ISSN 0917-057X

March 2014

防災科学技術研究所研究資料　第386号

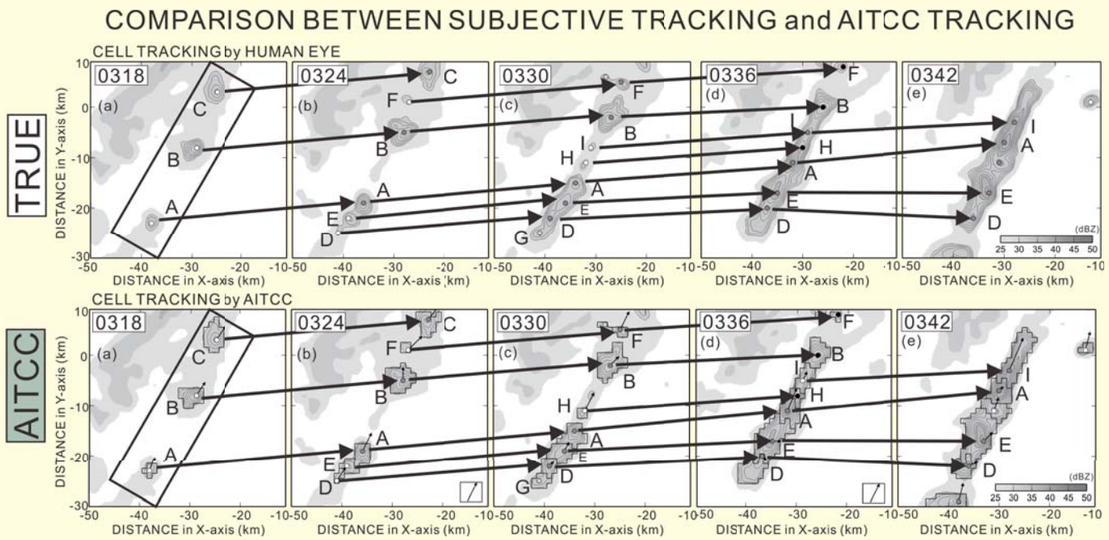
Technical Note of the National Research Institute
for Earth Science and Disaster Prevention: No.386

AITCC ユーザーガイド

—自動対流セル検出・追跡アルゴリズム—

The AITCC User Guide

– An Automatic Algorithm for the Identification and Tracking of Convective Cells –



独立行政法人
防災科学技術研究所
National Research Institute for Earth Science and Disaster Prevention, Japan

防災科学技術研究所研究資料　第三八六号

AITCC ユーザーガイド　―自動対流セル検出・追跡アルゴリズム―

防災科学技術研究所

National Research Institute for Earth Science and Disaster Prevention
Tennodai 3-1, Tsukuba, Ibaraki, 305-0006 Japan

防災科学技術研究所研究資料

第 315 号　地すべり地形分布図 第 35 集「八代」18 葉(5 万分の 1)．2008 年 3 月発行
第 316 号　地すべり地形分布図 第 36 集「熊本」15 葉(5 万分の 1)．2008 年 3 月発行
第 317 号　2004 年新潟県中越地震による斜面変動分布図(付録 CD-ROM)37pp．2008 年 3 月発行
第 318 号　強震ネットワーク　強震データ　Vol. 23(平成 19 年 No. 1) (CD-ROM 版)．2008 年 3 月発行
第 319 号　強震ネットワーク　強震データ　Vol. 24(平成 19 年 No. 2) (CD-ROM 版)．2008 年 3 月発行
第 320 号　平成 17 年度大都市大震災軽減化特別プロジェクトⅡ 木造建物実験 - 震動台活用による構造物の耐震性向上研究 -(付録 CD-ROM)152pp．2008 年 3 月発行
第 321 号　平成 17 年度大都市大震災軽減化特別プロジェクト 実大 6 層 RC 建物実験報告書(付録 CD-ROM)46pp. 2008 年 3 月発行
第 322 号　地すべり地形分布図 第 37 集「福岡・中津」24 葉(5 万分の 1)．2008 年 8 月発行
第 323 号　地すべり地形分布図 第 38 集「長崎・唐津」29 葉(5 万分の 1)．2008 年 9 月発行
第 324 号　地すべり地形分布図 第 39 集「鹿児島」24 葉(5 万分の 1)．2008 年 11 月発行
第 325 号　地すべり地形分布図 第 40 集「一関・石巻」19 葉(5 万分の 1)．2009 年 2 月発行
第 326 号　新庄における気象と降積雪の観測 (2007/08 年冬期)　33pp．2008 年 12 月発行
第 327 号　防災科学技術研究所 45 年のあゆみ(付録 DVD) 224pp．2009 年 3 月発行
第 328 号　地すべり地形分布図 第 41 集「盛岡」18 葉(5 万分の 1)．2009 年 3 月発行
第 329 号　地すべり地形分布図 第 42 集「野辺地・八戸」24 葉(5 万分の 1)．2009 年 3 月発行
第 330 号　地域リスクとローカルガバナンスに関する調査報告　53pp．2009 年 3 月発行
第 331 号　E-Defense を用いた実大 RC 橋脚 (C1-1 橋脚) 震動破壊実験研究報告書 -1970 年代に建設された基部曲げ破壊タイプの RC 橋脚震動台実験 -(付録 DVD)　107pp．2009 年 1 月発行
第 332 号　強震ネットワーク　強震データ　Vol. 25(平成 20 年 No. 1) (CD-ROM 版)．2009 年 3 月発行
第 333 号　強震ネットワーク　強震データ　Vol. 26(平成 20 年 No. 2) (CD-ROM 版)．2009 年 3 月発行
第 334 号　平成 17 年度大都市大震災軽減化特別プロジェクトⅡ 地盤基礎実験 - 震動台活用による構造物の耐震性向上研究 -(付録 CD-ROM)　62pp．2009 年 10 月発行
第 335 号　地すべり地形分布図 第 43 集「函館」14 葉(5 万分の 1)．2009 年 12 月発行
第 336 号　全国地震動予測地図作成手法の検討(7 分冊＋ CD-ROM 版)．2009 年 11 月発行
第 337 号　強震動評価のための全国深部地盤構造モデル作成手法の検討(付録 DVD)．2009 年 12 月発行
第 338 号　地すべり地形分布図 第 44 集「室蘭・久遠」21 葉(5 万分の 1)．2010 年 3 月発行
第 339 号　地すべり地形分布図 第 45 集「岩内」14 葉(5 万分の 1)．2010 年 3 月発行
第 340 号　新庄における気象と降積雪の観測 (2008/09 年冬期) 33pp．2010 年 3 月発行
第 341 号　強震ネットワーク　強震データ　Vol. 27(平成 21 年 No. 1) (CD-ROM 版)．2010 年 3 月発行
第 342 号　強震ネットワーク　強震データ　Vol. 28(平成 21 年 No. 2) (CD-ROM 版)．2010 年 3 月発行
第 343 号　阿寺断層系における深層ボーリング調査の概要と岩石物性試験結果(付録 CD-ROM)　15pp．2010 年 3 月発行
第 344 号　地すべり地形分布図 第 46 集「札幌・苫小牧」19 葉(5 万分の 1)．2010 年 7 月発行
第 345 号　地すべり地形分布図 第 47 集「夕張岳」16 葉(5 万分の 1)．2010 年 8 月発行
第 346 号　長岡における積雪観測資料(31) (2006/07 , 2007/08 , 2008/09 冬期)47pp．2010 年 9 月発行
第 347 号　地すべり地形分布図 第 48 集「羽幌・留萌」17 葉(5 万分の 1)．2010 年 11 月発行
第 348 号　平成 18 年度 大都市大震災軽減化特別プロジェクト実大 3 層 RC 建物実験報告書(付録 DVD) 68pp．2010 年 8 月発行
第 349 号　防災科学技術研究所による深層掘削調査の概要と岩石物性試験結果(足尾・新宮・牛伏寺) (付録 CD-ROM)12pp. 2010 年 8 月発行
第 350 号　アジア防災科学技術情報基盤(DRH-Asia) コンテンツ集　266pp．2010 年 12 月発行
第 351 号　新庄における気象と降積雪の観測(2009/10 年冬期)　31pp．2010 年 12 月発行
第 352 号　平成 18 年度 大都市大震災軽減化特別プロジェクトⅡ 木造建物実験 - 震動台活用による構造物の耐震性向上研究 -(付録 CD-ROM)120pp．2011 年 1 月発行
第 353 号　地形・地盤分類および常時微動の H/V スペクトル比を用いた地震動のスペクトル増幅率の推定　242pp. 2011 年 1 月発行
第 354 号　地震動予測地図作成ツールの開発(付録 DVD) 155pp．2011 年 5 月発行
第 355 号　ARTS により計測した浅間山の火口内温度分布(2007 年 4 月から 2010 年 3 月)　28pp．2011 年 1 月発行
第 356 号　長岡における積雪観測資料(32) (2009/10 冬期)　29pp．2011 年 2 月発行
第 357 号　浅間山鬼押出火山観測井コア試料の岩相と層序(付録 DVD)　32pp．2011 年 2 月発行

表紙図 ‥‥ 高度 3 km におけるレーダ反射強度の 5 分毎の時間変化．人間の目による 対流セルの追跡結果(上段)を矢印で示す．下段に自動対流セル追跡アルゴリズム AITCC(エイティック)による追跡結果を示す．

防災科学技術研究所研究資料

第 358 号　強震ネットワーク　強震データ　Vol. 29(平成 22 年 No. 1) (CD-ROM 版)．2011 年 2 月発行
第 359 号　強震ネットワーク　強震データ　Vol. 30(平成 22 年 No. 2) (CD-ROM 版)．2011 年 2 月発行
第 360 号　K-NET・KiK-net 強震データ(1996 － 2010) (DVD 版 6 枚組)．2011 年 3 月発行
第 361 号　統合化地下構造データベースの構築 ＜地下構造データベース構築ワーキンググループ報告書＞ 平成 23 年 3 月 238pp．2011 年 3 月発行
第 362 号　地すべり地形分布図 第 49 集「旭川」16 葉(5 万分の 1)．2011 年 11 月発行
第 363 号　長岡における積雪観測資料(33) (2010/11 冬期)　29pp．2012 年 2 月発行
第 364 号　新庄における気象と降積雪の観測(2010/11 年冬期)　45pp．2012 年 2 月発行
第 365 号　地すべり地形分布図 第 50 集「名寄」16 葉(5 万分の 1)．2012 年 3 月発行
第 366 号　浅間山高峰火山観測井コア試料の岩相と層序(付録 CD-ROM)　30pp．2012 年 2 月発行
第 367 号　防災科学技術研究所による関東・東海地域における水圧破砕井の孔井検層データ　29pp．2012 年 3 月発行
第 368 号　台風災害被害データの比較について(1951 年～ 2008 年，都道府県別資料) (付録 CD-ROM)19pp．2012 年 5 月発行
第 369 号　E-Defense を用いた実大 RC 橋脚 (C1-5 橋脚) 震動破壊実験研究報告書 - 実在の技術基準で設計した RC 橋脚の耐震性に関する震動台実験及びその解析 - (付録 DVD)　64pp．2012 年 10 月発行
第 370 号　強震動評価のための千葉県・茨城県における浅部・深部地盤統合モデルの検討(付録 CD-ROM)　410pp．2013 年 3 月発行
第 371 号　野島断層における深層掘削調査の概要と岩石物性試験結果 (平林・岩屋・甲山) (付録 CD-ROM)　27pp．2012 年 12 月発行
第 372 号　長岡における積雪観測資料 (34) (2011/12 冬期)　31pp．2012 年 11 月発行
第 373 号　阿蘇山一の宮および白水火山観測井コア試料の岩相記載(付録 CD-ROM)　48pp．2013 年 2 月発行
第 374 号　霧島山万膳および夷守台火山観測井コア試料の岩相記載(付録 CD-ROM)　50pp．2013 年 3 月発行
第 375 号　新庄における気象と降積雪の観測(2011/12 年冬期)　49pp．2013 年 2 月発行
第 376 号　地すべり地形分布図 第 51 集「天塩・枝幸・稚内」20 葉(5 万分の 1)．2013 年 3 月発行
第 377 号　地すべり地形分布図 第 52 集「北見・紋別」25 葉(5 万分の 1)．2013 年 3 月発行
第 378 号　地すべり地形分布図 第 53 集「帯広」16 葉(5 万分の 1)．2013 年 3 月発行
第 379 号　東日本大震災を踏まえた地震ハザード評価の改良に向けた検討　349pp．2012 年 12 月発行
第 380 号　日本の火山ハザードマップ集　第 2 版(付録 DVD)　186pp．2013 年 7 月発行
第 381 号　長岡における積雪観測資料 (35) (2012/13 冬期)　30pp．2013 年 11 月発行
第 382 号　地すべり地形分布図 第 54 集「浦河・広尾」18 葉(5 万分の 1)．2014 年 2 月発行
第 383 号　地すべり地形分布図 第 55 集「斜里・知床岬」23 葉(5 万分の 1)．2014 年 2 月発行
第 384 号　地すべり地形分布図 第 56 集「釧路・根室」16 葉(5 万分の 1)．2014 年 2 月発行
第 385 号　東京都市圏における水害統計データの整備(付録 DVD)　6pp．2014 年 2 月発行

防災科学技術研究所研究資料 第 386 号

平成 26 年 3 月 20 日 発行

編集兼　独立行政法人
発行者　防災科学技術研究所
〒 305-0006
茨城県つくば市天王台 3 － 1
電話 (029)863-7635
http://www.bosai.go.jp/

印刷所　松枝印刷株式会社
茨城県常総市水海道天満町 2438

※防災科学技術研究所の刊行物については，ホームページ（http://dil-opac.bosai.go.jp/publication/）をご覧下さい．

Technical Note of the National Research Institute for Earth Science and Disaster Prevention, No. 386 ; March, 2014

The AITCC User Guide

– An Automatic Algorithm for the Identification and Tracking of Convective Cells –

Shingo SHIMIZU *

*Storm, Flood, and Landslide Research Unit,
National Research Institute for Earth Science and Disaster Prevention, Japan
shimizus@bosai.go.jp

Abstract

The AITCC (algorithm for the identification and tracking of convective cells) is an automatic cell-tracking program that provides statistical analysis of convective cells and short-range nowcasting based on radar observations. The AITCC is freely available from http://mizu.bosai.go.jp/wiki/wiki.cgi?=AITCC_HOME for noncommercial use. This manual describes the basic operation of the AITCC, including how to run the program, the acceptable data formats, and the procedures to handle the outputs of the AITCC.

Key words: Cell tracking, Nowcasting, Statistical analysis of convective cells.

1. Introduction

1.1 What is AITCC?

The development of an automatic cell-tracking algorithm is necessary to improve the short-range forecasting of severe weather using radar datasets, and also to assist with the statistical analysis of severe thunderstorms.

In 2012, Shimizu and Uyeda (referred to as **SU12** hereafter) developed the cell-tracking algorithm AITCC (algorithm for the identification and tracking of convective cells) using the constant and adaptive threshold methods, which is able to identify various kinds of storms ranging from "weak convective cells embedded in widespread precipitation systems", to severe cells such as supercell thunderstorms. Most previous cell-tracking algorithms (e.g., TITAN: Dixon *et al.*, 1990) use a single constant threshold value, which is selected by the user, to identify a cell, and is defined as the region where the reflectivity value is higher than the threshold value. Severe cells would be more easily detected using higher threshold values. However, weak convective cells would not be detected using such high threshold values. In contrast, lower threshold values result in the detection of many severe cells, including convective cells.

AITCC uses the adaptive threshold method combined with the constant threshold method to solve the above problem. The user provides a "minimum threshold value for cell identification", and AITCC automatically calculates the optimal threshold to identify an individual convective cell with a single peak of reflectivity. AITCC provides physical information regarding the cells, such as the "cell dimension", "cell location", "strength of cell", and "lifetime of cell" for statistical analysis. In addition, historical data from the cell tracked by AITCC provide useful information for the short-range nowcasting of severe thunderstorms (Yoshida *et al.*, 2012).

The tracking algorithm in AITCC identifies two cells at successive time steps based on their similar physical properties. For the temporal identification of cells, to improve tracking ability, AITCC imposes three constraints, namely, "area conservation", "conservation of relative location to the other cells' motion", and "inhibiting crossing assignments", when cell merging or splitting occurs. This cell merging and splitting scheme is a great advantage and is a unique feature of AITCC.

From an analysis of more than two thousands of convective cells, **SU12** showed that these three constraints significantly improved cell-tracking ability, even when some weak convective cells were closely colocated and embedded within the stratiform precipitation system. The three constraints increased the critical success index (CSI) from 52.1 % to 71.4 %, and reduced the false alarm rate (FAR) from 46.7 % to 9.6 %, when compared with results obtained without any such constraints.

* 3-1 Tennodai, Tsukuba, Ibaraki, 305-0006, Japan

Table 1 List of directories (DIRs in italics) and executable files (in italics) in AITCC_OPEN.

DIR	Purpose	Executable files
Fix1	Detection algorithm of CG/MCS and CC	*DETECT_MCS_CELL*
Fix2	Linkage algorithm of CG/MCS at two successive times	*LINK_MCS*
Fix3	Jointing of individual CG/MCS in the observation period.	*TRACE_MCS*
Fix4	Linkage algorithm of CC at two successive times	*LINK_CELL*
Fix5	Jointing of individual CC in the observation period.	*TRACE_CELL*
Perl	Perl scripts for running AITCC	Perl scripts
output	Output directory for data damping	nothing
GMT	Drawing AITCC results using GMT tools	many perl scripts

Table 2 List of parameters for input data in chaser.h.

Parameter	Description	Default setting
X_NUM	Dimension of input data in the x-direction	
Y_NUM	Dimension of input data in the y-direction	
ZA_NUM	Dimension of input data in the z-direction	
Z_NUM	Dimension in the z-direction for analysis	1
DX	Grid resolution in the x-direction	1.0 (km)
DY	Grid resolution in the y-direction	1.0 (km)
DZ	Grid resolution in the z-direction	0.5 (km)
ERROR_UNDEF	Missing value for input data	−999.9
ERROR_VALUE	Slightly larger than ERROR_UNDEF	−900.0
W_OPT	Analysis of vertical velocity	1 (use) 0 (no use)

Table 3 List of parameters for options in stratiform-region elimination.

Option	Description	Default setting
BBF	Elimination of stratiform region	see *DETECT.pl*
BBF_DX	Range of BBF analysis column	3.0 (km)
BBF_DY	Range of BBF analysis column	3.0 (km)
BBF_DZ	Depth of BBF analysis column	2.0 (km)
MELT_H	Height of melting layer	5.0 (km) assuming summer season
BBF_THRESH	Threshold value for BBF method	0.3 (see Chen and Uyeda, 2003)
STN	Elimination of stratiform region	see *DETECT.pl*

Table 4 List of parameters used to define tracking targets.

Option	Description	Default setting
CUT_REF	Minimum reflectivity for analysis	10.0 (dBZ)
TRE_MCS	Constant threshold to identify CG or MCS	30 (dBZ)
ECHOTOP_DBZ	Reflectivity value to measure echo-top height	15 (dBZ)
MIN_AREA	Minimum area of CG	10 (km^2)
MIN_AREA_C	Minimum area of CC	10 (km^2)
MAX_AREA_C	Maximum area of CC	900 (km^2)
MIN_D_REF	Minimum reflectivity difference between at peak and edge.	3 (dB)
MIN_DIS_CORE	Minimum distance of neighboring cell's peak	2 (km)
LIM_ASPCT	Maximum aspect ratio of cell's outline	3
MERGE_DEL_CELL	Redistribution option for deleted cells	OFF

AITCC runs that used a higher temporal resolution (1 min) of observations showed improved skill scores when compared with those using a lower temporal resolution (10 min). The FAR was reduced to 0.7 % using a temporal resolution of 1 min. AITCC is expected to be a more powerful tool because next-generation phased array radars with higher temporal resolutions have now been developed (Heinselman and Torres, 2011).

AITCC is written in C and contains 30,000 lines of code. The algorithm can be divided into three parts: 1) cell detection (see Section 2.3 of this manual), 2) cell identification between two successive time steps (see Section 2.4 of this manual), and 3) temporal tracing of an individual cell (Section 2.5 of this manual). Sample Perl scripts that can be used to view the results generated by AITCC are introduced in Section 2.6.

1.2 Availability of AITCC

The complete AITCC software package can be obtained from: http://mizu.bosai.go.jp/wiki/wiki.cgi?=AITCC_HOME and is freely available for noncommercial use. It is requested that anyone using the software should acknowledge the work of the authors by citing this user guide as the official reference for AITCC. The AITCC software is copyrighted, but it is not trademarked; however, if modifications are made that affect the interface, functionality, or accuracy of the resulting software, the name of the routine should be changed. If modifications are made to the software, it is the responsibility of the individuals or company who modified the routine to support the routine. Users who agree to comply with the conditions specified on the web site can download the files "AITCC_OPEN.tar.gz", and "dual_data.tar.gz" as sample datasets to demonstrate the functionality of the algorithm.

1.3 Installation of AITCC

AITCC (version 1.0) is coded in C and assumes that it will run under a Linux/Unix environment. It uses a NetCDF library (version 3.6.2 or later) for managing output. Users should install the NetCDF library in advance using the same compiler as used by AITCC. After unzipping the downloaded files, users will find eight directories under AITCC_OPEN (see **Table 1**. In this manual, directory name, program file name, and executable file name are shown in italics). When all of the following compiling procedures have been completed, users will find the executable programs listed in **Table 1**. The procedure to follow (with example commands in italics) is outlined below.

1. Decompress the download files.
   tar –zxvf AITCC_OPEN.tar.gz
   cd AITCC_OPEN
   mv dual_data.tar.gz ./AITCC_OPEN
   tar –zxvf dual_data.tar.gz
2. Edit the file named "Makefile" in the Fix1 directory to set PATH for the NetCDF library in your own environment.
   set NETCDF /usr/local/netcdf-3.6.2 (example).
3. Execute the "make" command to compile in Fix1.
   cd AITCC_OPEN/Fix1
   make
4. Repeat this procedure for the other directories from *Fix2* to *Fix5*.

1.4 Script files and executable files for running AITCC procedures

Table 1 lists the directories under *AITCC_OPEN*. The detection algorithms for a cell group (CG) or a mesoscale convective system (MCS), and a convective cell (CC) are contained in the directory *Fix1*. AITCC defines a CC as a meso-γ-scale (2–20 km) contiguous region around a single local maximum in the reflectivity field for a given height. A CG or MCS is defined as a contiguous region with reflectivity greater than a given threshold value (default: 30 dBZ). If the horizontal scale of the contiguous region is larger than meso-β-scale (20–200 km), it should be defined as an "MCS", but if not, the target should be a CG. AITCC does not distinguish between an MCS or a CG. In this manual, the expression "CG/MCS" or just "CG" is used for such contiguous regions. The detection algorithm for a CG/MCS is applied first. If a CG/MCS includes several peaks in reflectivity, the CG/MCS will be divided into CCs that have a single reflectivity peak. All CGs (or MCSs), and all CCs, are detected for each time step. Details of the detection procedure are described in Section 2.3 of **SU12**, and Fig. 2 and Table 2 of **SU12**. The detection procedure is executed by DETECT.pl in the Perl directory.

The second procedure links the CGs (identified as the same CGs) between two successive time steps. The linkage algorithm is contained in the directory *Fix2*. The judgment for the identification of two CGs is conducted following the flowchart in Table 3 of **SU12**. The second procedure is executed by *LINK_MCS.pl* in the *Perl* directory.

The third procedure creates the joint linkage between the CGs and generates the individual history for each CG. This joint algorithm is contained in the directory *Fix3* and is based on recursive processing. In this document, the terms *parent* and *child* are used to provide a short, intuitive description of the temporal order as proposed in **SU12**. For instance, in the case of the history of a single isolated CG, a single parent-CG is assigned to a single child-CG for all time steps. Finally, the jointing procedure will make a single history file (see Fig. 6 in **SU12**). In the case of merging two CGs, two parent-CGs are assigned to the same child-CG at a given time step. In this case, the jointing procedure generates two

history files, but one of them ends at the time step. In the case of multiple assignments, AITCC determines the priority order for the assignments using Eq. 4 in **SU12**. The highest priority assignment is allowed to retain its history, while the history of the other (lower priority) assignments is deleted. In the case of splitting, a similar procedure is conducted. Users can determine whether or not the CG that splits is treated as a new CG (default: not allowed; the splitting/merging CGs or CCs are excluded from analysis). This third procedure is executed by *TRACE_MCS.pl* in the Perl directory.

The fourth procedure is similar to the second procedure but for CCs, and this CC linkage algorithm is contained in the directory *Fix4*. The details of this cell-tracking procedure are described in Section 2.4 and Table 4 of **SU12**. This fourth procedure is executed by *LINK_CELL.pl* in the *Perl* directory.

The final procedure is similar to the third procedure but for CCs, and this CC joint algorithm is contained in the directory *Fix5*. This procedure is executed by *TRACE_CELL.pl* in the *Perl* directory. Users can display the results using GMT tools (see Section 2.6). The details of the perl script are provided in Section 2.6.

Table 5 List of subroutines (in italics) in the preprocessing procedure.

Subroutine name	Description
option.c	Reading configure file
secure_mem.c	Securing memory
initial.c	Initializing memory
read_cpi.c	Reading observation file (CAPPI*)
BBF.c	[**Optional**] Eliminating Stratiform using BBF method
STEINER.c	[**Optional**] Eliminating Stratiform using Steiner's method
Select_height.c	Selecting analysis height and quantizing reflectivity data
W_peak.c	[**Optional**] Searching local peak of vertical velocity

*CAPPI indicates 3D reflectivity data in Cartesian coordinate system.

Table 6 List of subroutines for CG/MCS detection.

Subroutine name	Description
edge_MCS.c	Classifying grids (see **Fig. 2**)
name_MCS.c	Numbering ID (see **Fig. 3**)
count_MCS.c	Counting the number of CG/MCS
area_MCS.c	Counting the area
delete_MCS.c	Deleting CG/MCS following deleting option shown in Section 2.2
arrange_MCS.c	Renaming ID after deleting
height_MCS.c	Measure of echo top of CG/MCS
center_MCS.c	Calculation of reflectivity-weighted center of CG/MCS
peak_MCS.c	Counting the number of CC in a CG
peak_value_MCS.c	Getting the maximum reflectivity value in a CG
Calc_Ellipse_MCS.c	Calculation of ellipse parameter to approximate the shape of CG/MCS

2. User configuration

2.1 Definition of input data

Users must edit the file *chaser.h* in the *Fix1* directory to configure their observation input settings (**Table 2**) prior to running the algorithm. AITCC version 1.0 requires users to select a single analysis height for efficient calculation (AITCC is based on 2D analysis). In a future upgrade (version 2.0), 3D analysis will be possible. At present, the 3D reflectivity field is required only for the elimination of stratiform precipitation using the BBF method (bright-band fraction method, Chen and Uyeda, 2003). Users must set ZA_NUM when the BBF method is activated. "ERROR_UNDEF" corresponds to missing values in the input data. "ERROR_VALUE" must be set slightly higher than "ERROR_UNDEF" (In this manual, option parameter name is delimited with double quotation marks).

Available options for the detection procedures are given in **Table 3**. There are two options for eliminating stratiform regions: one is the BBF method, and the other is Steiner's method (STN, Steiner *et al.*, 1995). If W_OPT is activated (W_OPT is set to 1), the input file must include the vertical velocity dataset. If the number of CCs within a single CG exceeds 1000, or if the area of a CC exceeds 2000 grid points, AITCC will stop. In such situations, users must increase the value of "MAX_CELL_IN_CELL" and "MAX_CELL_AREA" *in chaser.h*.

2.2 Definition of the target

Users should define the tracking targets (**Table 4**). "CUT_REF" indicates the noise level for reflectivity. Any values lower than "CUT_REF" (default: 15 dBZ) are assumed to be reflections from non-precipitation targets. "TRE_MCS" indicates the constant threshold value to identify CGs/MCSs. "MIN_AREA" defines the minimum area required to distinguish the precipitation target from the noise signal in the detection of CGs/MCSs. "ECHOTOP_DBZ" (default: 15 dBZ) indicates the reflectivity value used to measure the highest altitude where the value is observed. Users can select another reflectivity value to measure this height ("ECHOTOP_DBZ2", default: 30 dBZ).

AITCC has seven options for deleting uncertain convective cells in its detection procedure. "MIN_AREA_C" and "MAX_AREA_C" indicate the minimum and maximum areas of a convective cell, respectively. If "SWITCH_DEL_CELL1" is activated, the CCs that do not satisfy the above range of areas will be deleted.

To detect a meaningful peak within a CC, AITCC checks the difference between the peak value and the adaptive threshold value to distinguish the cell from the other cells. If the difference is less than "MIN_D_REF" (3 dB) and "SWICTH_DEL_CELL2" is activated, AITCC will delete such cells.

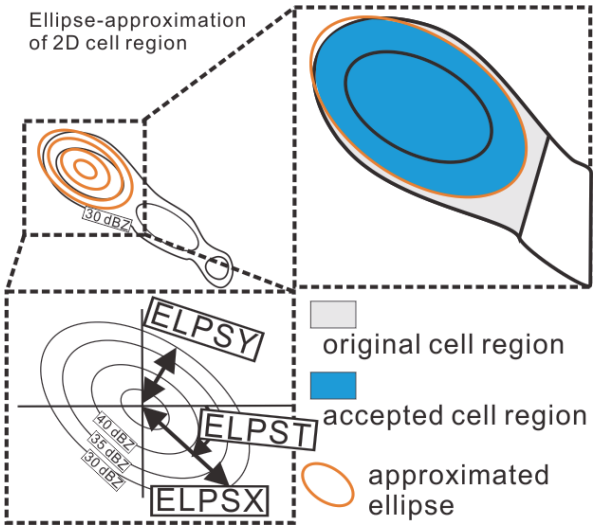


Fig. 1 Approximation of cell shape as an ellipse that best fits the 2D area of a cell (red contours). Ellipse parameters, such as major and minor radii ("ELPSX" and "ELPSY", respectively), and the orientation angle of the major axis relative to the x-axis ("ELPST"), are calculated. Cell grids that are inside the ellipse are accepted (blue region).

If "SWITCH_DEL_CELL3" is activated, CCs with a reflectivity peak adjacent to a region of missing data will be deleted because the detected peak is unreliable.

"SWITCH_DEL_CELL4" is an option used to delete cells lacking a reflectivity peak. In the detection procedure, AITCC fails to detect a peak of reflectivity within a CC when two (or more) adjacent grid points have the same reflectivity and the reflectivity is larger than that of their neighbors. In such cases, AITCC tries to generate a single artificial peak from among the adjacent peak grids. However, AITCC occasionally fails to detect the adjacent peak grids, especially when there are more than four. This happens often in the local peaks of stratiform precipitation regions. In such situations, the "SWITCH_DEL_CELL4" option or stratiform elimination option should be activated to avoid incorrect cell detection.

When two CCs whose peaks are located within "MIN_DIS_CORE" (default: 2 km) of each other and the "SWITCH_DEL_CELL5" is activated, the cell with weaker reflectivity is deleted.

Finally, AITCC checks whether or not the horizontal shape of the detected CCs is natural. All CCs approximate to an ellipse, and parameters such as the major and minor radii, and the orientation of the major axis relative to x-axis, are calculated (**Fig. 1**). The aspect ratio of a CC is calculated from the major and minor radii of the approximated ellipse. If "SWITCH_DEL_CELL6" is activated, the grids outside of the ellipse are deleted, and the cell area will be recalculated

(**Fig. 1**). If "SWITCH_DEL_CELL7" is activated, the cells whose aspect ratios are larger than "LIM_ASPCT" (default: 3.0) will be deleted.

The grids of the deleted cells, or some part of the cells, are redistributed to neighboring CCs around the deleted CC if "MERGE_DEL_CELL" (default: off) is activated.

2.3 AITCC detection algorithm

2.3.1 Preprocessing subroutines

Table 5 explains the subroutines used for preprocessing in the detection procedure. Input data should be 3D reflectivity data in a Cartesian coordinate system and in the NetCDF format. Details for the input data are provided in Appendix A. In *Select_height.c*, the user can specify the height of analysis.

2.3.2 Subroutines for the detection of cell groups or mesoscale convective systems

All programs for CG/MCS detection are included in *MCS_detect.c*. **Table 6** lists all subroutines in *MCS_detect.c*. **Figure 2** shows an example of the grid classification procedure conducted using the program *edge_MCS.c*. Each grid is classified into five classes: "CORE" is the local peak of eight neighboring points; "EDGE" is a grid point with a value higher than threshold, but one of four neighboring points lower than threshold; "OUTER" is a grid point with a value lower than threshold; "NODATA" is a grid point with a missing value; and "INNER" is a grid point with a value higher than threshold except for "OUTER" and "CORE".

A single CG/MCS is identified as a region enclosed by the "EDGE" grids in the program *name_MCS.c*. **Figure 3** indicates how the cell-group IDs are numbered. In the first step, AITCC searches "CG region", where the grid is classified as "EDGE", "INNER", or "CORE", from lower-left to upper-right. When the first grid point is found in the CG region, an ID of "1" is assigned to the grid point (see **Fig. 3**, STEP1). The same ID is then assigned to the next grid point unless the grid point is "NOT CG region", when the grid point is classified as either "OUTER" or "NODATA". If AITCC finds "NOT CG region", the ID-number increases (see **Fig. 3**, STEP1). After the search procedure in the first step, the ID number is given as shown in the upper row of **Fig. 3**. For the second step, if a given grid point at (x, y) and the upper grid point at (x, y + 1) have different IDs, the ID of the upper grid point is overwritten by the ID of the grid point at (x, y). Next, if a grid point at (x, y) and the grid point to the right at (x + 1, y) have different IDs, the ID of the grid point (x, y) is overwritten by the ID of the right-hand grid point at (x + 1, y). After the overwriting procedure in the second step, the ID is given as shown in the middle row of **Fig. 3**. For the third step, if two adjacent grid points have different IDs, the larger ID is overwritten by the smaller ID, and the same procedure is repeated for all other grid points possessing the larger ID. Following this uniting procedure, the ID number is given as shown in the lower row of **Fig. 3**. The first and second procedures are performed by the program *search_MCS.c*, and the third procedure by *unite_num_MCS.c*.

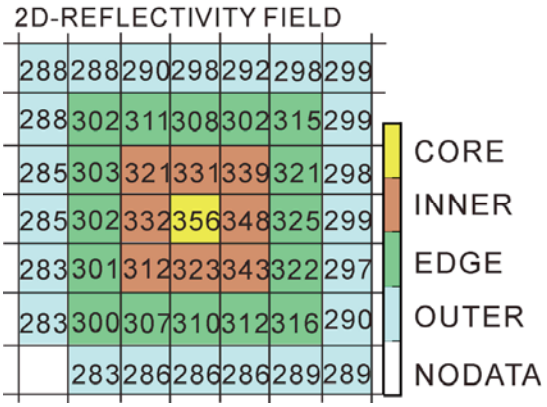


Fig. 2 Classification of grid points. Reflectivity values (×10) are shown at each grid point.

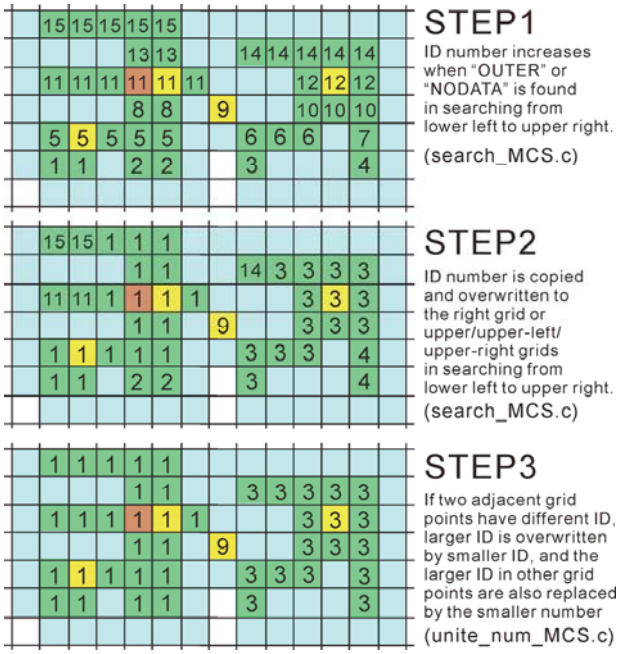


Fig. 3 Identification of CGs. The number indicates the ID number for each CG. The color scheme for the grid points is as for **Fig. 2**.

After generating the ID number for each CG/MCS, AITCC counts the number of CGs/MCSs (*count_MCS.c*). Next, AITCC counts the number of grid points in each CG/MCS to measure the area (*area_MCS.c*). If the area of a given CG is smaller than "MIN_AREA" (see **Table 4**), the CG will be deleted by the program *delete_MCS.c*. After deleting small CGs, a serial number is assigned to each CG/MCS (*arrange_MCS.c*). The maximum height of the CG/MCS is calculated using "ECHOTOP_DBZ" and "ECHOTOP_DBZ2" by the

program *height_MCS.c*. A reflectivity-weighted center of gravity is calculated for each CG/MCS (*center_MCS.c*). The number of the "CORE" grid is calculated for each CG/MCS to detect the maximum number of CCs within a CG/MCS (*peak_MCS.c*). The maximum reflectivity for each "CORE" grid point is obtained by the program *peak_value_MCS.c*. The shape parameters (major and minor radii of the approximated ellipse) of the CG/MCS are calculated by *Calc_Ellipse_MCS.c*. This shape parameter will be used to judge the similarity of CCs at two successive time points.

Figure 4 shows a sample classification of a grid point. Three CGs were detected. The southern CG contains four reflectivity peaks ("CORE": shown by yellow in the right panel of **Fig. 4**). The northern CG has 15 reflectivity peaks, but two were rejected (shown by dark green in the right panel of **Fig. 4**). The small northwestern CG was rejected because of its small area (just five grid points). In the next cell detection procedure, the CG grid points are divided into each CC region.

2.3.3 Subroutines for the detection of a convective cell

All programs for CC detection are included in *CELL_detect.c* (**Table 7**). A CG/MCS that has a single reflectivity peak becomes a straightforward CC (*direct_CELL.c*). For CGs/MCSs with multiple reflectivity peaks, AITCC divides the region into individual CCs. AITCC adds a check flag to the "already-defined cell region" (*reserve_CELL.c*), which is defined by the program direct_CELL.c or by the previous detection loop using a lower threshold value.

The cell detection algorithms listed in **Table 7** between the *edge_CELL.c* and *count_CELL.c* programs are the same as the CG detection algorithms listed between *edge_MCS.c* and *count_MCS.c* in **Table 6**, except for the threshold value. The reflectivity region enclosed by a given threshold value is referred to as a "segment" in this manual. The area of a segment will be smaller when a higher value threshold is used, as shown in **Fig. 5**. When the threshold is set to 30 dBZ, the segment is identical to a CG. AITCC increases the threshold and the program *peak_CELL.c* checks the number

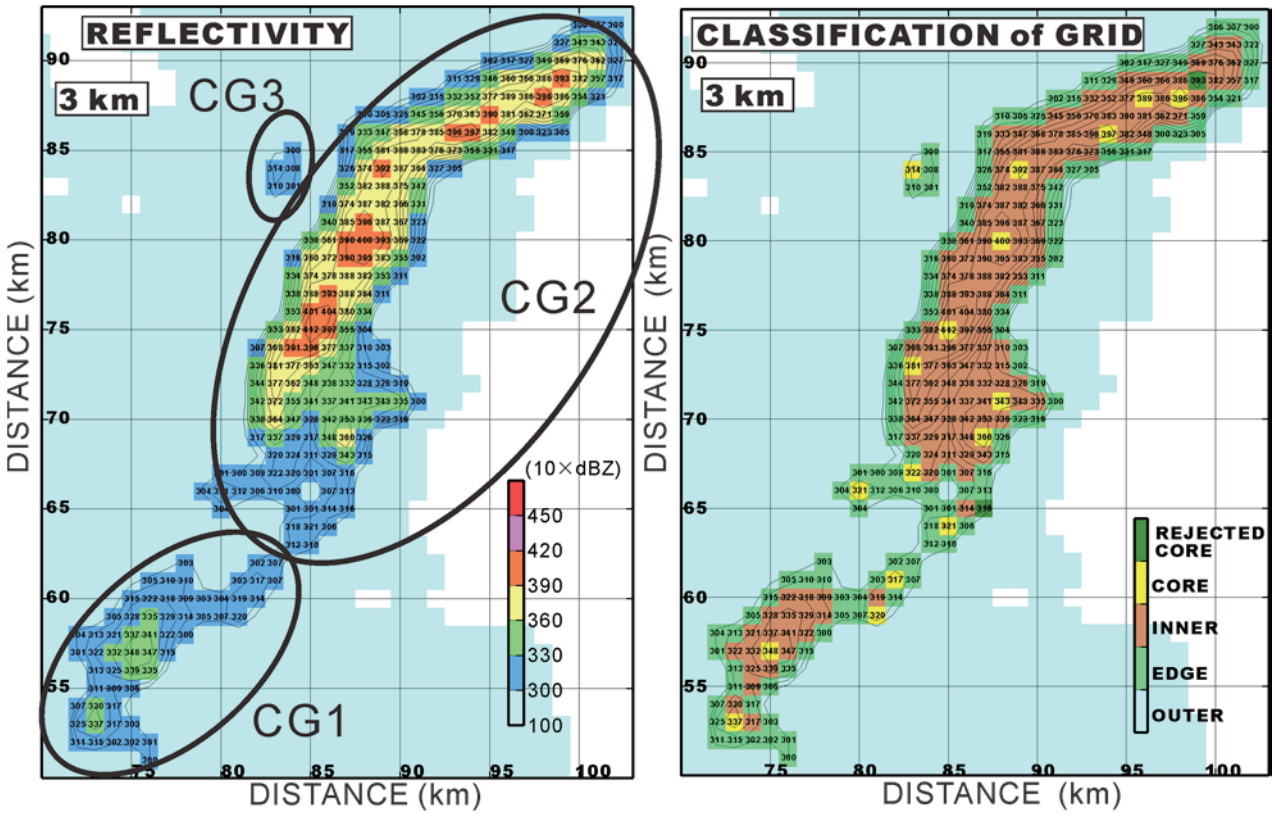


Fig. 4 Sample classification of a grid point. The left panel shows reflectivity value (×10). Contours indicate reflectivity values every 1 dBZ starting from 30 dBZ. The right panel shows the classification state of each grid point.

of reflectivity peaks for each segment. When the segment includes a single peak of reflectivity for a given threshold value, the segment directory becomes the cell region (**Fig. 5C** and **D**) in the program *divide_CELL.c*.

At the same time, the program *check_core_CELL.c* finally checks whether the cell region includes another peak using a higher threshold (**Fig. 6**). If the program *check_core_CELL.c* finds another peak, the division process for the segment with no peak will continue after the following artificial peak-adding process.

When the segment includes no peaks (see lower-right corner in **Fig. 7B** and **F**), the program *add_peak_CELL.c* will add an artificial peak at the "center of gravity" of the grid points with a reflectivity that is relatively higher than that of the other grid points (**Fig. 7C**). If the center of gravity is outside of the segment (see **Fig. 7F**), the program *add_peak_CELL.c* cannot add an artificial peak. In such cases, the segment with no peak is labeled "LOST_SEGMENT" by AITCC (see **Fig. 7G**). The grid points in the "LOST_SEGMENT" are assigned to the cell region whose CORE is closer to the grid point (**Fig. 7H**) by the programs *lost_CELL.c* and *AROUND_CORE.c*.

The lack of a peak in the segment is caused because two or more adjacent peaks have the same value, as shown in **Fig. 7**. If the cell deleting option "SWITCH_DEL_CELL4" is activated, such cells are finally deleted by the program *delete_CELL.c*.

Table 7 List of subroutines used in *CELL_detect.c*.

Subroutine name	Description
direct_CELL.c	Detection of single cells
reserve_CELL.c	Set flag for already-defined cell
edge_CELL.c	Same as *edge_MCS.c* but for CC
name_CELL.c	Same as *name_MCS.c* but for CC
BL2TR_CELL.c	Same as *search_MCS.c* but for CC
check_num_cell.c	Same as *unite_num_MCS.c* but for CC
count_CELL.c	Same as *count_MCS.c* but for CC
peak_CELL.c	Check if the segment includes a single peak of reflectivity
add_peak_CELL.c	Add artificial "CORE" grid for segment without any peak
check_core_CELL.c	Final check if the segment includes a single peak of reflectivity
divide_CELL.c	Secure cell area following procedure shown in Fig. 2 of **SU12**
near_edge.c	Distribution of undefined cell region to closer cell (see **Fig. 5**)
SEARCH_AROUND_CELL.c	Distribute undefined cell region to adjacent cell with strongest reflectivity
fix_CELL.c	Final determination of cell region
lost_CELL.c	Assign cell ID for grid points in lost segment
AROUND_CORE.c	Search the cell whose CORE is nearest to the grid points in lost segment
area_CELL.c	Same as *area_MCS.c* but for CC
attach_CELL.c	Join cells whose peaks are located next diagonally.
away_CELL.c	Set flag for cells whose grid points have same ID but are detached.
enclose_CELL.c	Redistribute detached grid points for other cell.
Calc_ellipse.c	Same as *Calc_ellipse_MCS.c* but for CC
delete_CELL.c	Delete CC following deleting option discussed in Sec. 2.2
arrange_CELL.c	Same as *arrange_MCS.c* but for CC
peak_Locate_CELL.c	Get position of detected cell
peak_value_CELL.c	Same as *peak_value_MCS.c* but for CC
center_CELL.c	Same as *center_MCS.c* but for CC
W_CELL.c	**[Optional]** Calculate area/velocity of updraft/downdraft in CC
height_CELL.c	Same as *height_MCS.c* but for CC

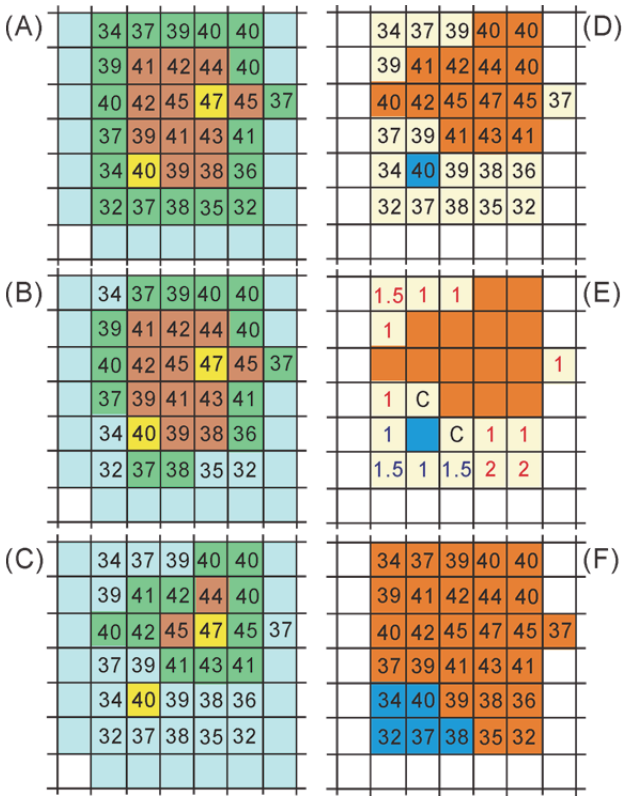


Fig. 5 Schematic representation of the determination of a cell region. Panels on the left indicate the grid point classification using different thresholds (A: 30 dBZ, B: 35 dBZ, C: 39 dBZ) and the number in each grid point indicates the reflectivity value. The color scheme of grid points in the left panels is as in **Fig. 2**. When using a threshold of 39 dBZ, the program peak_CELL.c detects a single core in two segments (C). The right column indicates how to distribute an undefined cell region (less than 39 dBZ: light yellow in D) to a defined cell region. The red grid points and a blue grid point (D) indicate an "already defined" cell region (>39 dBZ in D). The minimum distance from the undefined-grid point to the already-defined grid point is shown in (E). The color of the number indicates the nearest cell region. The letter "C" in (E) indicates "competition" for two or more cell regions at the same distance. In the competition grid, the grid points belong to the cell region with a stronger reflectivity at the grid point that is nearest to the competition grid. After distribution for the whole "undefined region", the cell region is finally determined (F).

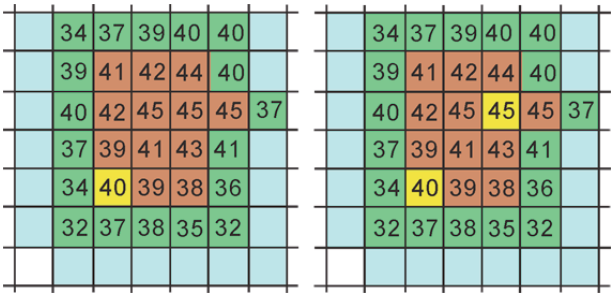


Fig. 6 Schematic representation of check_core_CELL.c. The colors and numbers in the grids are as in **Fig. 2**.

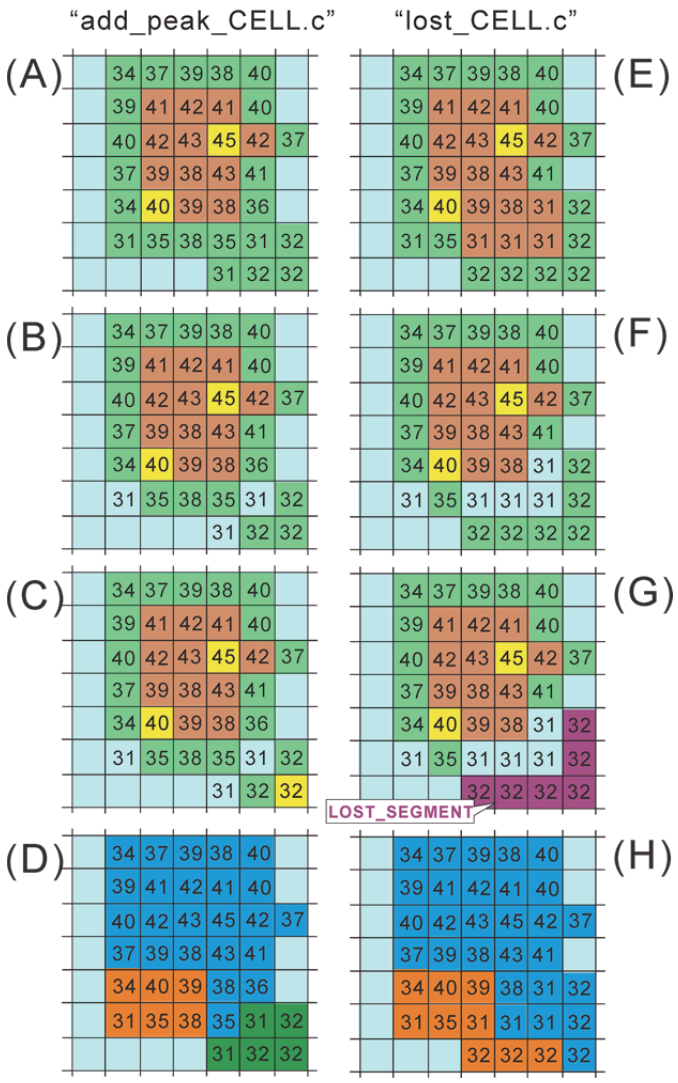


Fig. 7 Schematic representation of *add_peak_CELL.c* (left panels) and *lost_CELL.c* (right panels). The colors and the numbers in each grid (A–C and E–G) are as in **Fig. 2**. (D) and (H) indicate the final cell region (red, blue, green grids). The purple grid in (G) indicates "LOST_SEGMENT".

The accepted segment and neighboring "undefined" grid points of the segment finally become the cell region (**Fig. 5D–F**). The program *near_edge.c* finds the closest cell to the "undefined" region (**Fig. 5E**). The undefined region is distributed to the cell region that is closer to the grid point of the undefined region (see **Fig. 5E**).

If the program *near_edge.c* is unable to find an appropriate cell because of a failure to calculate the distance from the cell region to the grid point, the program *SEARCH_AROUND_CELL.c* assigns the undefined grid points to the nearest cell region with the strongest reflectivity value on their boundary. The program *fix_CELL.c* finally determines the cell region from the "undefined" grid points and cell region. More details regarding the detection of CCs can be found in Section 2.3 of **SU12**. If cell IDs are assigned to all grid points, the loop of increasing threshold values is terminated by the program *CELL_detect.c*. AITCC proceeds to cell analysis after deleting uncertain cells.

The area of a CC is calculated by the program *area_CELL.c*. If two COREs are located diagonally opposite to each other (e.g., (x, y) and (x – 1, y + 1), or similar), the program *attach_CELL.c* merges the two cells into one cell by deleting one of the two peaks (the weaker CORE will be deleted).

If some grid points have the same cell ID but the grid points are detached (although this is fairly rare), the smaller cell will be attached to a previously defined neighboring cell. The program *enclose_CELL.c* selects the appropriate cell that shares the largest number of grid points on the boundary of the smaller cell.

The shape parameters of a CC, such as the major and

minor radii of the ellipse by which the 2D cell region is approximated, are calculated by *Calc_ellipse.c*, which is similar to *Calc_Ellipse_MCS.c*.

Uncertain cells that satisfy one of the seven deletion conditions (see Sec. 2.2) are deleted by *delete_CELL.c*. After deletion, *arrange_CELL.c* allocates a serial number to each CC. The location of the reflectivity peak in a CC is calculated by *peak_Locate_CELL.c*, and the reflectivity value is obtained by *peak_value_CELL.c*. The reflectivity-weighted "center of gravity" of each CC is calculated by the program *center_CELL.c*, and the echo-top height is calculated by *height_CELL.c*. If the option "W_OPT" is activated, the maximum updraft and downdraft velocities in each CC are calculated, and the number of updraft/downdraft cores in each CC is also calculated by *W_CELL.c*.

All information regarding the identified CCs and CGs/MCSs at a given time is saved as one NetCDF file (see Appendix B). This NetCDF file includes not only the cell characteristics (i.e., dimensions, location, and strength), but also the temporal linkage information for the previous, present, and next time steps (i.e., parent ID and child ID). The temporal linkage information is empty when the detection algorithm has finished. The following procedure described in Section 2.4 will fulfill this information.

2.4 AITCC tracking algorithm

The goal of tracking is to link targets (CCs or CGs/MCSs) in consecutive radar images at times T and T + dt if they represent the same phenomenon. After detecting all CGs and CCs at all time steps, AITCC assigns the CGs between t = T and t = T + dt. Next, AITCC tracks the CCs in addition to the CGs. This order is followed because it is much easier to track CGs than to track CCs. The flowcharts for tracking CGs and CCs between two time steps are shown in Tables 3 and 4 of **SU12**. The details of the algorithm are also given in **SU12**. Therefore, this manual explains only the role of each subroutine in the tracking algorithm.

2.4.1 User configuration for tracking

Users can edit the file *chaser2.h* in the directory *Fix2* to configure the tracking rules for CGs (**Table 8**).

AITCC uses the "accumulated occupation ratio" (AOR) and "inhibiting crossing" (INH) methods to limit uncertain CG linkage (see **SU12**). A third method, the principal vector (PV) method, is used as well, but only for CC tracking. When the options "RELATIVE_MOTION_A" or "RELATIVE_MOTION_B" are activated, the PV method is also used for CG tracking. The former is used to merge judgments, and the latter to split judgments.

If the option "BK_RSTRIC" is activated, CG splitting is not allowed. For instance, if two CGs at t = T + dt are the candidate child of a CG at t = T, one of the two becomes the child, while the other is assumed to have been generated at t = T + dt.

AITCC estimates the moving vector (MV) of each CG, and the optimal MV that maximizes the area of CG overlap between t = T and t = T + dt is estimated. This method is referred to as the overlapping-area (OVR) method in AITCC (for details, see Fig. 3 of **SU12**). In the OVR method, users can impose the search angle and range limitation ("SEARCH_ANGLE" and "SEARCH_DIS").

In the AOR method, the area conservation of a CG (CC) is considered when merging or splitting occurs. If the sum of merging two CGs exceeds that of the child-CG, the merging is not allowed. When a child-CG has multiple candidate parents, AITCC calculates the ratio of the area of each parent to that of the child (the "occupation ratio"), which corresponds to A_{pi}/A_c in Eq. 1.

$$AOR = \sum_{i}^{n<N} \frac{A_{pi}}{A_c} \tag{1}$$

If (1 – AOR) < "MIN_AREA_R" when AOR is integrated to the i^{th} CG candidate, AITCC assumes that the next merging candidate is not allowed. "MIN_AREA_R" is the minimum area occupation required to allow more merging/splitting candidates to be generated. In the default, "MIN_AREA_R" is set to 0.25 (25 %).

In the CG/MCS tracking, the initial MV is calculated by cross correlation (CRR) between t = T and t = T + dt.

$$CRR(\delta x, \delta y, t) = \frac{\sum_{x,y} REF(x+\delta x, y+\delta y, t) * REF(x, y, t+\delta t)}{\sqrt{\sum_{x,y} REF(x+\delta x, y+\delta y, t)^2} \ * \sqrt{\sum_{x,y} REF(x, y, t+\delta t)^2}} \tag{2}$$

Here, δx and δy represent a horizontal shift in the 2D reflectivity field. Users can select the range of the shift using the parameters "CR.XM" and "CR.YM" in the x-direction and y-direction, respectively. AITCC finds the values of δx and δy that maximize the CRR (Eq. 2) and uses it as the initial MV.

2.4.2 Subroutines for tracking a CG

All programs for CG tracking are included in *MCS_chase.c*, and **Table 9** lists all of the subroutines contained in the program. As a preprocess to CG tracking, the program *check_file.c* opens two files at successive times, and checks the consistency of the size dimensions (XNUM, YNUM, ZNUM) between the two files. The program *secure_mem.c* obtains memory for the CG-tracking calculation. The program *read_data.c* reads the two NetCDF files and extracts the necessary information for CG tracking from them.

AITCC calculates the cross correlation (CRR) using Eq. 2 for each shift of δx and δy (–CR.XM ≤ δx ≤ CR.XM, and –CR.YM ≤ δy ≤ CR.YM), and saves all coefficients as a

matrix. The program *INIT_CROSS.c* initializes the matrix. Before estimation of the movement vector (MV) for each CG, AITCC tries to estimate the global movement vector that represents the average movement across the whole precipitation area within the analysis domain. The program *CRS_CRR.c* calculates the global movement vector that maximizes the cross correlation across the whole analysis domain. The global movement vector will be used in the inhibiting-crossing procedure. The program *CRS_CRR1.c* is the same as *CRS_CRR.c*, except that the analysis domain is the minimum rectangular region that includes the whole area of a CG, and CRR is calculated from the grid points with reflectivity more than 30 dBZ. The program *CRS_CRR1.c* generates an initial estimate of MV for each CG. The rectangle is then displaced by a given number of grid points in all directions, assuming that the movement speed is less than 1 km/min. The program *CRS_CRR1.c* finds three different MVs that give the first, second, and third largest CRR. If a CG has a parent-CG in the previous tracking, the MV of the parent-CG is used as a first estimate of the MV of the CG, and AITCC skips the estimation process of MV in the program *CRS_CRR1.c*.

The first estimate of the MV for each CG is modified by the overlap method (OVP) in the program *OVERLAP.c*. The OVP algorithm finds the optimal MV that maximizes the area of CG overlap between t = T and t = T + dt (see Fig. 3 in **SU12**). A parent-CG is translated using MVs modified by adding small perturbations in direction and length to account for the errors in MV estimation, and these perturbations are user defined ("SEARCH_ANGLE" and "SEARCH_DIS" in **Table 8**). If a parent-CG is translated using an MV and an overlapping area with a CG at t = T + dt is found for at least one grid point, then the CG at t = T + dt will be a candidate child-CG for the parent-CG. The MV vector that forms the maximum overlapping area is saved. If the parent-CG is translated using a different MV and an overlapping area with a different CG at t = T + dt is found, then the CG at t = T + dt will also be a candidate child-CG for the parent-CG. A parent-CG can have at most six candidates. If users set larger values for "SEARCH_ANGLE" and "SEARCH_DIS", the higher the number of candidates will be.

For the CG that does not inherit the MV from a parent-CG, the program *CRS_CRR1.c* provides three different first estimates of MV, and the program *OVERLAP.c* modifies each of the three MVs and finds several candidate child-CGs for each first estimate of MV. The program *selectNO.c* checks whether or not the several CG candidates are identical, and extracts the non-identical candidates. After this selection, the program *clean_Vtemp.c* cleans up the memory array for the first estimate of MV.

Figure 8 outlines the conceptual model used to determine the assignment vector between CGs/CCs at two successive time steps. The parent-CG finds candidate child-CGs using the overlap method (**Fig. 8A**). If multiple parent-CGs have the same candidate child-CG (**Fig. 8B**), AITCC judges whether or not the merging is to be accepted using the program *MERGE_DISSIPATE2.c*. A penalty function (see Eq. 4 of **SU12**) determines the priority of the parents as a candidate to be matched with the child. If the merging is not allowed, the parent-CG with a lower priority will not be assigned as the parent to the child (**Fig. 8C**). The program *FILL_MBOX.c* counts the number of parent-CGs for a given child-CG. The program *DECIDE_RATIO.c* calculates the penalty function to prioritize the candidate's parent-CGs.

If the PV (Principal Vector) option is activated ("RELATIVE_MOTION_A" in the *chaser2.h* is set to 1), the programs *CHECK_CRR.c* and *CHECK_OVR.c* calculate MV using the CRR and OVR method for the parent-CG that assigned the highest priority to a given child-CG to deduce the PV (see details in **SU12**). The PV method will restrict the merging that does not conserve relative locations between two merging parent-CGs.

After the forward-in-time procedure, AITCC looks for unassigned child-CGs (termed "Alone-Child-CG"). A candidate parent-CG is sought for each Alone-Child-CG, whereby candidate parents are found by the overlap method backwards in time (**Fig. 8D**). If multiple child-CGs have the same parent-CG, AITCC judges whether the splitting is to be accepted or not (**Fig. 8E**) using the program *SPLIT_GENERATE2.c*. For the splitting judgment, the programs *FILL_MBOX_b.c* and *DECIDE_RATIO_b.c* work in a similar manner to *FILL_MBOX.c* and *DECIDE_RATIO.c*, except backwards in time. Finally, AITCC categorizes all parent-CGs into three stages: maintaining, merging, and dissipating, and also categorizes all child-CGs into three stages: maintaining, splitting, and newly generated (**Fig. 8F**).

After the splitting judgment, the program *MULTI_COUNT.c* checks whether the assignment in the backward-in-time process has been duplicated with that in the forward-in-time process, and checks whether the assignment in the backward-in-time process has been rejected by the forward-in-time process.

AITCC checks whether each pair of assignment vectors is crossing using the programs *INH_CROSS.c* and *CHECK_CROSS.c*. If there is a crossing assignment, the program *SELCT_UV.c* selects an assignment vector that is similar to the global movement vector. The similar assignment vector is saved, and the other is rejected (see Fig. 5a of **SU12**).

2.4.3 Subroutines for tracking a CC

After applying the CG-tracking procedure, CC tracking is performed in a similar manner to CG tracking. The difference between CG tracking and CC tracking is summarized

in section 2.4 of **SU12**. This document focuses on an explanation of the programing used in CC tracking. **Table 10** lists the subroutines used for CC tracking. Most of the subroutines are similar to those used in CG tracking. The main differences between CG tracking and CC tracking are as follows: (1) the first estimate of MV for a CC is given by the MV of a CG; (2) there is no calculation of CRR for a CC; and (3) the allowance of the additional assignment between two parallel neighboring assignments in the program *Inbetween.c* (APN, see details in Fig. 5 of **SU12**). In the case of CC tracking, the cell that reaches the lateral boundary region is not analyzed. The program *NEAR_BOUNDARY.c* excludes the cell around the lateral boundary from analysis.

Table 8 List of parameters for input data in *chaser2.h*.

Parameter	Description	Default setting
RELATIVE_MOTION_A	Use of principal vector method in merging judgment	1 (activate)
RELATIVE_MOTION_B	Use of principal vector method in splitting judgment	1 (activate)
BK_RSTRIC	Inhibiting splitting	0 (inactivate)
SEARCH_ANGLE	Maximum angle to search candidate in OVERLAP method	30 (°)
SEARCH_DIS	Maximum range distance to search candidate in OVERLAP method	2.0 (km)
MIN_AREA_R	Minimum ratio of overlapping area to the original area	0.25
CR.XM/ CR.YM *1	Range of cross correlation in x- and y- directions	5 (grid)

*1 CR.XM and CR.YM are described in *Fix2/main.c*

Table 9 List of subroutines in *MCS_chase.c* for CG-tracking procedure.

Subroutine name	Description
INIT_CROSS.c	Initialize the cross-correlation array
CRS_CRR.c	Calculate CRR to obtain global MV
CRS_CRR1.c	Calculate CRR, ignoring grid points where reflectivity is less than 30 dBZ
OVERLAP.c	Estimate/modify the MV, and find tracking candidate
selectNO.c	Extract different MVs from 3 candidates of MVs determined by the OVR method.
clean_Vtemp.c	Initialize MV
MERGE_DISSIPATE2.c	Judgment of merging for multiple linkage candidates from t = T to t = T + dt
FILL_MBOX.c	Counting the number of parent candidates for a given child-CG.
DECIDE_RATIO.c	Calculate penalty function to prioritize the candidate's parents.
CHECK_CRR.c	Calculate first estimate of MV for parent-CG assigned the highest priority to a child-CG using CRR method.
CHECK_OVR.c	Modify the first estimate of MV using OVR method, and check if the overlapping area is found for at least one grid point when using the MV.
SPLIT_GENERATE2.c	Judgment of splitting for multiple-linkage candidates from t = T + dt to t = T
FILL_MBOX_b.c	Counting the number of child candidates for a given parent-CG.
DECIDE_RATIO_b.c	Counting the number of child candidates for a given parent-CG.
MULTI_COUNT.c	Check if the assignment in the backward-in-time process is not duplicated with that in the forward-in-time process
INH_CROSS.c	Inhibit crossing assignment.
SELECT_UV.c	Selection of assignment vector that is similar to global MV.
CHECK_CROSS.c	Check whether or not the two assignments are crossing.

MV: moving vector: CRR: Cross correlation

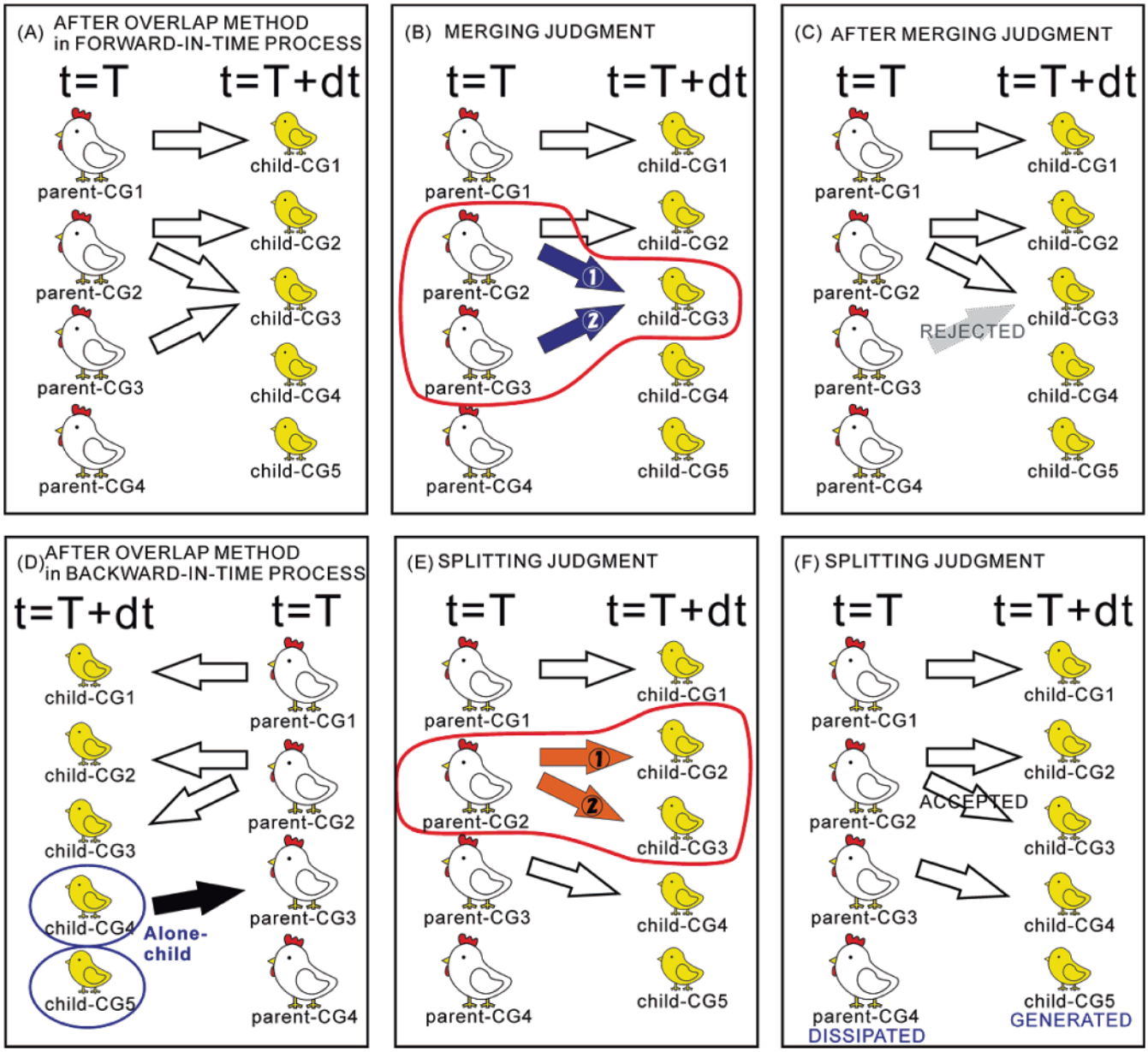


Fig. 8 Conceptual model of assignment between CGs/CCs at two successive time steps. The open arrow indicates a candidate assignment vector. Four CGs at the present time ("parent-CG") and five CGs at the sequential time step ("child-CG") are assumed. (A) Each Parent-CG finds candidate child-CGs by the overlap method in the forward-in-time procedure. (B) Merging is judged if multiple parent-CGs have the same child-CG. The blue vector indicates the merging assignment vector. The number in the arrow indicates the priority of the assignment vector. (C) After the merging judgment, the merging assignment vector with the lower priority will be rejected if the merging assignment is not allowed. (D) An alone-child-CG can find a parent-CG by the overlap method using the backward-in-time procedure. (E) Splitting is judged if multiple child-CGs have the same parent-CG. (F) As with the merging judgment, the splitting judgment determines whether the splitting assignment vector is accepted or rejected. Finally, a parent-CG that does not have a child is judged to be "dissipated", and a child-CG that does not have a parent is judged as "new generated", and the other child- or parent-CGs are judged to be maintained.

Table 10 List of subroutines in *CELL_chase.c* for CC-tracking procedure.

Subroutine name	Description
NEAR_BOUNDARY.c	Check if the cell area border is on a lateral boundary
OVERLAP.c	Estimate/modify the MV, and find tracking candidate
CHECK_SPLIT.c	Apply AOR method to limit splitting assignment vector
OVERLAP.c	Estimate/modify the MV, and find tracking candidate
MERGE_DISSIPATE2.c	Same as *MERGE_DISSIPATE2.c* in **Table 9** but for CC tracking
FILL_CBOX.c	Same as *FILL_MBOX.c* in **Table 9** but for CC tracking
DECIDE_RATIO.c	Same as *DECIDE_RATIO.c* in **Table 9** but for CC tracking
CHECK_CRR.c	Same as *CHECK_CRR.c* in **Table 9** but for CC tracking
CHECK_OVR.c	Same as *CHECK_OVR.c* in **Table 9** but for CC tracking
SPLIT_GENERATE2.c	Same as *SPLIT_GENERATE2.c* in **Table 9** but for CC tracking
FILL_CBOX_b.c	Same as *FILL_MBOX_b.c* in **Table 9** but for CC tracking
DECIDE_RATIO_b.c	Same as *DECIDE_RATIO_b.c* in **Table 9** but for CC tracking
MULTI_COUNT.c	Same as *MULTI_COUNT.c* in **Table 9** but for CC tracking
INH_CROSS.c	Same as *INH_CROSS.c* in **Table 9** but for CC tracking
SELECT_UV.c	Same as *SELECT_UV.c* in **Table 9** but for CC tracking
CHECK_CROSS.c	Same as *CHECK_CROSS.c* in **Table 9** but for CC tracking
Inbetween.c	Find assignment vector between two parallel neighboring assignment vectors
ORDER_PRIMARY.c	Reorder the linkages as the order of priority descends
DETERMINE_ORDER_a.c	Recalculation of penalty function for parent-CCs
DETERMINE_ORDER_b.c	Recalculation of penalty function for child-CCs
FINAL_REPLACEMENT_VELOCITY.c	The MV of CC is replaced by the movement of "center of gravity"

Table 11 List of subroutines for CG joint procedure.

Subroutine name	Description
main.c	Main driver to joint tracking segments between two sequential times to deduce lifetime of CGs.
ReadLIST.c	Read list of NetCDF files and define the total time step
ReadSTS.c	Read NetCDF file to get status ID
LIFE_MCS.c	Find a generating CG and start jointing
FILE2TIME.c	Read UNIXTIME and convert it to date and time information
SEARCH_MCS.c	Get ID for CGs with a given "status of CG's life-stage" *1
RECURSE.c	Recursive processing to find CGs at next time connected to a CG at present time until finding CG at the next time with the status of "**DIS**" or "**END**".
INQ_MCS_NUM.c	Inquire the number of CGs at the time recorded in the opening NetCDF file
INQ_MCS_STATUS.c	Get ID number for the "status of CG's life-stage" recorded in NetCDF file
INQ_MCS_NEXT.c	Inquire the CG ID at the next time and the number of CGs at the next time
chaser3.h	MAX_CHAIN determines the maximum lifetimes (defined as MAX_CHAIN × dt)

*1: "status of CG's life-stage" is defined in text.

2.5 AITCC joint algorithm for CG/CC tracking

The CG/CC jointing procedure joints tracking segments between two sequential time points determined by the previous CG-tracking/CC-tracking procedure. The purpose of the jointing procedure is to create a history file for the individual CG/CC. The history file includes temporal evolution information for an individual CG/CC such as lifetime, history of location, history of CG/CC dimensions (e.g., height, area), and history of CG/CC intensity (e.g., the maximum reflectivity).

Table 11 shows the list of subroutines used in the jointing procedure for a CG. Output from the detection and tracking algorithms is based on geophysical grid data at one time point. Each file includes the CG/CC information at the time step, and the relationship of the CG/CC to the previous and next time steps. The program *ReadLIST.c* reads time information from all of the individual NetCDF files to determine the total time steps. The jointing analysis is conducted over the total period covered by all time steps.

In the CG-tracking/CC-tracking algorithm, AITCC assigns a "status of life-stage" to all CGs/CCs. The "status of life-stages" are categorized into six groups: (1)"**INI**": CG/CC observed at the initial time of analysis; (2)"**SRT**": CG/CC observed at one time step (referred to as an instantaneous CG/CC in **SU12**); (3)"**GEN**": generating CG/CC observed at the time recorded in the NetCDF file; (4)"**DIS**": CG/CC dissipated at the time recorded in the NetCDF file; (5)"**MAN**": CG/CC maintaining at the time recorded in the NetCDF file; and (6)"**END**": CG/CC observed at the end time of analysis. The program *ReadSTS.c* reads the ID for the six categories from the NetCDF files.

This jointing procedure searches the time step at which a given CG/CC first generates in the program *LIFE_MCS.c*. If AITCC finds a generating CG/CC at a given time step ("time A"), it checks the number of the child-CG/CC from the generating CG/CC, and investigates the status of life-stage of the child using *SEARCH_MCS.c*.

If there is no child for the generating CG/CC, the CG/CC is categorized as an "instantaneous CG/CC", and the jointing procedure is terminated. Such instantaneous CGs/CCs are usually excluded from analysis because they may be observational noise, or their lifetimes may be too short to be resolved by the temporal resolution used in the analysis.

If there is more than one child-CG/CC to the generating CG/CC, AITCC searches the number and status of the grandchild recursively until the "status of life-stage" of the child becomes "**DIS**" or "**END**" in the program *RECURSE.c*. The time step when the "status of life-stage" becomes "**DIS**" ("time B") is the end of the history of the CG/CC generated at time A. AITCC saves the history of the CG/CC from "time A" to "time B" as a single NetCDF file. If there are multiple child-CGs/CCs for a given parent-CG/CC (splitting is accepted for he parent-CG/CC), recursive processing is conducted for each of the multiple child-CGs/CCs. Finally, two history files are generated for the two child-CGs/CCs. In the case of the CC jointing procedure, if users set the option "SELECT_MAJOR_CONNECTION" to 1 in chaser5.h, only one child-CC with the highest priority is jointed to the parent-CC, and a single history file is generated. **Figure 9** contains a sample history file, with the upper diagram showing a history file with the option of "SELECT_MAJOR_CONNECTION" and the lower diagram without the option. **Table 12** is same as **Table 11** but for CC jointing procedure.

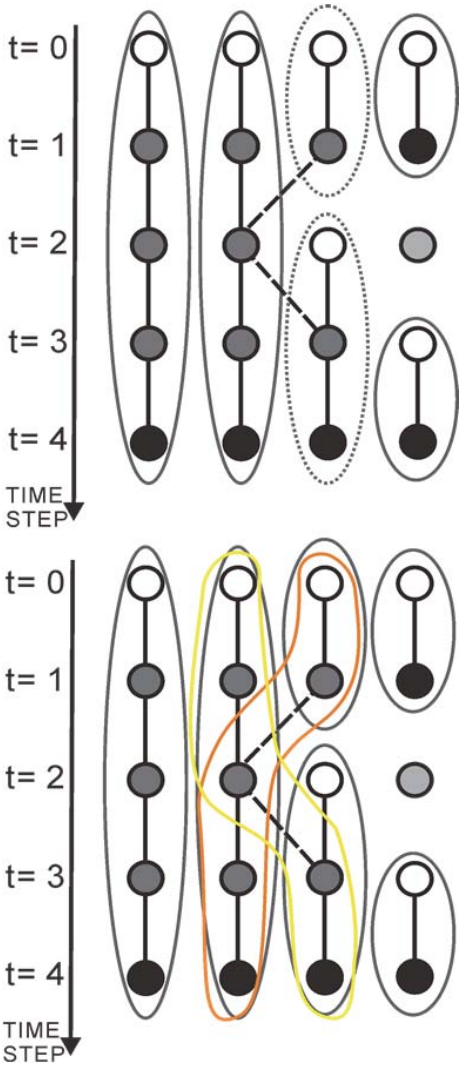


Fig. 9 Sample of history files with the option "SELECT_MAJOR_CONNECTION" (upper), and without the option (lower). Orange and yellow lines indicate history files that include double counting.

Table 12 List of subroutines for CC joint procedure.

Subroutine name	Description
main.c	Main driver to joint tracking segments between two sequential times to deduce the lifetime of CCs.
ReadLIST.c	Same as in **Table 11**
ReadSTS.c	Same as in **Table 11**
LIFE_CELL.c	Same as in **Table 11** but for CC jointing
FILE2TIME.c	Same as in **Table 11**
SEARCH_CELL.c	Same as in **Table 11** but for CC jointing
SEARCH_SPLIT.c	Check if the parent-CC has splitting
RECURSE.c	Same as in **Table 11** but for CC jointing
INQ_CELL_NUM.c	Same as in **Table 11** but for CC jointing
INQ_CELL_STATUS.c	Same as in **Table 11** but for CC jointing
INQ_CELL_NEXT.c	Same as in **Table 12** but for CC jointing
INQ_connect_order.c	Inquire the order of priority for each child-CC
Chaser5.h	Same as chaser3.h in **Table 11** except for "SELECT_MAJOR_CONNECTION", which controls the option to assume whether the minor-splitting child is treated as a newly-generated child.

2.6 Drawing tool for presenting AITCC results

There are two different types of AITCC output. One is a data grid object, and the other is a history file for an individual CG/CC. The former can be easily drawn using GMT (Graphical Mapping Tools), and users can draw the distribution of reflectivity with the cell location (**Fig. 10**). Users should modify both *option.pl* and *main.pl* in the GMT/Perl01 directory. In option.pl, the parameter "NCFdir" sets the path for the NetCDF data. In main.pl, the user must set the dimensions of the total domain (i.e., "X_MIN", "X_MAX", "Y_MIN", and "Y_MAX", although "X_MIN" and "Y_MIN" are usually zero). The user can set the drawing domain by changing parameters such as "FIG_LBX", "FIG_LBY", and "FIG_XY". The parameters "FIG_LBX" and "FIG_LBY" indicate the starting grid number, which corresponds to the southwestern corner of **Fig. 10**. The value of "FIG_XY" indicates the range from the southwest corner. If "FIG_XY" is set to be small value the user can show the detailed structure within a small domain.

A line-shaped CG is shown in **Fig. 10a**. Several CCs are detected within the line-shaped CG. Some CCs are rejected in the detection procedure due to unnatural shape (e.g., the region reflectivity peak region between cell 69 and cell 65 has a too large aspect ratio) or due to minimum area threshold (e.g., the small reflectivity region to the right of cell 69). If "W_OPT" is activated, the location of updraft/downdraft core can be drawn (**Fig. 10b**).

Before running perl scripts, user should install "NetCDFPerl" module from web page of University Corporation for Atmospheric Research (UCAR):

http://www.unidata.ucar.edu/software/netcdf-perl/

For the install guide of NetCDFPerl module, the following web page is available (in Japanese):

http://shimizus.hustle.ne.jp/wiki/wiki.cgi?page=NetCDFPerl

To download source files of GMT, the following web page is available:

http://gmt.soest.hawaii.edu/home

There are three major programs to draw figure. (1) *GMT/Perl01/main.pl* for the drawing the CC distribution at a given time (**Fig. 10**), (2) *GMT/Perl3/main3.pl* for the drawing the linkage information of CC at three sequential time steps (**Fig. 11**), and (3) *GMT/Perl5/draw_life.pl* to show the result of statistical analysis (e.g., frequency distribution for the lifetime of CC in **Fig. 12**).

To check individual history file made by jointing procedure, free software *ncview* provided by Dr. David Pierce is very useful (sample is shown in **Fig. 13**), and the program is available from the following web page:

http://meteora.ucsd.edu/~pierce/ncview_home_page.html .

To run the *GMT/Perl01/main.pl*,

perl main.pl [year] [mon] [dd] [hh] [mn] [height level]

Here, *[year]*, *[mon]*, *[dd]*, *[hh]*, and *[mn]* are date information for snapshot. The *[height level]* indicates the layer number to draw distributions. For the *main3.pl*, the above arguments are same but for the date indicating center time of three time steps (see **Fig. 11**).

To run the *GMT/Perl5/draw_life.pl*,

perl draw_life.pl

Finally, user can obtain Post Script file generated by GMT commands in GMT/psfile directory.

To customize drawing script for users, users need to modify other perl scripts as well as *main.pl (main3.pl)* in *GMT/Perl01* directory (*GMT/Perl3* directory). **Table 13** explains the role of each perl subroutine in *GMT/Perl* directory.

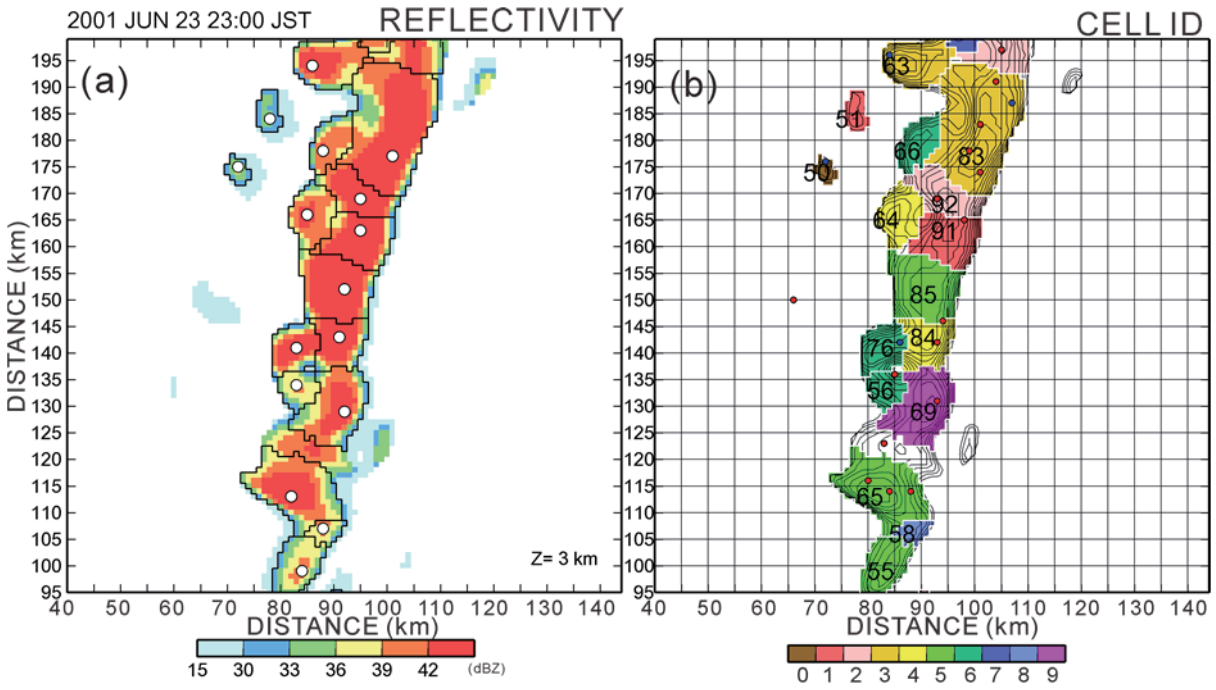


Fig. 10 Sample of Graphical Mapping Tools drawing. (a) Reflectivity with the cell boundary and location of "center of gravity" (open circle) at a height of 3 km. (b) CELL-ID (number) with updraft (red circle) and downdraft (blue circle) cores at a height of 3 km. Contours indicate reflectivity every 1 dBZ starting from 30 dBZ.

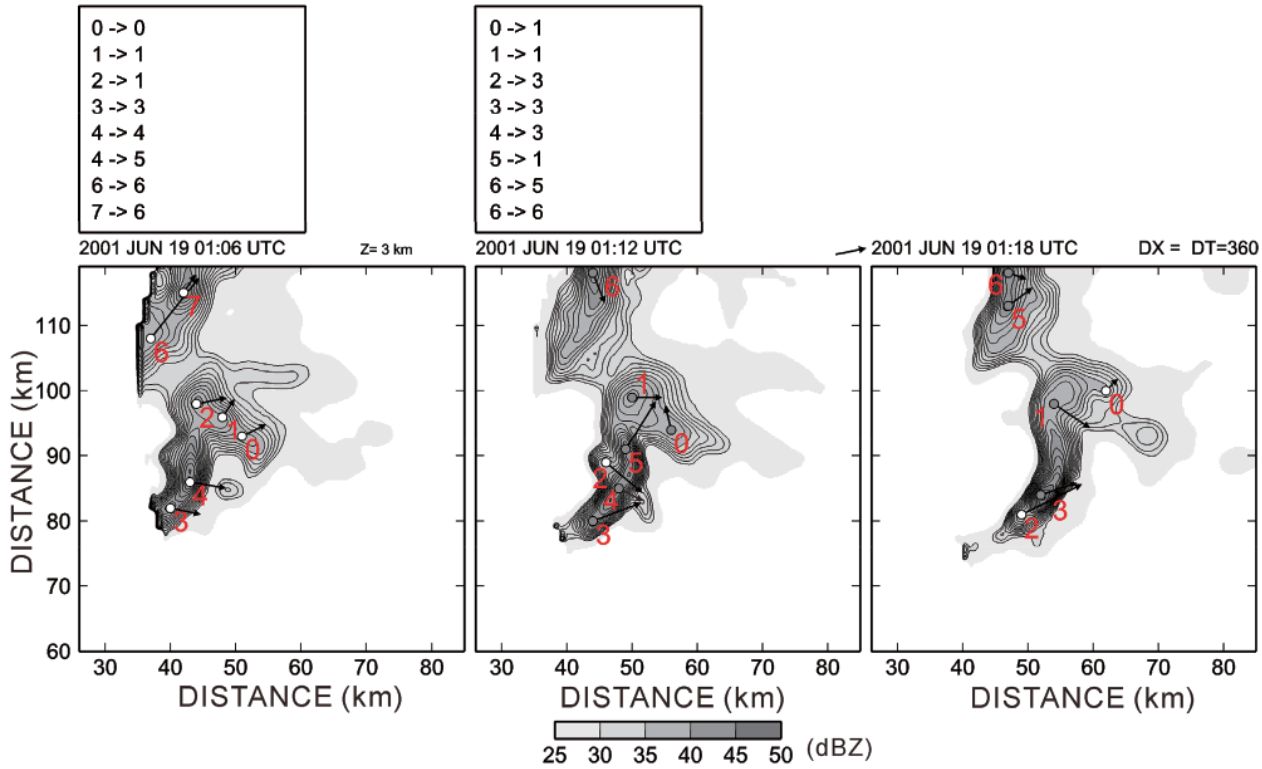


Fig. 11 Sample of cell tracking from 01:06 UTC to 01:18 UTC. Cell ID is shown by red digit number. Reflectivity is shown every 1 dBZ starting from 30 dBZ. Vector indicates the movement vector of each cell. Cell tracking result is shown in upper panel.

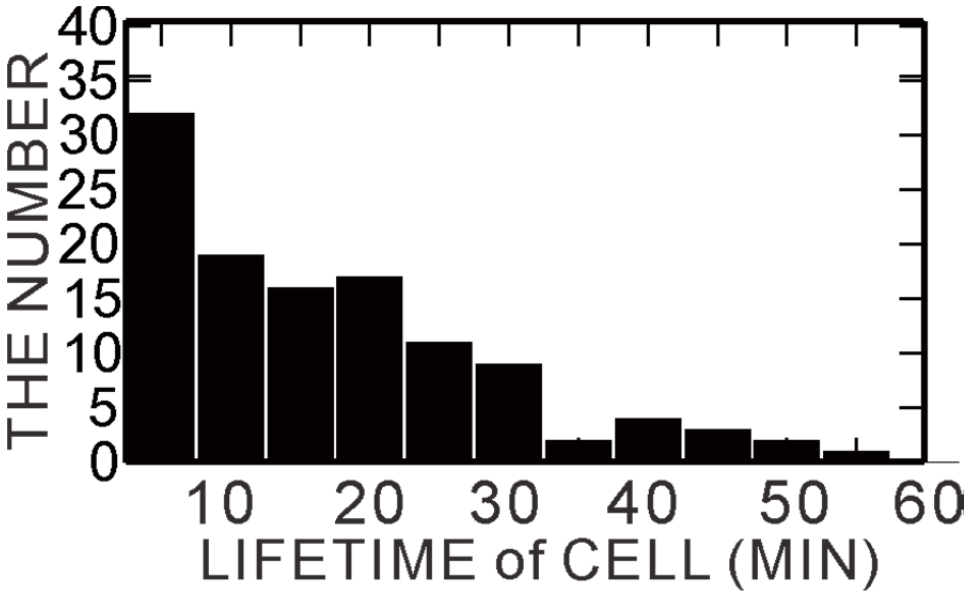


Fig. 12 Sample of statistical analysis of cell. Frequency distribution of lifetime of cell is shown.

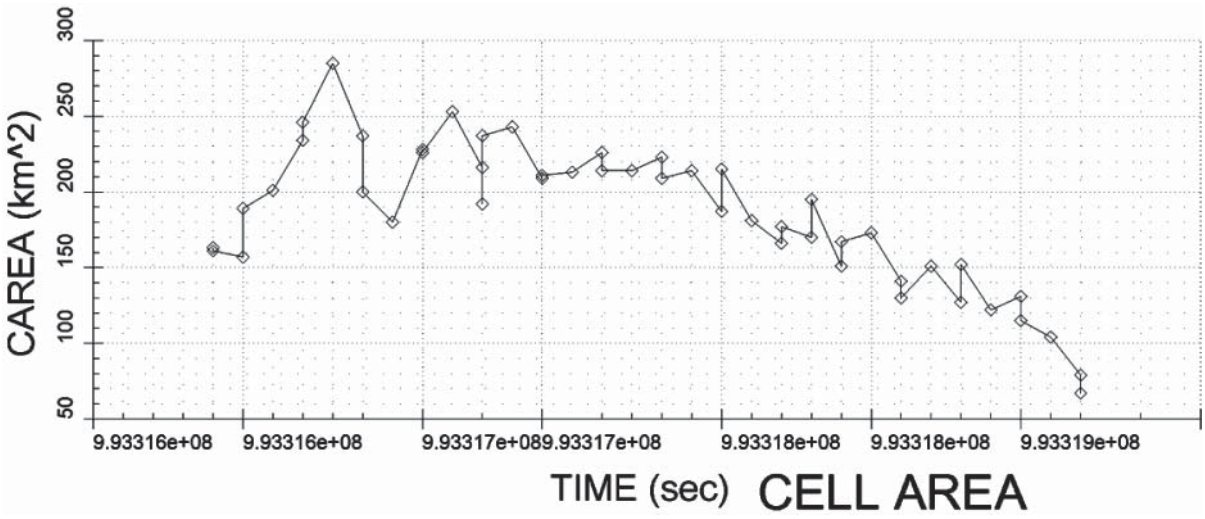


Fig. 13 Sample of cell history file. The time evolution of cell area is drawn by ncview software.

Table 13 List of perl subroutines for GMT drawing.

Subroutine name	Description
main.pl	Main driver to drawing with GMT.
option.pl	Set PATH of NetCDF data
TIME_DEF.pl	Read time information
CHECK_DIM.pl	Check dimension consistency
READ_Z.pl	Read height information
READ_REF.pl	Read reflectivity data
READ_W.pl	Read vertical air motion data
READ_EDGE.pl	Read grid classification data
READ_CCENX.pl	Read the x-location of "center of gravity" for CC
READ_CCENY.pl	Read the y-location of "center of gravity" for CC
READ_PEAKX.pl	Read the x-location of "reflectivity peak" for CC
READ_PEAKY.pl	Read the y-location of "reflectivity peak" for CC
READ_WCORE1.pl	Read updraft core location
READ_WCORE2.pl	Read downdraft core location
READ_ELPSX.pl	Read major radii of approximated ellipse for CC
READ_ELPSY.pl	Read minor radii of approximated ellipse for CC
READ_ELPST.pl	Read inclined angle of approximated ellipse for CC
Draw_incontious.pl	Draw the cell boundary
Add_corelocation.pl	Draw "CORE" location for CG
Add_centerlocation.pl	Draw "center of gravity" location for CC
Add_peaklocation.pl	Draw "peak" location for CC
Add_elpse.pl	Draw "approximated ellipse" around CC
Draw_cellnum.pl	Draw "CELL ID number" around CC

3. Summery

This document describes the detailed process of cell-tracking system developed by Shimizu and Uyeda, 2012 (Algorithm for Identification and Tracking of Convective Cell: AITCC), and basic operation of the AITCC, including how to run the program, the acceptable formats, and the procedure for drawing figure to show the AITCC results. The complete AITCC software package can be obtained from web, and is freely available for noncommercial use. It is requested that anyone using the software should acknowledge the work of the authors by citing this user guide as the official reference for AITCC

In a future upgrade (version 2.0), 3D analysis of convective cell and quality control using 2D-FFT for smoothing reflectivity field will be possible in FORTRAN code with shared memory parallel calculation. The AITCC code will be included in cloud-resolving numerical simulation to conduct statistical analysis for simulated-cell to validate simulation performance.

Acknowledgments

The author would like to express his special appreciation of Prof. Uyeda (Hydrospheric Atmospheric Research Center, Nagoya University) for his valuable suggestion to cell analysis. The author is grateful to Dr. T. Kato (Meteorological Research Institute) for his inspiring suggestions and fruitful discussions through revising JMSJ paper. All figures are made by Generic Mapping Tools (GMT) by university of Hawaii.

References

1) Chen, J., Uyeda, H., and D. -I. Lee (2003): A method using radar reflectivity data for the objective classification during the Baiu season. *J. Meteor. Soc. Japan*. **81**, 229-249.
2) Dixon, M., and Wiener, G. (1993): TITAN: Thunderstorm identification, tracking, analysis, and nowcasting -A radar based methodology. *J. Atmos. Oceanic Tech.*, **10**, 785-797.
3) Heinselman, P. L., and Torres, S. M. (2011): High-temporal-resolution capabilities of the national weather radar testbed phased-array radar. *J. Appl. Meteor. Climatol*. **50**, 579-593.
4) Shimizu, S., and Uyeda, H. (2012): Algorithm for the identification and tracking of convective cells based on constant and adaptive threshold methods using a new cell-merging and -splitting scheme. *J. Meteor. Soc. Japan*. **90**, 869-889.
5) Steiner, M., Houze, Jr. R. A., and Yuter, S. E. (1995): Climatological characterization of three-dimensional storm structure from operational radar and rain gauge data. *J. Appl. Meteor*. **34**, 1978-2007.
6) Yoshida, S., Misumi, R., Shimizu, S., Maesaka, T., Iwanami, K., and Maki, M. (2013): Validation of short-term forecasting of meso--scale convective systems based on a cell-tracking system, *SOLA*, **8**, 141-144.

(Accepted: December 24, 2013)

Appendix A Input file format

Fig. A indicates the sample of ncdump command for NetCDF input file required by AITCC. At least, dimension name, mandatory parameter name, and units of mandatory parameter must be coincided with this sample.

```
netcdf file name {
  dimensions:
        TIME = 1 ;   dimension name of time
        LEV = 61 ;   dimension name of altitude
        LAT = 201 ;  dimension name of latitude
        LON = 201 ;  dimension name of longitude
         ITR = 42 ;
 variables:
        double TIME(TIME) ;   Double precision is required. Unit must be seconds (UNIX TIME).
                TIME:long_name = “unix time” ;
                TIME:units = “s” ;
        double LEV(LEV) ;     Double precision is required. Unit must be meter.
                LEV:long_name = “height” ;
                LEV:units = “m” ;
        double LAT(LAT) ;
                LAT:long_name = “Latitude” ;  Double precision is required. Unit must be degree.
                LAT:units = “degree_N “ ;
        double LON(LON) ;
                LON:long_name = “Longitude” ;  Double precision is required. Unit must be degree.
                LON:units = “degree_E “ ;
        float REF(TIME, LEV, LAT, LON) ;   Single precision is required. Unit must be dBZ.
                REF:long_name = “reflectivity” ;
                REF:units = “dBZ” ;
                REF:missing_value = -999.9f ;
        float W(TIME, LEV, LAT, LON) ;   Single precision is required. Unit must be meter per second.
                W:long_name = “z components of velocity” ;
                W:units = “m/s” ;
                W:missing_value = -999.9f ;
```

Fig. A Sample of result of ncdump command for input file. Mandatory parameters are highlighted by yellow. The parameters required for “W_OPT” are highlighted by light-blue.

Appendix B Output file format of data grid object

```
netcdf file name {
dimensions:
        TIME = UNLIMITED ; // (1 currently)
        Z = 1 ;
        ZA = 61 ;
        Y = 201 ;
        X = 201 ;
        MNUM = 5 ;
        CNUM = 15 ;
        CONNECT = 20 ;
variables:
        double TIME(TIME) ;
                TIME:long_name = “UNIX TIME(from 1970/1/1 00:00Z)” ;
                TIME:units = “sec” ;
        int CHECK ;
                CHECK:long_name = “1:MCS exist 0: No MCS” ;
                CHECK:units = “” ;
                CHECK:missing_value = -999 ;
        int CHECK2 ;
                CHECK2:long_name = “1:CELL exist 0: No CELL” ;
                CHECK2:units = “” ;
                CHECK2:missing_value = -999 ;
        double Z(Z) ;
                Z:long_name = “GRID NUMBER in Z(SELECT)” ;
                Z:units = “num” ;
        double ZA(ZA) ;
                ZA:long_name = “GRID NUMBER in Z(ALL)” ;
                ZA:units = “num” ;
                ZA:missing_value = -999. ;
        double Y(Y) ;
                Y:long_name = “GRID NUMBER in Y” ;
                Y:units = “num” ;
        double X(X) ;
                X:long_name = “GRID NUMBER in X” ;
                X:units = “num” ;
        short MNUM(MNUM) ;
                MNUM:long_name = “the NUMBER of MCS at standard height” ;
                MNUM:units = “num” ;
                MNUM:missing_value = -999s ;
        short CNUM(CNUM) ;
                CNUM:long_name = “the NUMBER of CELL at standard height” ;
                CNUM:units = “num” ;
                CNUM:missing_value = -999s ;
        float RAWREF(Z, Y, X) ;
                RAWREF:long_name = “Raw Reflectivity” ;
                RAWREF:units = “dBZe” ;
                RAWREF:missing_value = -999.9f ;
        short REF(Z, Y, X) ;
                REF:long_name = “Reflectivity” ;
                REF:units = “dBZe” ;
```

REF:missing_value = -999s ;
REF:add_offset = 0.f ;
REF:scale_factor = 0.1f ;
short W(Z, Y, X) ;
W:long_name = “Vertical Velicity” ;
W:units = “m/s” ;
W:missing_value = -999s ;
W:add_offset = 0.f ;
W:scale_factor = 0.01f ;
short STN(Y, X) ;
STN:long_name = “Steiner INDEX” ;
STN:units = “non” ;
STN:missing_value = -999s ;
short EDGE(Z, Y, X) ;
EDGE:long_name = “MCS EDGE(0:out,1;edge,2:inner,3:core)” ;
EDGE:units = “non” ;
EDGE:missing_value = -999s ;
EDGE:OUTER = 0s ;
EDGE:EDGE = 1s ;
EDGE:INNER = 2s ;
EDGE:CORE = 3s ;
EDGE:REJEC = 4s ;
short NAME(Z, Y, X) ;
NAME:long_name = “MCS NAME” ;
NAME:units = “num” ;
NAME:missing_value = -999s ;
short CNAME(Z, Y, X) ;
CNAME:long_name = “CELL NAME” ;
CNAME:units = “num” ;
CNAME:missing_value = -999s ;
short CWCORE1(Z, Y, X) ;
CWCORE1:long_name = “CELL UP CORE” ;
CWCORE1:units = “num” ;
CWCORE1:missing_value = -999s ;
short CWCORE2(Z, Y, X) ;
CWCORE2:long_name = “CELL DW CORE” ;
CWCORE2:units = “num” ;
CWCORE2:missing_value = -999s ;
short MAREA(MNUM) ;
MAREA:long_name = “MCS AREA” ;
MAREA:units = “km^2” ;
MAREA:missing_value = -999s ;
short MTYPE(MNUM) ;
MTYPE:long_name = “MCS TYPE(s:single, m:multi,n:nodata)” ;
MTYPE:units = “type” ;
MTYPE:missing_value = -999s ;
MTYPE:single = 1s ;
MTYPE:multi = 2s ;
MTYPE:no_core = 0s ;
short MPEAK(MNUM) ;
MPEAK:long_name = “The number of peak in a MCS” ;

MPEAK:units = “num” ;
MPEAK:missing_value = -999s ;
short MCENTX(MNUM) ;
MCENTX:long_name = “Center X-location of a MCS” ;
MCENTX:units = “grid” ;
MCENTX:missing_value = -999s ;
short MCENTY(MNUM) ;
MCENTY:long_name = “Center Y-location of a MCS” ;
MCENTY:units = “grid” ;
MCENTY:missing_value = -999s ;
short MMAXX(MNUM) ;
MMAXX:long_name = “X-location of Max Ref in MCS” ;
MMAXX:units = “grid” ;
MMAXX:missing_value = -999s ;
short MMAXY(MNUM) ;
MMAXY:long_name = “Y-location of Max Ref in MCS” ;
MMAXY:units = “grid” ;
MMAXY:missing_value = -999s ;
float MHEGT(MNUM) ;
MHEGT:long_name = “The maximum height of echo top in a MCS” ;
MHEGT:units = “km” ;
MHEGT:missing_value = -999.f ;
short MVCRR(Z, MNUM) ;
MVCRR:long_name = “number of vertical correlated MCS” ;
MVCRR:units = “num” ;
MVCRR:missing_value = -999s ;
short MVELX(MNUM) ;
MVELX:long_name = “x-componet of velocity of MCS” ;
MVELX:units = “m s-1” ;
MVELX:missing_value = -999s ;
MVELX:INITUV = -999s ;
short MVELY(MNUM) ;
MVELY:long_name = “y-componet of velocity of MCS” ;
MVELY:units = “m s-1” ;
MVELY:missing_value = -999s ;
MVELY:INITUV = -999s ;
short MREFX ;
MREFX:long_name = “x-component of Reference velocity” ;
MREFX:units = “m/s” ;
MREFX:missing_value = -999s ;
MREFX:INITUV = -999s ;
short MREFY ;
MREFY:long_name = “y-component of Reference velocity” ;
MREFY:units = “m/s” ;
MREFY:missing_value = -999s ;
MREFY:INITUV = -999s ;
short MPSTN(MNUM) ;
MPSTN:long_name = “The number of post link” ;
MPSTN:units = “num” ;
MPSTN:missing_value = -999s ;
short MPREN(MNUM) ;
MPREN:long_name = “The number of previous link” ;

MPREN:units = “num” ;
MPREN:missing_value = -999s ;
short MPST(MNUM, CONNECT) ;
MPST:long_name = “The number of post link” ;
MPST:units = “num” ;
MPST:missing_value = -999s ;
short MPRE(MNUM, CONNECT) ;
MPRE:long_name = “The number of previous link” ;
MPRE:units = “num” ;
MPRE:missing_value = -999s ;
short MSTUS(MNUM) ;
MSTUS:long_name = “Life stage of MCS” ;
MSTUS:units = “status” ;
MSTUS:missing_value = -999s ;
MSTUS:INI = 0s ;
MSTUS:SRT = 1s ;
MSTUS:GEN = 2s ;
MSTUS:DIS = 3s ;
MSTUS:MAN = 4s ;
MSTUS:END = 5s ;
short MMEAN(MNUM) ;
MMEAN:long_name = “Mean Reflectivity” ;
MMEAN:units = “dBZ” ;
MMEAN:missing_value = -999s ;
MMEAN:add_offset = 0.f ;
MMEAN:scale_factor = 0.1f ;
short MMAX(MNUM) ;
MMAX:long_name = “Max Reflectivity” ;
MMAX:units = “dBZ” ;
MMAX:missing_value = -999s ;
MMAX:add_offset = 0.f ;
MMAX:scale_factor = 0.1f ;
float MELPSX(MNUM) ;
MELPSX:long_name = “Coeficient of Ellipse in x direction of MCS” ;
MELPSX:units = “” ;
MELPSX:missing_value = -999.f ;
float MELPSY(MNUM) ;
MELPSY:long_name = “Coeficient of Ellipse in y direction of MCS” ;
MELPSY:units = “” ;
MELPSY:missing_value = -999.f ;
float MELPST(MNUM) ;
MELPST:long_name = “Ellipse inclining angle of MCS” ;
MELPST:units = “” ;
MELPST:missing_value = -999.f ;
short CELLN(MNUM) ;
CELLN:long_name = “The number of CELL in MCS” ;
CELLN:units = “num” ;
CELLN:missing_value = -999s ;
short MBEC(MNUM, CNUM) ;
MBEC:long_name = “CELL number belogs to MCS” ;
MBEC:units = “num” ;
MBEC:missing_value = -999s ;

```
short CAREA(CNUM) ;
        CAREA:long_name = “CELL AREA” ;
        CAREA:units = “km^2” ;
        CAREA:missing_value = -999s ;
short CBEM(CNUM) ;
        CBEM:long_name = “the MCS number CELL belogs to” ;
        CBEM:units = “num” ;
        CBEM:missing_value = -999s ;
short CPEAX(CNUM) ;
        CPEAX:long_name = “x-grid where CELL\’s peak” ;
        CPEAX:units = “grid” ;
        CPEAX:missing_value = -999s ;
short CPEAY(CNUM) ;
        CPEAY:long_name = “y-grid where CELL\’s peak” ;
        CPEAY:units = “grid” ;
        CPEAY:missing_value = -999s ;
short CCENX(CNUM) ;
        CCENX:long_name = “x-grid where CELL\’s gravity center” ;
        CCENX:units = “grid” ;
        CCENX:missing_value = -999s ;
short CCENY(CNUM) ;
        CCENY:long_name = “y-grid where CELL\’s gravity center” ;
        CCENY:units = “grid” ;
        CCENY:missing_value = -999s ;
float CMEAN(CNUM) ;
        CMEAN:long_name = “Mean reflectivity in CELL” ;
        CMEAN:units = “dBZ” ;
        CMEAN:missing_value = -999.f ;
        CMEAN:add_offset = 0.f ;
        CMEAN:scale_factor = 0.1f ;
float CMAXR(CNUM) ;
        CMAXR:long_name = “Mean reflectivity in CELL” ;
        CMAXR:units = “dBZ” ;
        CMAXR:missing_value = -999.f ;
        CMAXR:add_offset = 0.f ;
        CMAXR:scale_factor = 0.1f ;
float CHEGT(CNUM) ;
        CHEGT:long_name = “echo height in a CELL(15)” ;
        CHEGT:units = “km” ;
        CHEGT:missing_value = -999.f ;
float CHEGT2(CNUM) ;
        CHEGT2:long_name = “echo height in a CELL(30)” ;
        CHEGT2:units = “km” ;
        CHEGT2:missing_value = -999.f ;
float CRCOH(CNUM) ;
        CRCOH:long_name = “height of reflectivity core in a CELL” ;
        CRCOH:units = “km” ;
        CRCOH:missing_value = -999.f ;
float CRCOR(CNUM) ;
        CRCOR:long_name = “Reflectivity of reflectivity core height” ;
        CRCOR:units = “dBZ” ;
        CRCOR:missing_value = -999.f ;
```

float CRPRO(CNUM, ZA) ;
CRPRO:long_name = “Vertical profile of reflectivity” ;
CRPRO:units = “dBZe” ;
CRPRO:missing_value = -999.f ;
short CVELX(CNUM) ;
CVELX:long_name = “x-component of velocity of CELL” ;
CVELX:units = “m/s” ;
CVELX:missing_value = -999s ;
short CVELY(CNUM) ;
CVELY:long_name = “y-component of velocity of CELL” ;
CVELY:units = “m/s” ;
CVELY:missing_value = -999s ;
short CSTUS(CNUM) ;
CSTUS:long_name = “Life stage of CELL” ;
CSTUS:units = “status” ;
CSTUS:missing_value = -999s ;
CSTUS:INI = 0s ;
CSTUS:SRT = 1s ;
CSTUS:GEN = 2s ;
CSTUS:DIS = 3s ;
CSTUS:MAN = 4s ;
CSTUS:END = 5s ;
short CPSTN(CNUM) ;
CPSTN:long_name = “The number of post link” ;
CPSTN:units = “num” ;
CPSTN:missing_value = -999s ;
short CPREN(CNUM) ;
CPREN:long_name = “The number of previous link” ;
CPREN:units = “num” ;
CPREN:missing_value = -999s ;
short CPST(CNUM, CONNECT) ;
CPST:long_name = “The number of post link” ;
CPST:units = “num” ;
CPST:missing_value = -999s ;
short CPRE(CNUM, CONNECT) ;
CPRE:long_name = “The number of previous link” ;
CPRE:units = “num” ;
CPRE:missing_value = -999s ;
short CVCRR(Z, CNUM) ;
CVCRR:long_name = “number of vertical correlated CELL” ;
CVCRR:units = “num” ;
CVCRR:missing_value = -999s ;
short CMUP(CNUM) ;
CMUP:long_name = “Mean updraft in CELL” ;
CMUP:units = “m/s” ;
CMUP:missing_value = -999s ;
CMUP:add_offset = 0.f ;
CMUP:scale_factor = 0.01f ;
short CMDW(CNUM) ;
CMDW:long_name = “Mean downdraft in CELL” ;
CMDW:units = “m/s” ;
CMDW:missing_value = -999s ;

```
                CMDW:add_offset = 0.f ;
                CMDW:scale_factor = 0.01f ;
        short CMXUP(CNUM) ;
                CMXUP:long_name = “Maximum updraft in CELL” ;
                CMXUP:units = “m/s” ;
                CMXUP:missing_value = -999s ;
                CMXUP:add_offset = 0.f ;
                CMXUP:scale_factor = 0.01f ;
        short CMXDW(CNUM) ;
                CMXDW:long_name = “Maximum downdraft in CELL” ;
                CMXDW:units = “m/s” ;
                CMXDW:missing_value = -999s ;
                CMXDW:add_offset = 0.f ;
                CMXDW:scale_factor = 0.01f ;
        short CAREART(CNUM) ;
                CAREART:long_name = “ratio of updraft and downdraft in CELL” ;
                CAREART:units = “non” ;
                CAREART:missing_value = -999s ;
                CAREART:add_offset = 0.f ;
                CAREART:scale_factor = 0.001f ;
        short W_PEAK1(CNUM) ;
                W_PEAK1:long_name = “The number of updraft core in CELL” ;
                W_PEAK1:units = “num” ;
                W_PEAK1:missing_value = -999s ;
        short W_PEAK2(CNUM) ;
                W_PEAK2:long_name = “The number of downdraft core in CELL” ;
                W_PEAK2:units = “num” ;
                W_PEAK2:missing_value = -999s ;
        float ELPSX(CNUM) ;
                ELPSX:long_name = “Coeficient of Ellipse in x direction” ;
                ELPSX:units = “” ;
                ELPSX:missing_value = -999.f ;
        float ELPSY(CNUM) ;
                ELPSY:long_name = “Coeficient of Ellipse in y direction” ;
                ELPSY:units = “” ;
                ELPSY:missing_value = -999.f ;
        float ELPST(CNUM) ;
                ELPST:long_name = “Ellipse inclining angle” ;
                ELPST:units = “” ;
                ELPST:missing_value = -999.f ;

// global attributes:
                :history = “DATA Created:201312191305” ;
                :CUT_REF = 10.f ;
                :TRE_MCS = 300s ;
                :MIN_AREA = 10s ;
                :MIN_DIS_CORE = 2s ;
                :ECHOTOP_DBZ = 15s ;
                :ECHOTOP_DBZ2 = 30s ;
                :DX = 1.f ;
                :DY = 1.f ;
                :DZ = 0.5f ;
```

Appendix C Output file format of history file for individual cell/cell group

netcdf file name
dimensions:
TIME = UNLIMITED ; // (48 currently)
CONNECT = 20 ;
variables:
short CLIFE ;
CLIFE:long_name = “CELL LIFE TIME” ;
CLIFE:units = “min” ;
CLIFE:missing_value = -999s ;
double TIME(TIME) ;
TIME:long_name = “UNIX TIME(from 1970/1/1 00:00Z)” ;
TIME:units = “sec” ;
short NUM(TIME) ;
NUM:long_name = “CELL TRACE NUM” ;
NUM:units = “num” ;
NUM:missing_value = -999s ;
short CAREA(TIME) ;
CAREA:long_name = “CELL TRACE AREA” ;
CAREA:units = “km^2” ;
CAREA:missing_value = -999s ;
short CCENX(TIME) ;
CCENX:long_name = “CELL TRACE CENTER X” ;
CCENX:units = “grid” ;
CCENX:missing_value = -999s ;
short CCENY(TIME) ;
CCENY:long_name = “CELL TRACE CENTER Y” ;
CCENY:units = “grid” ;
CCENY:missing_value = -999s ;
short CMEAN(TIME) ;
CMEAN:long_name = “CELL TRACE MEAN AREA” ;
CMEAN:units = “dBZ” ;
CMEAN:missing_value = -999s ;
CMEAN:add_offset = 0.f ;
CMEAN:scale_factor = 0.1f ;
short CMAXR(TIME) ;
CMAXR:long_name = “CELL TRACE MAX AREA” ;
CMAXR:units = “dBZ” ;
CMAXR:missing_value = -999s ;
CMAXR:add_offset = 0.f ;
CMAXR:scale_factor = 0.1f ;
short CPSTN(TIME) ;
CPSTN:long_name = “CELL TRACE PST NUMBER” ;
CPSTN:units = “num” ;
CPSTN:missing_value = -999s ;
short CPREN(TIME) ;
CPREN:long_name = “CELL TRACE PRE NUMBER” ;
CPREN:units = “num” ;
CPREN:missing_value = -999s ;
float CHEGT(TIME) ;
CHEGT:long_name = “CELL TRACE HEIGHT” ;

```
        CHEGT:units = “km” ;
        CHEGT:missing_value = -999.f ;
short CVELX(TIME) ;
        CVELX:long_name = “CELL TRACE x-comp of VEL” ;
        CVELX:units = “m/s” ;
        CVELX:missing_value = -999s ;
short CVELY(TIME) ;
        CVELY:long_name = “CELL TRACE y-comp of VEL” ;
        CVELY:units = “m/s” ;
        CVELY:missing_value = -999s ;
short PST_order(TIME, CONNECT) ;
        PST_order:long_name = “order of split” ;
        PST_order:units = “non” ;
        PST_order:missing_value = -999s ;
short PRE_order(TIME, CONNECT) ;
        PRE_order:long_name = “order of merge” ;
        PRE_order:units = “non” ;
        PRE_order:missing_value = -999s ;
short CMUP(TIME) ;
        CMUP:long_name = “Mean updraft in CELL” ;
        CMUP:units = “m/s” ;
        CMUP:missing_value = -999s ;
        CMUP:add_offset = 0.f ;
        CMUP:scale_factor = 0.01f ;
short CMDW(TIME) ;
        CMDW:long_name = “Mean downdraft in CELL” ;
        CMDW:units = “m/s” ;
        CMDW:missing_value = -999s ;
        CMDW:add_offset = 0.f ;
        CMDW:scale_factor = 0.01f ;
short CMXUP(TIME) ;
        CMXUP:long_name = “Maximum updraft in CELL” ;
        CMXUP:units = “m/s” ;
        CMXUP:missing_value = -999s ;
        CMXUP:add_offset = 0.f ;
        CMXUP:scale_factor = 0.01f ;
short CMXDW(TIME) ;
        CMXDW:long_name = “Maximum downdraft in CELL” ;
        CMXDW:units = “m/s” ;
        CMXDW:missing_value = -999s ;
        CMXDW:add_offset = 0.f ;
        CMXDW:scale_factor = 0.01f ;
short CAREART(TIME) ;
        CAREART:long_name = “ratio of updraft and downdraft in CELL” ;
        CAREART:units = “non” ;
        CAREART:missing_value = -999s ;
        CAREART:add_offset = 0.f ;
        CAREART:scale_factor = 0.001f ;
short W_PEAK1(TIME) ;
        W_PEAK1:long_name = “The number of updraft core in CELL” ;
        W_PEAK1:units = “num” ;
```

```
                W_PEAK1:missing_value = -999s ;
        short W_PEAK2(TIME) ;
                W_PEAK2:long_name = “The number of downdraft core in CELL” ;
                W_PEAK2:units = “num” ;
                W_PEAK2:missing_value = -999s ;
// global attributes:
                :history = “DATA Created:201101211815” ;
```

AITCC ユーザーガイド

― 自動対流セル検出・追跡アルゴリズム ―

清水 慎吾*

*防災科学技術研究所 水・土砂防災研究ユニット

要　旨

AITCC (**A**lgorithm for **I**dentification and **T**racking of **C**onvective **C**ell; エイティック) はレーダーデータを用いた対流セルの統計解析や短時間ナウキャストを目的とした自動対流セル検出・追跡アルゴリズムである．AITCC のプログラムは，非商用目的として防災科学技術研究所のホームページで公開されている (http://mizu.bosai.go.jp/wiki/wiki.cgi?=AITCC_HOME)．本研究資料は，AITCC の基本的な利用方法，入出力データフォーマット，対流セル解析結果の表示方法について記載している．AITCC は，過去の観測計画などで，膨大に保存されているレーダーデータを統計解析として有効に利用できることや，今後の高時間分解能レーダによる積乱雲の三次元解析やその短時間予測などにも適用でき，幅広い利用が期待できる．

キーワード：対流セル，ナウキャスティング，積乱雲の統計解析