78–85, Mar. 2000. 6

A
A Biblioteca

A biblioteca proposta utiliza como base um sistema de componentes que naturalmente atende aos requisitos de modularidade exigidos. O sistema de componentes escolhido permite a sua execução distribuída o que possibilita a interação do simulador com componentes reais de um radar e também permite a conexão em tempo de execução de seus componentes. Para a biblioteca proposta forão implementados diversos componentes que permitam simular diferentes cenários de simulação por meio de suas conexões e configuração.

A divisão proposta para a biblioteca consiste em quatro camadas conforme a Figura A.1. Estas camadas podem ser diretamente mapeadas nas entidades da Figura 2.2 que mostra uma visão de um sistema radar. Na camada de ambiente foram reunidos os componentes relativos a simulação dos vetores e dos efeitos da propagação eletromagnética no meio. A camada de *hardware* do radar reflete a parte concreta com as entidades referentes a antena, ao receptor e ao motor. A camada de *software* do radar pode subdivida em duas camadas uma de processamento de sinais e uma de processamento de dados, estas camadas contemplam os componentes abstratos de programação. E por fim uma camada de suporte para prestar apoio a operações de registro (*log)* e operações de validação como a análise estatística dos dados processados.

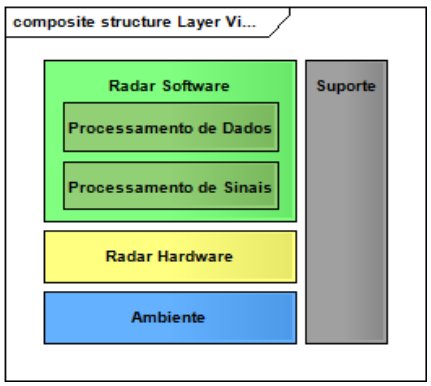

Figura A.1: Divisão lógica dos componentes em camadas

Com o intuito de reduzir o número de componentes, diminuir a complexidade e diminuir o tempo de programação foram tomadas algumas decisões de projeto. Como a acumulação das responsabilidades do componente meio

de propagação da onda no componente antena. Outra simplificação foi a não implementação dos componentes relativos a transmissão, neste caso adotou-se uma modelagem que oferece sempre os mesmos resultados para os pulsos de transmissão, assim os componentes de transmissão foram omitidos e os resultados esperados da etapa de transmissão foram gerados uma vez e re-utilizados. Os componentes de cada camada e as suas responsabilidades serão apresentados a seguir:

A.1 Ambiente

A camada de ambiente da biblioteca de *software* possui componentes que simulam os efeitos externos a um radar. Ela é composta por dois principais componentes, o componente vetor e o componente meio de propagação da onda.

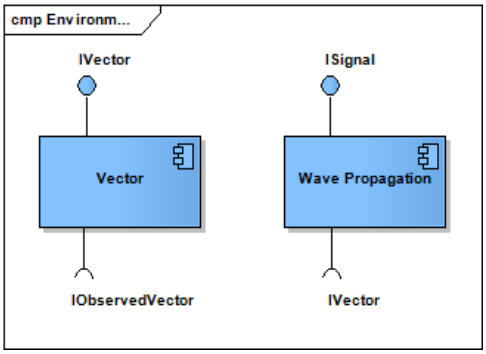

Figura A.2: Componentes da camada ambiente

O componente vetor simula as características e o comportamento do objeto de interesse que se deseja detectar. Neste componente foram modelados os efeitos de reflexão de um sinal, por exemplo a reflexão na fuselagem de um avião ou num automóvel. As principais características de um vetor são a sua posição, velocidade, aceleração e área de reflexão. Na Figura A.2 pode-se observar o componente vetor com a interface *IVector* com as funções de configuração *setId*, *setRange, setAzimuth, setElevation, setVelocity* e *setCrossSection* que configuram respectivamente o identificador do vetor, a distância, o azimute, a elevação, a velocidade e a área de reflexão do vetor, também conhecida como seção reta radar do vetor; e as funções *move*, *getPosition* e *getFeatures* que, respectivamente, atualiza as características cinéticas do vetor, informa a posição do vetor relativa a uma referência e informa as características do vetor relativas a uma posição, na Figura A.3 pode-se observar as interfaces e suas funções com mais detalhes. E Na Figura A.2 pode-se observar também a interface requerida *IObservedVectorReport*, que recebe os dados observados

do vetor para comparação estatística com os dados medidos, é acionada a cada chamada da função *move*.

De acordo com a decisão de projeto que visa diminuir o número de componentes e a complexidade da programação, o componente meio de propagação da onda foi incorporado ao componente antena. Mas, para efeitos didáticos, ele será explicado separadamente.

O componente meio de propagação da onda recebe o pulso de transmissão e monta o sinal de recepção aplicando os efeitos de reflexão do pulso nos vetores, os efeitos de sobreposição de reflexões, os efeitos de perda de potência com a distância de propagação da onda e o efeito de ambiguidade em distância (02) nas reflexões, que é percebido pelo recebimento de pulsos transmitidos anteriormente. Na Figura A.2 pode-se observar o componente meio de propagação da onda, a interface *IWavePropagation* com as funções *setCarrierFrequency, setPulse, setSamplingRate, setSignalSize e setRangeAmbiguityDeep* que configuram respectivamente a frequência da portadora do sinal de transmissão, a taxa de amostragem do sinal, o número de amostras do sinal de recepção e a profundidade de ambiguidade em distância que será tolerada; a interface requerida *Ivector* com as funções *getVectorPosition* e *getVectorFeatures* que coletam as informações dos vetores para montar o sinal de reflexão e a interface *ISignal* com a função *getSignal* que disponibiliza o sinal de recepção.

A.2 Hardware do Radar

A camada de *hardware* da biblioteca é composta por componentes que simulam o *hardware* de um radar. Ela é composta pelos componentes antena, conformador de feixe, somador de sinais, receptor e motor.

Por decisão de projeto para diminuir a complexidade, sem que haja perdas para os objetivos propostos, o componente antena acumula as funções do componente antena de transmissão, meio de propagação da onda, antena de recepção, defasador e algumas funções do transmissor, ou seja, o componente antena pode ser visto como uma composição destes outros componentes. O componente antena recebe o sinal refletido, aplica os ganhos de potência de transmissão e os ganhos dos diagramas de transmissão e recepção da antena, e defasa o sinal proporcionalmente à distância do vetor e, se for o caso, também adiciona a defasagem de sinal controlada pelo componente *BeamFormControl*.

Na Figura A.4 é ilustrado o componente *antenna* com a interface *IAntenna* que possui as funções, também exibidas na Figura A.6, setAzimuth, setElevation, setReceptionLocation, setTransmissionLocation, setChannel, setAzimuthReceptionDiagram, setElevationReceptionDiagram, setGain, se-

```
module radar
{
 interface ISignal
 {
  void getSignal(out SSignal ss);
 };
 module environment
 {
  interface IVector
  {
   void move(in float dtime);
   Point getPosition(in Point reference);
   VectorFeatures getFeatures(in Point reference);
   void setId(in short id);
   void setRange(in float range);
   void setAzimuth(in float azimuth);
   void setElevation(in float elevation);
   void setVelocity(in float radialVelocity);
   void setCrossSection(in float crossSection);
   void setRangeAmbiguityDeep(in short deep);
  };
    interface IWavePropagation : ISignal
    {
      void setCarrierFrequency(in float carrierFrequency);
      void setPulse(in ComplexSequence pulse);
      void setSamplingRate(in float samplingRate);
      void setSignalSize(in long signalSize);
    };
  };
};
```

Figura A.3: Trecho de código da linguagem de descrição de interfaces

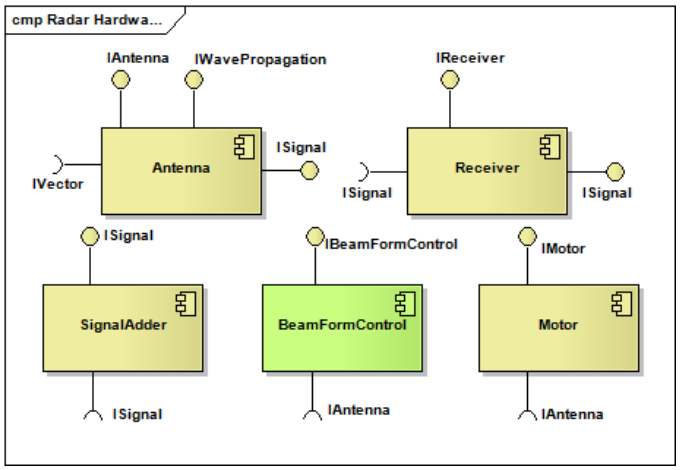

Figura A.4: Componentes da camada de *hardware*

tAzimuthTransmissionDiagram, setElevationTransmissionDiagram, setPower-Transmission, setBeamForm e getGain, que, respectivamente, são responsáveis pela configuração do azimute de transmissão/recepção da antena, da elevação de transmissão/recepção da antena, da posição de recepção no espaço, da

posição de transmissão no espaço, do identificador do canal de recepção, do diagrama de recepção em azimute, do diagrama de recepção em elevação, do ganho da antena, do diagrama de transmissão em azimute, do diagrama de transmissão em elevação, da potência de transmissão, das características de fase e amplitude do conformador de feixe, e a função que retorna o ganho dos diagramas da antena dada uma direção. A interface *IWavePropagation* que acumula as funções relativas ao meio de propagação da onda. E a interface *ISignal* que prove o sinal de recepção da antena.

Na Figura A.5 o componente antena é exibido como uma composição de demais componentes acumulando suas funcionalidades, mas como a implementação em linguagem C do modelo de componentes utilizado, o SCS, ainda não permite a composição de terceiros para um componente, esta modelagem de composição não foi implementada utilizando técnicas de composição previstas pelo modelo de componentes de *software*.

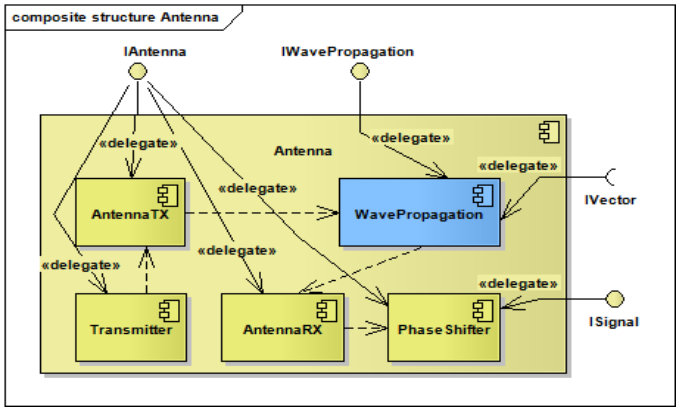

Figura A.5: Componente antena como uma composição de terceiros

O componente *receiver*, ilustrado na Figura A.4, simula o processamento de sinal analógico como filtragem do sinal, conversão para sinal digital(quantização) e reamostragem; e também insere ruído no sinal recebido. Possuindo interface *IReceiver,* exibida na Figura A.6, com as funções *setSamplingRate, setCarrierFrequency, setPassBand, setQuantizationBits, setNoiseFigureDb, setSignalSize e getSignalReplica* que configuram a taxa de amostragem do sinal, a frequência da portadora do sinal, a banda de frequência do filtro, o número de bits da quantização, a figura de ruído, o número de amostras do sinal digitalizado e a função que disponibiliza o sinal de referência. E a interface requerida *ISignal* que recebe o sinal e a interface *ISignal* que prove o sinal tratado.

O componente somador de sinais(*signal adder*) pode coletar vários sinais em seu receptáculo e prover a soma deles em sua faceta. Realizando a simples adição dos sinais, é utilizado para promover ganho do sinal de reflexão em

relação ao ruído ou é utilizado em conjunto com o conformador de feixe. As suas interfaces são a *ISignalAdder* com as funções *setAzimuth* e *setElevation* que configuram o azimute e a elevação da direção proveniente do sinal, para o caso do direcionamento de feixe pelo conformador de feixe, e a função setChannel que configura um novo identificador para o canal resultante da soma dos demais.

O componente conformador de feixe(*BeamformControl*) recebe em sua faceta *IBeamformControl* na função *setDirection* a direção de posicionamento do feixe e realiza os cálculos das fases e pesos para os sinais provenientes das antenas. As antenas conectadas em seu receptáculo *IAntenna* recebem a fase e o peso do sinal que devem aplicar. Na interface *IBeamformControl* também existem as funções *addAntenna, setRowCol, setCarrierFrequency, setHammingAzimuth e setHammingElevation* para adicionar as características das antenas, para configurar o número de linhas e colunas, a frequência da portadora, a curva de *hamming* para o azimute e para a elevação respectivamente.

E por fim, o componente motor simula o giro de um motor possuindo uma interface IMotor com as funções *rotate, getAngle, setAngle e setAngularVelocity* que, respectivamente, atualiza a posição angular do motor, retorna a posição angular do motor, configura a posição angular do motor e configura a velocidade de rotação do motor. E uma interface requerida *IAntenna* que informa aos componentes conectados a direção angular atual.

A.3 Software do Radar

Na camada de *software* da biblioteca encontram-se os componentes de *software* de um radar. Esta camada está dividida em duas camadas: uma de processamento de sinais e outra de processamento de dados. Em uma visão de componentes de *software* esta camada poderia ser uma composição de um componente de processamento de sinais e um componente de processamento de dados. Devido ao sistema de componentes utilizada não ter suporte a composição de componentes não foi implementada. A seguir serão apresentadas as subcamadas de processamento de sinais e de processamento de dados ilustradas na Figura A.7.

A.3.1 Processamento de Sinais

A camada de processamento de sinais modifica o sinal facilitando, permitindo a futura extração da informação útil contida nele. Ela é composta pelos componentes de compressão de sinal e de detecção da informação útil.

```
module radar {
 interface ISignal {
  void getSignal(out SSignal as);
 };
 module radarHardware {
  interface IAntenna : ISignal, environment::IWavePropagation {
   float getGain(in float azimuth,in float elevation);
   void setAzimuth(in float azimuth);
   void setElevation(in float elevation);
   void setLocation(in float positionx, in float positiony, in
      float positionz);
   void setLocationTransmission(in float positionx, in float
      positiony, in float positionz);
   void setChannel(in short channel);
   void setAzimuthReceptionDiagram(in FloatSequence gainDB, in
      FloatSequence angleRef);
   void setElevationReceptionDiagram(in FloatSequence gainDB, in
      FloatSequence angleRef);
   void setAzimuthTransmissionDiagram(in FloatSequence gainDB, in
      FloatSequence angleRef);
   void setElevationTransmissionDiagram(in FloatSequence gainDB,
      in FloatSequence angleRef);
   void setGain(in float transmissionGainDb, in float
      receptionGainDb, in float lossDb);
   void setPowerTransmission(in float powerTransmission);
   void setBeamForm(in ComplexFloat phasor);
  };
  interface IMotor {
   void rotate(in float time);
   float getAngle();
   void setAngle(in float angle);
   void setAngularVelocity(in float angle);
  };
  interface IBeamformControl {
   void setDirection(in float azimuth, in float elevation);
   void addAntenna(in float x, in float y, in float z, in short
      channel, in short conection, in short row, in short col);
   void setRowCol(in short row, in short col);
   void setCarrierFrequency(in float carrierFrequency);
   void setHammingAzimuth(in boolean hamming);
   void setHammingElevation(in boolean hamming);
  };
  interface ISignalAdder : ISignal {
   void setAzimuth(in float azimuth);
   void setElevation(in float elevation);
   void setChannel(in short channel);
  };
  interface IReceiver : ISignal {
   void setSamplingRate(in float samplingRate);
   void setCarrierFrequency(in float carrierFrequency);
   void setPassBand(in float passBand);
   void setQuantizationBits(in short quantizationBits);
   void setNoiseFigureDb(in float noiseFigureDb);
   void setSignalSize(in long signalSize);
   void getSignalReplica(in ComplexSequence sin, out
      ComplexSequence sout);
  };
 };
};
```

Figura A.6: Trecho de código da linguagem de descrição de interfaces

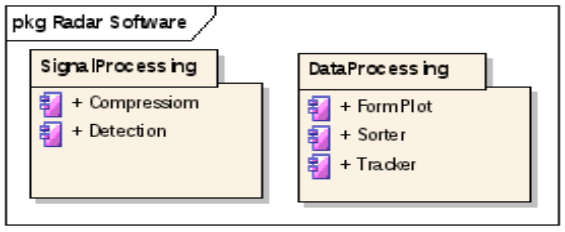

Figura A.7: Camada de *software* dividida em duas camadas de processamento

O componente de compressão de sinal realiza a correlação de sinal recebido com o sinal transmitido, no caso um pulso. A Figura A.8 ilustra o componente de compressão de sinal que implementa a interface *ICompression* com a função *setReplicaPulse* que atribui o pulso transmitido para realizar a correta correlação, a interface *ISignal* que disponibiliza o sinal comprimido, e a interface requerida *ISignal* que recebe o sinal a ser comprimido.

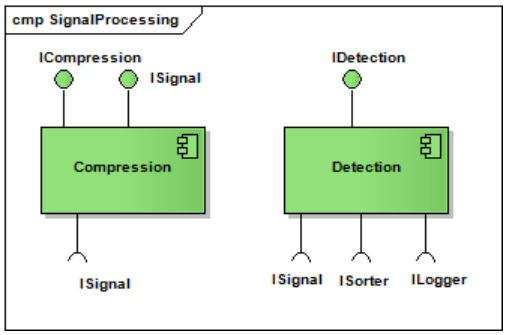

Figura A.8: Componentes da camada de processamento de sinais

O componente de detecção acumula sinais no domínio do tempo, aplica uma transformada matemática e aplica um algorítimo de detecção. O componente é ilustrado na Figura A.8, apresentando a interface *IDetection* com as funções *setThreshold* , que configura o limiar de detecção, *step* , que executa um ciclo de simulação e *detect*, que recolhe os sinais acumulados e aplica o processo de detecção. Na Figura A.8 são ilustradas também as interfaces requeridas *ISignal*, que recebe os sinais que serão acumulados, *ISorter*, que envia os sinais de reflexão detectados e *ILogger*, que é a interface para registro de *log*.

A.3.2 Processamento de dados

A camada de processamento de dados extrai informação útil dos dados selecionados. Para tal, primeiro ela ordena os dados e depois extrai a informação que compreende as características cinemáticas dos vetores e por fim pode montar a trajetória destas informações coletadas.

```
module radar
{
 interface ISignal
 {
  void getSignal(out SSignal as);
 };
 module radarSoftware
 {
  module signalProcessing
  {
   interface ICompression : ISignal
   {
    void setReplicaPulse(in ComplexSequence pulse);
   };

   interface IDetection
   {
    short step(in float time);
    boolean detect();
    void setThreshold(in double threshold);
   };
  };
 };
};
```

Figura A.9: Trecho de código da linguagem de descrição de interfaces

Os componentes desta camada, ilustrados na Figura A.10, são o *Sorter*, o *FormPlot* e o *Tracker,* e suas interfaces são exibidas na Figura A.11. O componente *Sorter* ordena, armazena os dados detectados e seleciona o conjunto de dados referente a um vetor para o cálculo das informações cinemáticas do vetor. Ele implementa a interface *ISorter* que possui as funções *setData* que recebe os dados detectados, *setMinimumAge* que configura o número mínimo de dados para se calcular as informações de um vetor, *setMaximumAge* que configura o número máximo de dados para se calcular as informações de um vetor, e *setMaximumWaiting* que configura o número máximo de ciclos para esperar dados antes de se tentar calcular as informações de um vetor. E requer as interfaces *IFormPlot* que envia os dados selecionada para se calcular as informações cinemáticas de um vetor e *ILogger* que envia dados para registro de *log*.

O componente *FormPlot* calcula as informações de posição e velocidade dos vetores. Ele implementa a interface *IFormPlot* que possui as funções *buildPlot* que recebe os dados para o cálculo das informações do vetor, *setInterferometricAzimuthCalculosMap*, que mapeia quais canais serão utilizados no cálculo do azimute pela técnica da interferometria, *setInterferometricElevationCalculosMap* que mapeia quais canais serão utilizados no cálculo da

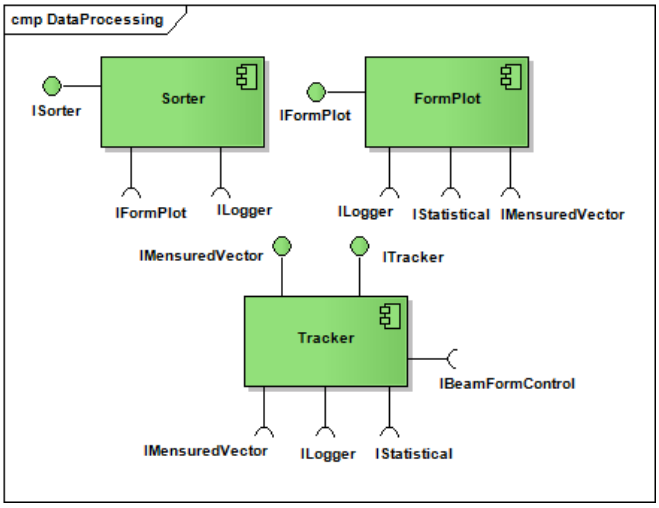

Figura A.10: Componentes da camada de processamento de dados

elevação pela técnica da interferometria, *setRangeCalculosMap* que mapeia quais canais serão utilizados no cálculo da distância, *setVelocityCalculosMap* que mapeia quais canais serão utilizados no cálculo da velocidade, *setMonopulseAzimuthCalculosMap* que mapeia quais canais serão utilizados no cálculo do azimute pela técnica do mono pulso, *setTransmissionFrequency* que configura a frequência de transmissão, *setSamplingRate* que configura a taxa de amostragem do sinal, e *setPulseRateFrequency* que configura a frequência de repetição de pulso. E requer as interfaces *IMensuredVector* que envia as informações calculadas de um vetor, a interface ILogger que faz registro de log e a interface IStatistical que envia as informações calculadas de um vetor para cálculos estatísticos.

E por fim o componente *Tracker* que monta as trajetórias dos vetores detectados. Ele recebe os dados referentes a um vetor, procura se esses dados pertencem a uma trajetória existente e faz a associação, caso contrário uma nova trajetória é criada. Ele disponibiliza os dados de um vetor, uma trajetória ou a direção em azimute e elevação de um vetor, que é útil para um radar de acompanhamento. Ele implementa a interface *ITracker* que possui as funções *setTimeOfPrediction* que configura uma estimativa do tempo de predição, *setErrorTolerance* que configura o erro normalizado máximo tolerado, *setTimeTolerance*, *setRangeResolution*, *setAzimuthResolution*, *setElevationResolution*, *setVelocityResolution* que configuram a resolução de distância, azimute, elevação e velocidade que serve para normalizar as medidas. E requer as interfaces *IMensuredVector* que recebe as informações de um vetor, *ILogger* que faz registro de *log*, *IStatistical* que recebe as informações de um vetor para cálculos estatísticos e *IBeamFormControl* que recebe as informações de azimute e elevação para o direcionamento da transmissão/recepção.

```
1  module radar {
2   interface IMensuredVector {
3    void addVector(in RadarVector v);
4   };
5   module dataProcessing {
6    interface ISorter {
7     void setData(in SSignalMatrix blocks);
8     void setMinimumAge(in short minimumAge);
9     void setMaximumAge(in short maximumAge);
10    void setMaximumWaiting(in short maximumWaiting
11   };
12   interface IFormPlot {
13    void buildPlot(in SSignalMatrix blocks);
14    void setInterferometricAzimuthCalculosMap(in radar::
          ShortSequence idLeft, in radar::ShortSequence idRight, in
          radar::FloatSequence distanceBetweenCenters, in boolean
          calculate);
15    void setInterferometricElevationCalculosMap(in radar::
          ShortSequence idTop, in radar::ShortSequence idBottom, in
          radar::FloatSequence distanceBetweenCenters,in boolean
          calculate);
16    void setRangeCalculosMap(in radar::ShortSequence id, in boolean
           calculate);
17    void setVelocityCalculosMap(in radar::ShortSequence id, in
          boolean calculate);
18    void setMonopulseAzimuthCalculosMap(in radar::ShortSequence id,
           in boolean calculate);
19    void setTransmissionFrequency(in float transmissionFrequency);
20    void setPulseRateFrequency(in float pulseRateFrequency);
21    void setSamplingRate(in float samplingRate);
22   };
23   interface ITracker : IMensuredVector {
24    void setTimeOfPrediction(in float timeOfPrediction);
25    void setErrorTolerance(in float errorTolerance);
26    void setTimeTolerance(in float timeTolerance);
27    void setRangeResolution(in float rangeResolution);
28    void setAzimuthResolution(in float azimuthResolution);
29    void setElevationResolution(in float elevationResolution);
30    void setVelocityResolution(in float velocityResolution);
31   };
32  };
33 };
```

Figura A.11: Trecho de código da linguagem de descrição de interfaces

A.3.3 Suporte

A camada de suporte oferece serviços para monitoramento das operações dos componentes e para comparação dos dados determinados pelo processamento com os dados observados dos vetores. Esta camada é composta pelo componente *Logger* responsável pelo registro das informações recebidas na faceta *report* e pelo componente *Statistics* que faz análise estatística comparando os dados calculados e os dados observados dos vetores. Os componentes estão ilustrados na Figura A.12 e as interfaces são exibidas na Figura A.13.

O componente *Logger* implementa a interface *ILogger* que possui as funções *setFileNameSignal* que configura o nome do arquivo binário onde são escritos os dados do sinal, *writeSignal* que recebe o sinal a ser escrito no arquivo binário, *setFileNameString* que configura o nome do arquivo onde são escritos os textos de *log*, *printfString* que recebe os textos que serão escritos no arquivo de *log*, *setFileNameObservedVector e setFileNameMeasuredVector* que configuram os arquivos onde são escritos os dados observados e calculados dos vetores. E implementa as interfaces *IMensuredVector, IObservedVector* que recebem os dados calculados e observados dos vetores para serem escritos nos arquivos.

O componente *Statistics* implementa a interface *IStatistical* que possui as funções *setErrorTolerance* que configura o erro normalizado máximo tolerado, *setTimeTolerance* que configura o erro máximo de tempo entre as amostras observadas e calculadas, *setRangeResolution, setAzimuthResolution, setElevationResolution, setVelocityResolution* que configuram a resolução de distância, azimute, elevação e velocidade que serve para normalizar as medidas. E implementa as interfaces *IMensuredVector* que recebe as informações calculadas de um vetor, e*IObservedVector* que recebe as informações observadas de um vetor. E requer a interface *ILogger* que faz registro de *log*.

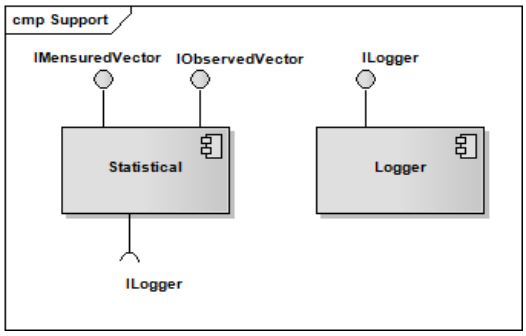

Figura A.12: Componentes da camada de suporte

```
module radar
{
 struct PolarPoint
 {
  float range;
  float azimuth;
  float elevation;
 };
 struct RadarVector
 {
  short id;
  float time;
  PolarPoint position;
  float radialVelocity;
 };
 interface IMensuredVector
 {
  void addMensuredVector(in RadarVector mv);
 };
 interface IObservedVector
 {
  void addObservedVector(in RadarVector ov);
 };
 module support
 {
  interface IStatistical : IMensuredVector, IObservedVector
  {
   void setErrorTolerance(in float errorTolerance);
   void setTimeTolerance(in float timeTolerance);
   void setRangeResolution(in float rangeResolution);
   void setAzimuthResolution(in float azimuthResolution);
   void setElevationResolution(in float elevationResolution);
   void setVelocityResolution(in float velocityResolution);
  };
  interface ILogger : IMensuredVector, IObservedVector
  {
   void writeSignal(in SSignal ss) raises (WriteFile);
   void printfString(in string str, in boolean monitor);
   void setFileNameSignal(in string name);
   void setFileNameObservedVector(in string name);
   void setFileNameMeasuredVector(in string name);
   void setFileNameSring(in string name);
  };
 };
};
```

Figura A.13: Trecho de código da linguagem de descrição de interfaces

B
Manual do Usuário

A seguir será apresentado o manual do usuário, onde estão os procedimentos para se criar uma simulação e executa-la. Na seção B.1 serão apresentados os requisitos para se executar uma simulação e como os componentes são disponibilizados. Na seção B.1.1 são apresentados os passos para se montar uma simulação. E, por fim, na seção B.2 é explicado como se executa a simulação.

B.1
A biblioteca de componentes e requisitos

Nesta seção serão apresentados os requisitos para se executar uma simulação e como os componentes são disponibilizados. Para usar a biblioteca e executar uma simulção é necessário ter instalado o Sistema de Componentes de *Software* SCS (34) na linguagem C, ter instalado o MICO (21) que é uma implementação da arquitetura CORBA (24) e ter instalado o servidor de nomes NSD (22).

A bibliotéca de componentes é disponibilizada como um conjunto de arquivos, um arquivo para cada componente. Cada arquivo recebe o nome do componente que ele representa seguido de sua sua verção, por exemplo para o componente que simula o motor tem-se o arquivo *motorComponent100.so*, onde o número 100 representa a sua versão.

B.1.1
Montando uma simulação

Nesta seção serão apresentados os passos para se montar uma simulação. Para se criar uma simulação é necessário gerar um código *C*, ilustrado na Figura B.1, onde:

1. O *middleware* de comunição CORBA, o MICO, é inicializado;
2. O servidor de nomes é referenciado;
3. A infraestrutura de execução do modelo de componentes é carregada;
4. Os componentes são carregados, conectados, configurados e inicializados;

5. E os serviços(providos pelos componentes) necessários para se executar a simulação são requisitados.

B.2 Execução

Por fim, nesta seção é explicado como se executa a simulação. O primeiro passo para se executar a simulação é confirmar a existência de todos os arquivos de biblioteca dos componentes que serão carregados e a existência do arquivo executável proveniente da compilação do arquivo em C que monta a simulação pretendida. O segundo passo é a execução do servidor de nomes. Na sequência deve-se executar a infraestrutura de execução do modelo de componentes e por fim executar o arquivo da simulação. Esses comandos são ilustrados na Figura B.2.

```
#include <stdio.h>
#include <string.h>
#include <pthread.h>
#include <coss/CosNaming.h>

#include <radar.h>
#include <scs.h>
#include <deployment.h>

int main(int argc, char * argv[])
{
 /*
 * LOADING CORBA MIDDLEWARE E NAME SERVER
 */
 CORBA::ORB * orb = CORBA::ORB_init(argc, argv);
 CORBA::Object_var objVar;
 CosNaming::NamingContext_var rootContext;
 objVar = orb->resolve_initial_references("NameService");
 rootContext = CosNaming::NamingContext::_narrow(objVar);
 CORBA::Object_var objIC;
 /*
 * LOADING THE CONTAINER COMPONENT
 */
 CosNaming::Name_var tmpName;
 tmpName = new CosNaming::Name(1);
 tmpName->length(1);
 tmpName[0].id = CORBA::string_dup("Container");
 tmpName[0].kind = CORBA::string_dup("");
 objIC = rootContext->resolve(tmpName);
 scs::core::IComponent_var iComponentFacet = scs::core::IComponent
     ::_narrow(objIC);
 CORBA::Object_var objComponentLoader = iComponentFacet->
     getFacetByName("ComponentLoader");
 scs::container::ComponentLoader * componentLoader = scs::
     container::ComponentLoader::_narrow(objComponentLoader);
 scs::container::StringSeq args;
 args.length(0);
 /*
 * LOADING APLICATION COMPONENTS
 */
 scs::core::ComponentId componentId;
 componentId.name = "AerialVectorComponent";
 componentId.major_version = '1';
 componentId.minor_version = '0';
 componentId.patch_version = '0';
 componentId.platform_spec = "none";
  scs::container::ComponentHandle * aerialVectorHandle1 =
      componentLoader->load(componentId, args);
 scs::container::ComponentHandle * aerialVectorHandle2 =
     componentLoader->load(componentId, args);
 scs::container::ComponentHandle * aerialVectorHandle3 =
     componentLoader->load(componentId, args);
 scs::container::ComponentHandle * aerialVectorHandle4 =
     componentLoader->load(componentId, args);
 componentId.name = "AntennaComponent";
 componentId.major_version = '1';
 componentId.minor_version = '0';
 componentId.patch_version = '0';
 componentId.platform_spec = "none";
 scs::container::ComponentHandle * antennaHandle1 =
     componentLoader->load(componentId, args);
```

```
54  componentId.name = scs::radar::LOGGER_COMPONENT
55  componentId.major_version = '1';
56  componentId.minor_version = '0';
57  componentId.patch_version = '0';
58  componentId.platform_spec = "none";
59  scs::container::ComponentHandle * loggerHandle1 = componentLoader
        ->load(componentId, args);
60  /*
61  * GETTING THE COMPONENT FACETS
62  */
63  CORBA::Object_var objaerialVectorFac1 = aerialVectorHandle1->cmp
        ->getFacetByName("AerialVectorFacet");
64  CORBA::Object_var objaerialVectorFac2 = aerialVectorHandle2->cmp
        ->getFacetByName("AerialVectorFacet");
65  CORBA::Object_var objaerialVectorFac3 = aerialVectorHandle3->cmp
        ->getFacetByName("AerialVectorFacet");
66  CORBA::Object_var objaerialVectorFac4 = aerialVectorHandle4->cmp
        ->getFacetByName("AerialVectorFacet");
67  CORBA::Object_var objAntennaFac1 = antennaHandle1->cmp->
        getFacetByName("AntennaFacet");
68  CORBA::Object_var objLoggerFac1 = loggerHandle1->cmp->
        getFacetByName(scs::radar::LOGGER_FACETA);
69  scs::radar::environment::IAerialVector * aerialVectorFac1 = scs::
        radar::environment::IAerialVector::_narrow(objaerialVectorFac1
        );
70  scs::radar::environment::IAerialVector * aerialVectorFac2 = scs::
        radar::environment::IAerialVector::_narrow(objaerialVectorFac2
        );
71  scs::radar::environment::IAerialVector * aerialVectorFac3 = scs::
        radar::environment::IAerialVector::_narrow(objaerialVectorFac3
        );
72  scs::radar::environment::IAerialVector * aerialVectorFac4 = scs::
        radar::environment::IAerialVector::_narrow(objaerialVectorFac4
        );
73  scs::radar::radarHardware::IAntenna * antennaFac1 = scs::radar::
        radarHardware::IAntenna::_narrow(objAntennaFac1);
74  scs::radar::assistance::ILogger * loggerFac1 = scs::radar::
        assistance::ILogger::_narrow(objLoggerFac1);
75  /*
76  * GETTING THE COMPONENT RECEPTACLES
77  */;
78  CORBA::Object_var objAerialVectorRec1 = aerialVectorHandle1->cmp
        ->getFacetByName("IReceptacles");
79  CORBA::Object_var objAerialVectorRec2 = aerialVectorHandle2->cmp
        ->getFacetByName("IReceptacles");
80  CORBA::Object_var objAerialVectorRec3 = aerialVectorHandle3->cmp
        ->getFacetByName("IReceptacles");
81  CORBA::Object_var objAerialVectorRec4 = aerialVectorHandle4->cmp
        ->getFacetByName("IReceptacles");
82  CORBA::Object_var objAntennaRec1 = antennaHandle1->cmp->
        getFacetByName("IReceptacles");
83  scs::core::IReceptacles * aerialVectorRecpt1 = scs::core::
        IReceptacles::_narrow(objAerialVectorRec1);
84  scs::core::IReceptacles * aerialVectorRecpt2 = scs::core::
        IReceptacles::_narrow(objAerialVectorRec2);
85  scs::core::IReceptacles * aerialVectorRecpt3 = scs::core::
        IReceptacles::_narrow(objAerialVectorRec3);
86  scs::core::IReceptacles * aerialVectorRecpt4 = scs::core::
        IReceptacles::_narrow(objAerialVectorRec4);
87  scs::core::IReceptacles * antennaRecpt1 = scs::core::IReceptacles
        ::_narrow(objAntennaRec1);
```

```
88  /*
89  * BIDING THE COMPONENTS
90  */
91  aerialVectorRecpt1->connect(scs::radar::LOGGER_RECEPTACULO,
        loggerFac1);
92  aerialVectorRecpt2->connect(scs::radar::LOGGER_RECEPTACULO,
        loggerFac1);
93  aerialVectorRecpt3->connect(scs::radar::LOGGER_RECEPTACULO,
        loggerFac1);
94  aerialVectorRecpt4->connect(scs::radar::LOGGER_RECEPTACULO,
        loggerFac1);
95  antennaRecpt1->connect("AerialVectorRecept",aerialVectorFac1);
96  antennaRecpt1->connect("AerialVectorRecept",aerialVectorFac2);
97  antennaRecpt1->connect("AerialVectorRecept",aerialVectorFac3);
98  antennaRecpt1->connect("AerialVectorRecept",aerialVectorFac4);
99  statisticalRec1->connect(scs::radar::LOGGER_RECEPTACULO,
        loggerFac1);
100 /*
101 * STARTUPING THE COMPONENTS
102 */
103 aerialVectorHandle1->cmp->startup();
104 aerialVectorHandle2->cmp->startup();
105 aerialVectorHandle3->cmp->startup();
106 aerialVectorHandle4->cmp->startup();
107 antennaHandle1->cmp->startup();
108 loggerHandle1->cmp->startup();
109 /*
110 * SETTING THE COMPONENTS
111 */
112 loggerFac1->setFileNameObservedAerialVector("
        aerial_vector_observed.txt");
113 loggerFac1->setFileNameMeasuredAerialVector("
        aerial_vector_measured.txt");
114 loggerFac1->setFileNameSring("log.txt");
115 aerialVectorFac1->setId(1);
116 aerialVectorFac1->setRange(45000.0);
117 aerialVectorFac1->setAzimuth(25*M_PI/180);
118 aerialVectorFac1->setElevation(30.0*M_PI/180);
119 aerialVectorFac1->setRadialVelocity(-76.0);
120 aerialVectorFac1->setCrossSection(12.0);
121 scs::radar::ComplexSequence signal;
122 signal.length(m);
123 scs::radar::FloatSequence elevationDiagram, elevationAngleRef;
124 elevationDiagram.length(n);
125 elevationAngleRef.length(n);
126 scs::radar::FloatSequence azimuthDiagram, azimuthAngleRef;
127 azimuthDiagram.length(p);
128 azimuthAngleRef.length(p);
129 antennaFac1->setChannel(1);
130 antennaFac1->setAzimuthDiagramReception(azimuthDiagram,
        azimuthAngleRef);
131 antennaFac1->setElevationDiagramReception(elevationDiagram,
        elevationAngleRef);
132 antennaFac1->setAzimuthDiagramTransmission(azimuthDiagram,
        azimuthAngleRef);
133 antennaFac1->setElevationDiagramTransmission(elevationDiagram,
        elevationAngleRef);
134 antennaFac1->setAzimuth(0*M_PI/180);
135 antennaFac1->setElevation(7*M_PI/180);
136 antennaFac1->setLocation(0,0,0);
137 antennaFac1->setLocationTransmission(0,0,0);
```

```
138  antennaFac1->setSamplingRate(samplingRateFI);
139  antennaFac1->setGain(transmissionGainDb,receptionGainDb,lossDb);
140  antennaFac1->setPowerTransmission(powerTransmission);
141  antennaFac1->setCarrierFrequency(carrierFrequency);
142  antennaFac1->setPulse(signal);
143  antennaFac1->setSignalSize(sizeAd);
144  /*
145  * SERVICES TO PERFORM THE SIMULATION
146  */
147  float time = 0;
148  int burst = 100;
149  while( time <= maxTimeRevolution )
150  {
151   if( detectionFac1->step(time) == burst ) detectionFac1->detect()
        ;
152     aerialVectorFac1->move(1.0/pulseRateFrequency);
153     aerialVectorFac2->move(1.0/pulseRateFrequency);
154     aerialVectorFac3->move(1.0/pulseRateFrequency);
155     aerialVectorFac4->move(1.0/pulseRateFrequency);
156     time += 1.0/pulseRateFrequency;
157  }
158  orb->shutdown(1);
159  orb->destroy();
160  return 0;
161 }
```

Figura B.1: Exemplo simplificado de código de uma simulação

```
1 > nsd -ORBIIOPAddr inet:127.0.0.1:12456 &
2 > ./container_mico -ORBInitRef NameService=corbaloc::localhost.
    localdomain:12456/NameService &
3 > ./app-radar -ORBInitRef NameService=corbaloc::localhost.
    localdomain:12456/NameService
```

Figura B.2: Comandos para a execução de uma simulação

C
Dados da simulação de um radar de busca e vigilância 3D com motor

A seguir serão mostrados dados coletados a partir das simulações de testes. Os dados das trajetórias observadas são dados coletados diretamente dos componentes vetores e são dados de referência para os cálculos estatísticos. Os dados das trajetórias calculadas são medidas calculadas pelo processamento dos sinais simulados.

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	0.18	26984.62	0.209439	0.680625	-75.00
2	4.19	26649.44	0.209419	0.679445	-75.00
3	8.17	26315.66	0.209411	0.678241	-75.00
4	12.18	25980.36	0.209381	0.676998	-75.00
5	16.19	25645.11	0.209359	0.675723	-75.00
6	20.08	25319.47	0.209339	0.674453	-75.00
7	24.08	24984.55	0.209394	0.673102	-75.00
8	28.08	24650.98	0.209479	0.671717	-75.00
9	32.07	24317.49	0.209566	0.670295	-75.00
10	36.06	23984.05	0.209655	0.668832	-75.00
11	40.05	23650.66	0.209747	0.667329	-75.00

Tabela C.1: Trajetória observada do avião 01 da simulação de um radar de busca e vigilância 3D com motor

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	0.13	26976	0.210414	0.681814	-86.25
2	4.14	26640	6.487235	0.675679	-86.61
3	8.13	26304	12.763816	0.668981	-86.61
4	12.13	25968	19.052084	0.685928	-86.61
5	16.14	25632	25.328663	0.674831	-86.61
6	20.13	25344	31.593557	0.659284	-86.61
7	24.13	25008	37.897182	0.663672	-86.22
8	28.13	24672	44.178432	0.671411	-86.61
9	32.12	24336	50.459682	0.664341	-86.61
10	36.11	24000	56.738586	0.674811	-86.61
11	40.10	23664	63.033901	0.670407	-86.61

Tabela C.2: Trajetória calculada do avião 01 da simulação de um radar de busca e vigilância 3D com motor

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	0.13	-8.62	0.000975	0.001189	-11.25
2	4.14	-9.44	-0.005371	-0.003766	-11.61
3	8.13	-11.66	-0.011955	-0.009259	-11.61
4	12.13	-12.36	-0.006852	0.008931	-11.61
5	16.14	-13.11	-0.013438	-0.000893	-11.61
6	20.13	24.53	-0.031709	-0.015168	-11.61
7	24.13	23.45	-0.011325	-0.009431	-11.22
8	28.13	21.03	-0.013344	-0.000307	-11.61
9	32.12	18.51	-0.015365	-0.005955	-11.61
10	36.11	15.95	-0.019735	0.005967	-11.61
11	40.10	13.34	-0.007696	0.003078	-11.61
	Média	5.60	-0.012347	-0.002329	-11.54
	Desvio padrão	16.27	0.008485	0.007224	0.15

Tabela C.3: Erro entre a trajetória calculada e a observada do avião 01 da simulação de um radar de busca e vigilância 3D com motor

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	0.45	31655.20	0.628337	0.593291	-47.00
2	4.37	31440.09	0.628496	0.592225	-47.00
3	8.34	31221.88	0.628661	0.591129	-47.00
4	12.35	31001.97	0.628827	0.590009	-47.00
5	16.46	30776.61	0.629001	0.588844	-47.00
6	20.36	30562.24	0.629168	0.587721	-47.00
7	24.45	30337.81	0.629345	0.586526	-47.00
8	28.34	30124.34	0.629516	0.585373	-47.00
9	32.36	29904.10	0.629695	0.584167	-47.00
10	36.43	29679.77	0.629878	0.582856	-47.00
11	40.32	29461.94	0.630056	0.581419	-47.00

Tabela C.4: Trajetória observada do avião 02 da simulação de um radar de busca e vigilância 3D com motor

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	0.40	31632	0.631236	0.597508	-55.11
2	4.41	31440	6.908011	0.603827	-55.11
3	8.39	31248	13.184592	0.587329	-55.11
4	12.40	31008	19.472861	0.578731	-55.11
5	16.41	30768	25.761127	0.600824	-55.11
6	20.41	30576	32.037846	0.583191	-55.11
7	24.40	30336	38.330776	0.584456	-55.11
8	28.39	30144	44.600307	0.592137	-55.11
9	32.37	29904	50.869839	0.587301	-58.18
10	36.38	29664	57.174526	0.585202	-55.11
11	40.37	29472	63.441711	0.577372	-55.11

Tabela C.5: Trajetória calculada do avião 02 da simulação de um radar de busca e vigilância 3D com motor

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	0.40	-23.20	0.002911	0.004218	-8.11
2	4.41	-0.09	-0.003671	0.011602	-8.11
3	8.39	26.13	-0.010439	-0.003811	-8.11
4	12.40	6.04	-0.005523	-0.011278	-8.11
5	16.41	-8.61	-0.000615	0.011981	-8.11
6	20.41	13.76	-0.007251	-0.004529	-8.11
7	24.40	-1.81	0.002319	-0.002071	-8.11
8	28.39	19.66	-0.011505	0.006763	-8.11
9	32.37	-0.10	-0.025336	0.003134	-11.18
10	36.38	-15.77	-0.004016	0.002346	-8.11
11	40.37	10.06	-0.020193	-0.004047	-8.11
	Erro médio	2.37	-0.007575	0.001302	-8.39
	Desvio padrão	14.83	0.008869	0.007219	0.92

Tabela C.6: Erro entre a trajetória calculada e a observada do avião 02 da simulação de um radar de busca e vigilância 3D com motor

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	0.59	17040.14	1.029746	0.628675	60.00
2	4.69	17319.24	1.029755	0.631105	60.00
3	8.60	17584.83	1.029763	0.633346	60.00
4	12.60	17857.50	1.029771	0.635576	60.00
5	16.72	18138.05	1.029781	0.637811	60.00
6	20.60	18402.08	1.029788	0.639831	60.00
7	24.61	18675.00	1.029795	0.641869	60.00
8	28.70	18953.76	1.029803	0.643889	60.00
9	32.57	19218.02	1.029811	0.645751	60.00
10	36.59	19492.16	1.029817	0.647627	60.00
11	40.67	19770.12	1.029824	0.649476	60.00

Tabela C.7: Trajetória observada do avião 03 da simulação de um radar de busca e vigilância 3D com motor

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	0.64	17040.00	1.005293	0.625998	70.86
2	4.64	17328.00	7.293724	0.624638	70.86
3	8.65	17568.00	13.584116	0.633292	70.86
4	12.65	17856.00	19.870258	0.630433	70.86
5	16.67	18144.00	26.170218	0.649482	70.53
6	20.65	18384.00	32.424526	0.641262	70.50
7	24.66	18672.00	38.729214	0.646157	70.86
8	28.65	18960.00	44.998745	0.645697	70.86
9	32.62	19200.00	51.268276	0.642659	70.86
10	36.64	19488.00	57.584675	0.651612	70.86
11	40.62	19776.00	63.842487	0.660605	70.86

Tabela C.8: Trajetória calculada do avião 03 da simulação de um radar de busca e vigilância 3D com motor

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	0.64	-0.14	-0.024452	-0.002677	10.86
2	4.64	8.76	-0.019216	-0.006467	10.86
3	8.65	-16.83	-0.012018	0.000001	10.86
4	12.65	-1.50	-0.009069	-0.005143	10.86
5	16.67	5.95	0.007696	0.011681	10.53
6	20.65	-18.08	-0.021189	0.001431	10.50
7	24.66	-3.00	0.000307	0.004288	10.86
8	28.65	6.24	-0.013354	0.001807	10.86
9	32.62	-18.02	-0.027014	-0.003091	10.86
10	36.64	-4.16	0.006194	0.003985	10.86
11	40.62	5.88	-0.019185	0.011130	10.86
	Erro médio	-3.17	-0.011936	0.001535	10.80
	Desvio padrão	10.19	0.012039	0.005991	0.14

Tabela C.9: Erro entre a trajetória calculada e a observada do avião 03 da simulação de um radar de busca e vigilância 3D com motor

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	1.03	25053.77	1.535890	0.559000	45.00
2	4.92	25256.46	1.535891	0.560846	45.00
3	9.02	25470.34	1.535893	0.562761	45.00
4	13.03	25679.29	1.535894	0.564600	45.00
5	16.94	25883.13	1.535896	0.566365	45.00
6	21.04	26097.44	1.535897	0.568190	45.00
7	25.00	26304.18	1.535898	0.569921	45.00
8	28.91	26508.40	1.535900	0.571605	45.00
9	33.00	26722.18	1.535901	0.573339	45.00
10	36.94	26928.04	1.535902	0.574982	45.00
11	40.88	27134.35	1.535903	0.576603	45.00

Tabela C.10: Trajetória observada do avião 04 da simulação de um radar de busca e vigilância 3D com motor

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	0.98	25056.00	1.529009	0.565114	55.11
2	4.97	25248.00	7.796317	0.560869	55.11
3	8.97	25488.00	14.096274	0.573148	55.11
4	12.98	25680.00	20.384542	0.563214	55.11
5	16.99	25872.00	26.672810	0.555560	55.11
6	20.99	26112.00	32.951870	0.590069	55.11
7	24.97	26304.00	39.221401	0.584842	55.11
8	28.96	26496.00	45.502651	0.579468	54.68
9	32.95	26736.00	51.783901	0.581105	55.11
10	36.96	26928.00	58.076870	0.578362	54.62
11	40.93	27120.00	64.346397	0.577780	55.11

Tabela C.11: Trajetória calculada do avião 04 da simulação de um radar de busca e vigilância 3D com motor

	Tempo(seg)	Distância(m)	Azimute(rad)	Elevação(rad)	Velocidade(m/s)
1	0.98	2.23	-0.006881	0.006114	10.11
2	4.97	-8.46	-0.022760	0,0000001	10.11
3	8.97	17.66	-0.005989	0.010387	10.11
4	12.98	0.71	-0.000908	-0.001386	10.11
5	16.99	-11.13	0.004172	-0.010804	10.11
6	20.99	14.56	0,000001	0.021880	10.11
7	24.97	-0.18	-0.013609	0.014920	10.11
8	28.96	-12.40	-0.015544	0.007863	9.68
9	32.95	13.82	-0.017479	0.007766	10.11
10	36.96	-0.04	-0.007696	0.003380	9.62
11	40.93	-14.35	-0.021354	0.001176	10.11
	Erro médio	0.22	-0.009818	0.005574	10.03
	Desvio padrão	11.29	0.008981	0.008709	0.19

Tabela C.12: Erro entre a trajetória calculada e a observada do avião 04 da simulação de um radar de busca e vigilância 3D com motor

D
Dados da simulação de um radar com matriz de antenas

A seguir serão mostrados dados coletados a partir da simulação do primeiro teste. Os dados das trajetórias observadas são dados coletados diretamente dos componentes vetores e são dados de referência para os cálculos estatísticos. Os dados das trajetórias calculadas são medidas calculadas pelo processamento dos sinais simulados.

	Tempo(seg)	Distância(m)	Azimute(rad)	Velocidade(m/s)
1	0.193488	22982	0.292242	-94.482422
2	4.182475	22601	6.557365	-94.482422
3	8.184663	22230	12.845632	-94.482422
4	12.190766	21840	19.14559	-94.482422
5	16.196869	21452	25.433861	-94.482422
6	20.202972	21067	31.710442	-94.482422
7	24.194183	20686	37.990925	-94.482422
8	28.185394	20354	44.283894	-94.482422
9	32.176605	19968	50.565144	-94.482422
10	36.167816	19579	56.84639	-94.482422
11	40.159027	19240	63.115925	-94.482422
12	44.165131	18864	69.420616	-94.044998
13	48.141449	18529	75.679893	-87.100983
14	52.147552	18138	81.983124	-86.608887
15	56.138763	17818	88.264374	-86.608887
16	60.129974	17427	94.55735	-86.608887
17	64.121185	17080	100.8386	-86.608887
18	68.112396	16754	107.108124	-86.608887
19	72.1185	16375	113.412819	-86.608887
20	76.094818	16082	119.68235	-86.608887
21	80.086029	15742	125.963593	-86.608887
22	84.07724	15400	132.233124	-78.735352

Tabela D.1: Trajetória calculada do avião 01 da simulação de um radar com matriz de antenas

	Tempo(seg)	Distância(m)	Azimute(rad)	Velocidade(m/s)
1	0.083256	22991.978516	0.31416	-84.86
2	4.076279	22607.980469	0.314177	-84.53
3	8.084651	22224.015625	0.314196	-83.83
4	12.110232	21840.007812	0.314214	-83.69
5	16.153954	21455.976562	0.314234	-83.41
6	20.21628	21071.992188	0.314254	-82.89
7	24.29907	20688.009766	0.314275	-82.78
8	27.889769	20351.998047	0.314294	-82.62
9	32.01535	19967.994141	0.314317	-82.43
10	36.166046	19584.003906	0.31434	-82.15
11	39.820465	19247.988281	0.314361	-81.78
12	44.024185	18863.992188	0.314386	-81.66
13	47.728374	18528.017921	0.314409	-81.48
14	51.996746	18143.994141	0.314436	-81.29
15	55.764187	17807.986328	0.314461	-80.95
16	60.106979	17424.011719	0.314491	-80.46
17	63.943256	17087.986328	0.314518	-80.23
18	67.815819	16751.980469	0.314547	-79.97
19	72.274887	16369.327148	0.314581	-79.82
20	75.684189	16080.012695	0.314608	-79.45
21	79.688835	15744.005859	0.31464	-79.21
22	83.746979	15407.998047	0.314675	-78.89

Tabela D.2: Trajetória observada do avião 01 da simulação de um radar com matriz de antenas

	Tempo(seg)	Distância(m)	Azimute(rad)	Velocidade(m/s)
1	0.193488	-9.978516	-0.021918	-15.592422
2	4.182475	-6.980469	-0.0399973	-15.272422
3	8.184663	5.984375	-0.0349346	-15.032422
4	12.190766	-0.007812	-0.0181799	-14.662422
5	16.196869	-3.976562	-0.0131142	-14.512422
6	20.202972	-4.992188	-0.0197385	-14.252422
7	24.194183	-2.009766	-0.0224618	-14.022422
8	28.185394	2.001953	-0.0126971	-13.532422
9	32.176605	0.005859	-0.0146554	-13.192422
10	36.167816	-5.003906	-0.0166177	-13.002422
11	40.159027	-7.988281	-0.030289	-12.822422
12	44.165131	0.007812	-0.0088083	-12.264998
13	48.141449	0.982079	-0.0327396	-4.950983
14	52.147552	-5.994141	-0.0127209	-4.178887
15	56.138763	10.013672	-0.0146812	-3.988887
16	60.129974	2.988281	-0.0049205	-3.828887
17	64.121185	-7.986328	-0.0068828	-3.718887
18	68.112396	2.019531	-0.0205731	-3.198887
19	72.1185	5.672852	0.0009026	-2.918887
20	76.094818	1.987305	-0.0127787	-2.778887
21	80.086029	-2.005859	-0.014753	-2.078887
22	84.07724	-7.998047	-0.0284423	6.124648
	Média	-1.0068355	-0.01568535	-12.54371
	Desvio padrão	5.2984600442	0.0101653336	6.2426656488

Tabela D.3: Erro entre a trajetória calculada e a observada do avião 01 da simulação de um radar com matriz de antenas

	Tempo(seg)	Distância(m)	Azimute(rad)	Velocidade(m/s)
1	1.101384	45078	1.718389	62.988281
2	5.089991	45307	7.983327	62.988281
3	9.09311	45556	14.271598	62.988281
4	13.099214	45842	20.55987	62.988281
5	17.105316	46081	26.848135	62.988281
6	21.11142	46325	33.139362	62.988281
7	25.072845	46560	39.373734	62.988281
8	29.078949	46850	45.678421	62.988281
9	33.085052	47088	51.983112	62.988281
10	37.061371	47322	58.240932	62.988281
11	41.067474	47610	64.545624	62.988281
12	45.043793	47859	70.803436	62.988281
13	49.049896	48092	77.108131	62.988281
14	53.026215	48389	83.365944	62.988281
15	57.032318	48626	89.670624	62.988281
16	61.038422	48863	95.975319	62.988281
17	65.01474	49101	102.233131	62.988281
18	69.020844	49397	108.537819	62.988281
19	72.997162	49634	114.795631	62.988281
20	76.988373	49916	121.076881	62.988281
21	80.979584	50161	127.358131	62.988281
22	84.985687	50401	133.662827	62.988281

Tabela D.4: Trajetória calculada do avião 02 da simulação de um radar com matriz de antenas

	Tempo(seg)	Distância(m)	Azimute(rad)	Velocidade(m/s)
1	1.142791	45072.28125	1.500993	49.81
2	4.932093	45312.09375	1.501026	49.84
3	8.72	45552.03125	1.501059	49.93
4	13.263256	45840.09375	1.501098	49.96
5	17.044186	46080.046875	1.501131	49.97
6	20.823256	46320.085938	1.501163	49.99
7	24.598606	46560.085938	1.501194	50.07
8	29.125582	46848.121094	1.501232	50.09
9	32.893955	47088.105469	1.501263	50.11
10	36.658604	47328.035156	1.501293	50.13
11	41.174885	47616.113281	1.50133	50.16
12	44.936279	47856.242188	1.50136	50.18
13	48.690701	48096.101562	1.501389	50.19
14	53.17907	48383.082031	1.501425	50.23
15	56.944653	48624.039062	1.501454	50.27
16	60.697678	48864.359375	1.501483	50.31
17	64.440468	49104.191406	1.501511	50.36
18	68.929306	49392.046875	1.501545	50.42
19	72.669769	49632.089844	1.501573	50.45
20	77.126976	49918.332031	1.501607	50.48
21	80.887909	50160.03125	1.501634	50.54
22	84.620934	50400.09375	1.501662	50.57

Tabela D.5: Trajetória observada do avião 02 da simulação de um radar com matriz de antenas

	Tempo(seg)	Distância(m)	Azimute(rad)	Velocidade(m/s)
1	1.101384	5.71875	0.217396	13.178281
2	5.089991	-5.09375	0.1991157	13.148281
3	9.09311	3.96875	0.2041684	13.058281
4	13.099214	1.90625	0.2092161	13.028281
5	17.105316	0.953125	0.2142628	13.018281
6	21.11142	4.914062	0.2222725	12.998281
7	25.072845	-0.085938	0.1734282	12.918281
8	29.078949	1.878906	0.1948919	12.898281
9	33.085052	-0.105469	0.2163666	12.878281
10	37.061371	-6.035156	0.1909713	12.858281
11	41.067474	-6.113281	0.212441	12.828281
12	45.043793	2.757812	0.1870377	12.808281
13	49.049896	-4.101562	0.2085184	12.798281
14	53.026215	5.917969	0.1831101	12.758281
15	57.032318	1.960938	0.2045758	12.718281
16	61.038422	-1.359375	0.2260565	12.678281
17	65.01474	-3.191406	0.2006552	12.628281
18	69.020844	4.953125	0.2221239	12.568281
19	72.997162	1.910156	0.1967226	12.538281
20	76.988373	-2.332031	0.1947533	12.508281
21	80.979584	0.96875	0.192791	12.448281
22	84.985687	0.90625	0.2142737	12.418281
	Média	0.9609375	0.2043721	12.818281
	Desvio padrão	3.6924994375	0.0138384696	0.2218263429

Tabela D.6: Erro entre a trajetória calculada e a observada do avião 02 da simulação de um radar com matriz de antenas

E
Dados reais provenientes do Radar Saber M60

A seguir serão apresentados dados de trajetórias reais coletados do radar M60 e comparados com dados de GPS das aeronaves, onde os valores -999 foram medidas descartadas e não entraram no cálculo do erro e do desvio padrão.

		Distância(m) (m)		
1	Tempo(seg)	Radar	GPS	Diferença
2	2499.41	20970	20864	-92
3	2502.98	21136	21006	-114
4	2506.96	21218	21124	-79
5	2514.96	21150	21129	-6
6	2518.89	20978	20982	-12
7	2522.92	20721	20745	9
8	2526.84	20396	20451	40
9	2534.85	19697	19680	-2
10	2574.62	14121	14243	108
11	2582.58	12893	13014	106
12	2590.53	11606	11726	105
13	2610.42	8248	8333	70
14	2622.4	6347	6335	3
15	2626.38	5768	5700	-53
16	2630.36	5197	5325	113
17	2634.34	4661	4750	74
18	2650.09	2748	2771	8
19	2653.97	2441	2462	6
20	2657.79	2253	2158	-80
21	3698.19	4098	4200	87
22	3898.28	7521	7638	102
23	3902.26	8064	7935	-114
24	4566.23	13231	13294	48
25	4570.26	12655	12645	5
26	4574.23	12024	12006	-3
27	4578.16	11444	11433	4
28	4642.16	2322	2392	55
29	5398.06	7557	7611	39
30	5422.08	4203	4329	111
31	5606.11	7921	8007	71
32	6230.05	11322	11261	-47
33	7154.29	4810	4718	-76
34	8089.9	6035	5946	-74
35	8145.89	12661	12535	-112
	Média			24
	Desvio padrão			77

Tabela E.1: Dados de distância do Radar Saber M60

		Azimute(rad) (graus)		
1	Tempo(seg)	Radar	GPS	Diferença
2	2499.41	262.002	262.246	0.245
3	2502.98	260.593	260.501	-0.093
4	2506.96	258.709	258.532	-0.177
5	2514.96	254.180	254.481	0.301
6	2518.89	252.256	252.461	0.205
7	2522.92	250.464	250.402	-0.062
8	2526.84	248.046	248.429	0.382
9	2534.85	243.918	244.506	0.588
10	2574.62	226.884	226.761	-0.123
11	2582.58	224.018	223.538	-0.480
12	2590.53	220.758	220.641	-0.117
13	2610.42	213.826	213.627	-0.200
14	2622.4	-9999	203.429	-9999
15	2626.38	-9999	200.866	-9999
16	2630.36	205.300	206.419	1.120
17	2634.34	203.203	203.845	0.643
18	2650.09	180.215	180.908	0.693
19	2653.97	168.352	167.238	-1.114
20	2657.79	152.905	153.779	0.874
21	3698.19	200.022	201.033	1.012
22	3898.28	208.623	209.490	0.867
23	3902.26	208.370	208.391	0.022
24	4566.23	207.260	207.225	-0.035
25	4570.26	207.482	209.628	2.146
26	4574.23	-9999	211.995	-9999
27	4578.16	-9999	211.844	-9999
28	4642.16	-9999	191.442	-9999
29	5398.06	206.509	208.017	1.507
30	5422.08	206.171	205.659	-0.511
31	5606.11	209.295	208.805	-0.490
32	6230.05	207.864	206.234	-1.630
33	7154.29	-9999	240.404	-9999
34	8089.9	209.255	208.047	-1.208
35	8145.89	-9999	210.669	-9999
	Média			0.113
	Desvio padrão			0.729

Tabela E.2: Dados de azimute do Radar Saber M60

		Elevação (graus)		
1	Tempo(seg)	Radar	GPS	Diferença
2	2499.41	0.432	2.055	1.623
3	2502.98	-0.316	2.183	2.499
4	2506.96	0.435	2.298	1.863
5	2514.96	1.498	2.447	0.949
6	2518.89	2.067	2.428	0.362
7	2522.92	2.123	2.409	0.286
8	2526.84	1.856	2.414	0.558
9	2534.85	1.738	2.479	0.741
10	2574.62	3.485	3.401	-0.084
11	2582.58	2.822	3.687	0.865
12	2590.53	2.354	4.012	1.658
13	2610.42	4.947	5.615	0.668
14	2622.4	5.032	7.396	2.364
15	2626.38	6.674	8.260	1.586
16	2630.36	7.636	9.986	2.351
17	2634.34	10.505	11.358	0.854
18	2650.09	17.974	20.358	2.384
19	2653.97	-9999	23.701	-9999
20	2657.79	-9999	26.991	-9999
21	3698.19	12.257	12.918	0.661
22	3898.28	6.078	5.603	-0.475
23	3902.26	5.607	6.122	0.515
24	4566.23	-9999	3.394	-9999
25	4570.26	-9999	2.854	-9999
26	4574.23	4.572	2.323	-2.250
27	4578.16	4.201	2.519	-1.682
28	4642.16	19.672	22.126	2.454
29	5398.06	7.105	8.780	1.675
30	5422.08	-9999	14.817	-9999
31	5606.11	8.219	6.976	-1.243
32	6230.05	4.859	5.862	1.003
33	7154.29	10.821	9.029	-1.792
34	8089.9	6.756	7.482	0.726
35	8145.89	3.641	3.522	-0.120
	Média			0.797
	Desvio padrão			0.927

Tabela E.3: Dados de elevação do Radar Saber M60

F
Diagramas de sequência

A seguir serão apresentados partes dos diagramas de sequência da simulação de um radar 3D com motor e da simulação de um radar de míssil.

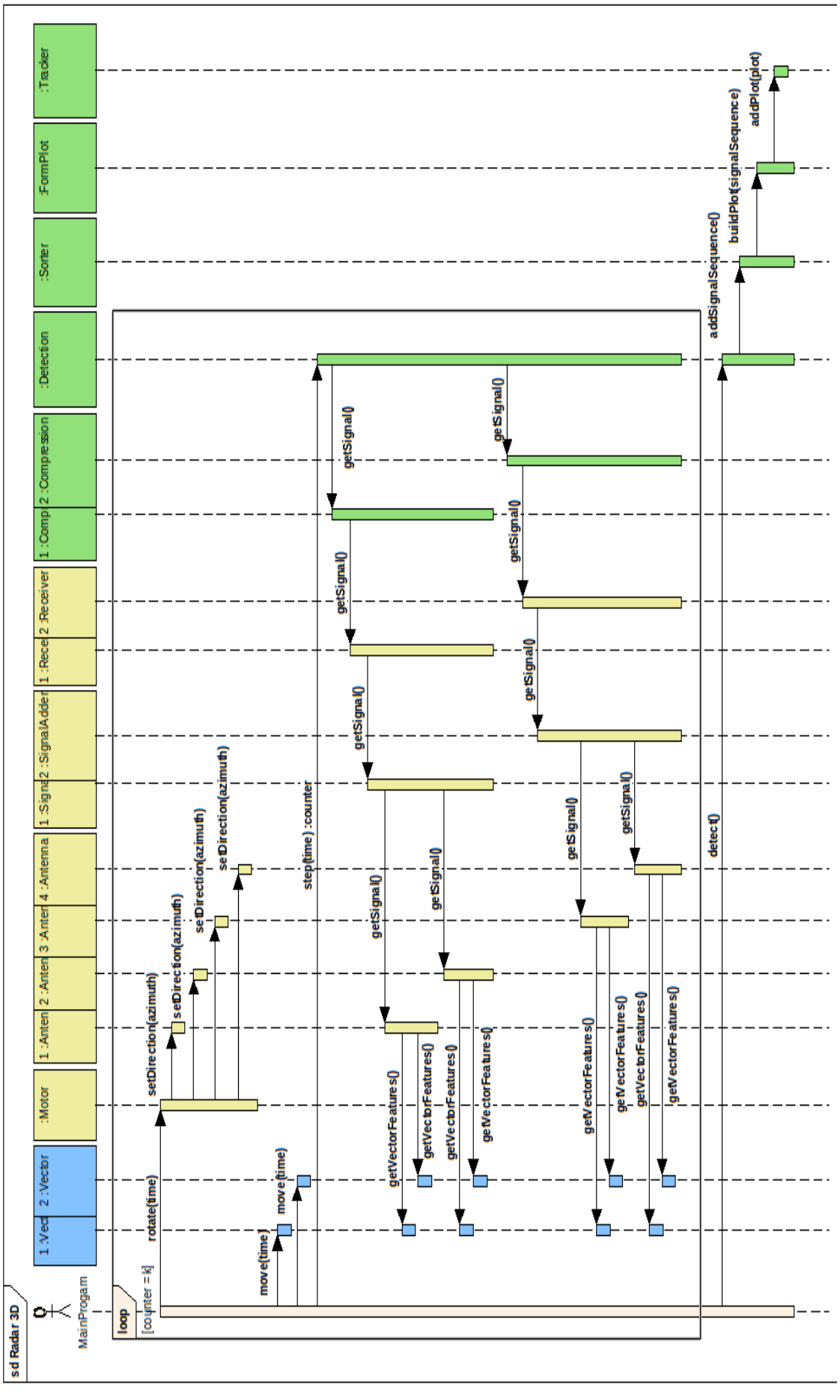

Figura F.1: Parte do diagrama de sequência da simulação do radar 3D. Foram omitidos os componentes da camada de suporte para não comprometer o tamanho da figura

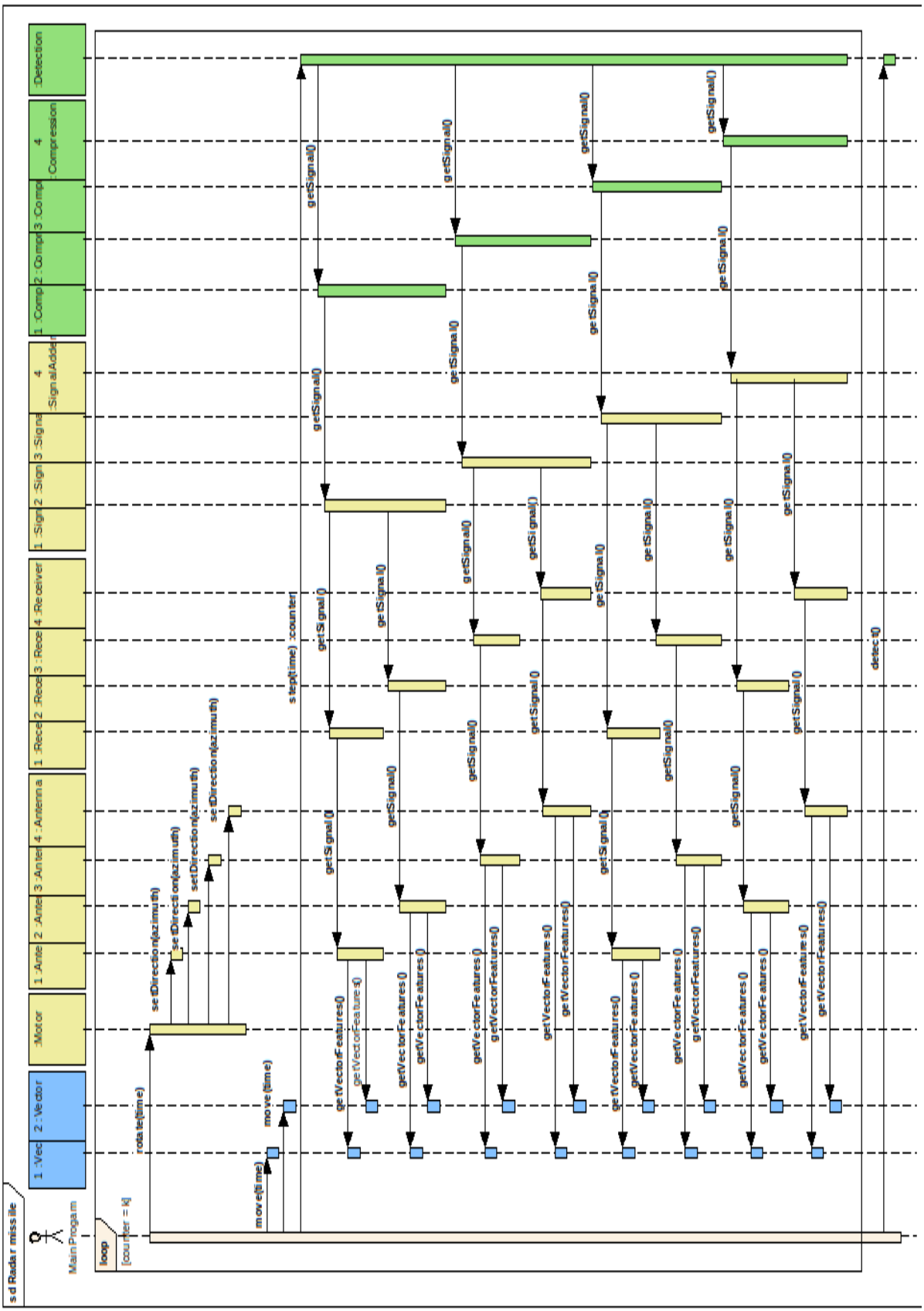

Figura F.2: Parte do diagrama de sequência da simulação do radar de míssil. Foram omitidos os componentes do processamento de dados e da camada de suporte para não comprometer o tamanho da figura