Development of an Object Oriented Program for Traverse Computation

Odumosu, J. O (1); Ajayi, O. G (2); Ibrahim. P(3); Okorocha, V. C (4); Idowu. F. F(5)
1,2, 3 Department of Surveying and Geoinformatics, Federal University of Technology, Minna, Nigeria.
4Department of Surveying and Geoinformatics, University of Lagos, Lagos, Nigeria
5Department of Surveying and Geoinformatics, The Polytechnic Ibadan, Nigeria
Corresponding Author: odumossu4life@yahoo.com

Abstract: Determining the location of points and orientation of lines frequently depend on measurement of angles and directions in surveying operations. Survey computations are often the most tedious aspect of many surveying operations. The conventional method of achieving point positioning is dreary and cumbersome. In order to ease this process and to avoid various computational errors, this research presents an Object Oriented Program for the complete automation of the Traverse survey adjustment/computation process. The transit method of traverse adjustment was adopted using Visual Basic Language. The design and description of the program was vividly explained bearing in mind the theory of approximation so as to minimize round off errors. Finally, the completed program was packaged into a user-friendly Graphical User Interphase and an easily executable/installable software-file format using the Visual Basic 6.0 Installer.

Keywords: Object Oriented Programming, Traverse Surveying, Computation/Adjustment, Visual Basic, Point Positioning.

1.0 INTRODUCTION

Technology is rapidly changing every facet of human life. Satellite based systems and computer technology are ushering in integrated work environment. The necessary input for surveying in the 21st century is becoming fully digital, and as such depends on satellite and computer technology (Ndukwe, 2001). One major task of the surveyor in virtually all fields of human endeavor (ranging from boundary demarcation, route profiling, oil well fixing, etc) is the determination of the planimetric co-ordinates of points on the earth's surface. This task of co-ordinate determination forms the basis on which all subsequent surveys and engineering decisions rely. Therefore, in order to obtain the final co-ordinates of points on the earth surface, a traverse which is a series of connected straight lines, each of which whose distance and the angle connecting it to the next line is properly and accurately measured. The conventional method of achieving these co-ordinates is very tedious and cumbersome at every of the stages involved. Considering the field operation, followed by the stress of computation and finally the rigours of adjustment - the entire process could be laborious and time consuming.

However, recent developments in electronics and computer capability, digital technology and survey instrumentation have improved surveying and mapping in a way that has never been before. These instruments have changed the ways in which topographic and other land related information are acquired, stored and managed. As a result, the conventional and analogue instruments are quickly sliding into the archives while the classical methods of data acquisition are giving way to new and modern methods. (Ndukwe, 2001). The process of traverse computation is generally known to be a long, time consuming and iterative (thus making it error prone) process. In addition to the several challenges posed by this process, position determination remains a prerequisite for virtually all surveying tasks. It is therefore necessary to automate this process in-order to enhance the surveyor's job as well as to give more integrity to the results obtained.

Developments in computer and satellite technologies, image processing, Database management, navigation and position fixing systems etc. have changed the tools available to the surveyor. These new integrating tools and techniques have been complementing or replacing established surveying techniques and geoinformation production process (Beek et al, 1996). According to (Ndukwe, 2001) one of the most significant developments in computing technology in the last decade was the advent of the personal computer (commonly called PC). Computer programming is the iterative process of writing or editing source code. Editing source code involves testing, analyzing, and refining, and sometimes coordinating with other programmers on a jointly developed program. A comprehensive package on surveying computations was done by (Barnes, 1977). (Adejare, 2003) also wrote a similar algorithm using Microsoft Excel Spread Sheet.

This research therefore seeks to develop object oriented program for traverse computation and plotting by developing an Object Oriented Program that will perform all the sub-computations in a traverse computational process and generating a form, which can be printed out and kept as record of traverse computation, and to compute the area of the traverse. Visual basic (VB 6.0) software was adopted due to its object oriented nature and graphic user interface capacity (User friendly and interactive forms). The program shall link with Microsoft Access where database was stored before the final display in Visual Basic environment.

2.0 CONCEPT OF OBJECT ORIENTED PROGRAMMING

Object-oriented programming (**OOP**) is a programming paradigm that uses "objects" – data structures consisting of data fields and methods together with their interactions – to design applications and computer programs. Programming techniques may include features such as data abstraction, encapsulation, modularity, polymorphism, and inheritance. These features are briefly described below:

DATA ABSTRACTION: Abstraction is simplifying complex reality by modeling classes appropriate to the problem, and working at the most appropriate level of inheritance for a given aspect of the problem. For example, Lassie the Dog may be treated as a Dog much of the time, a Collie when necessary to access Collie-specific attributes or behaviors, and as an Animal (perhaps the parent class of Dog) when counting Timmy's pets. (Armstrong, 2006; John, 2003)

ENCAPSULATION: Encapsulation conceals the functional details of a class from objects that send messages to it. For example, the Dog class has a bark() method variable data. The code for the bark() method defines exactly how a bark happens (e.g., by inhale() and then exhale(), at a particular pitch and volume). The reason for encapsulation is to prevent clients of an interface from depending on those parts of the implementation that are likely to change in the future, thereby allowing those changes to be made more easily, that is, without changes to clients. (Armstrong, 2006; John, 2003)

POLYMORPHISM: Polymorphism allows the programmer to treat derived class members just like their parent class' members. More precisely, Polymorphism in object-oriented programming is the ability of objects belonging to different data types to respond to calls of methods of the same name, each one according to an appropriate type-specific behavior. One method, or an operator such as +, -, or *, can be abstractly applied in many different situations. If a Dog is commanded to speak(), this may elicit a bark.(Armstrong, 2006)

INHERITANCE: Inheritance is a process in which a class inherits all the state and behavior of another class. This type of relationship is called Child-Parent or is-a relationship. "Subclasses" are more specialized versions of a class, which *inherit* attributes and behaviors from their parent classes, and can introduce their own. For example, the class Dog might have sub-classes called *Collie*, *Chihuahua*, and *GoldenRetriever*. In this case, *Lassie* would be an instance of the *Collie* subclass. Suppose the Dog class defines a method called bark() and a property called furColor. Each of its sub-classes (*Collie, Chihuahua, and GoldenRetriever*) will inherit these members, meaning that the programmer only needs to write the code for them once. (Armstrong, 2006)

3.0 METHODOLOGY IN VISUAL BASIC PROGRAMMING

The following step-by-step process (**program development cycle**) was adopted to design the error-free programs that produce the desired output.

Analyze: This defines the problem statement and provides a clear idea of what data (or input) are given and the relationship between the input and the desired output. (Schneider, 1995)

Design: Plan the solution to the problem. Find a logical sequence of precise steps that solve the problem. Such a sequence of steps is called an **algorithm**. Planning also involves using representative data to test the logic of the algorithm by hand to ensure that it is correct. At this stage, decision variables are decided.

Choose the interface: Select the objects (text boxes, command buttons, etc.). Determine how the input will be obtained and how the output will be displayed. Then create objects to receive the input and display the output. Also, create appropriate command buttons to allow the user to control the program. The designer will reason along on how the various parameters are to be entered in to the system in a system recognisable format that will satisfy the purpose and still provide the required result correctly.

Code: Translate the algorithm into a programming language. **Coding** is the technical word for writing the program. During this stage, the program is written in Visual Basic and entered into the computer. The programmer uses the algorithm devised in Design stage along with knowledge of Visual Basic. Appropriate codes are used to represent the various mathematical and logical formulae involved in the task. At this stage, necessary precautions were taken to ensure that negative conditions are properly taken care of and divisions by zero are avoided.

Test and debug: Locate and remove any errors in the program. **Testing** is the process of finding errors in a program, and **debugging** is the process of correcting errors that are found.

Complete the documentation: Organize all the material that describes the program. Documentation is intended to allow another person or the programmer at a later date, to understand the program. Internal documentation consists of statements in the program that are not executed, but point out the purposes of various parts of the program.

3.1 TRAVERSING METHODS

Traverse network consists of straight lines of connected points whose distances and bearings (directions) have been determined from measurements carried out in the field. Given the co-ordinates of the first station and the bearing of the first line, the co-ordinates of all successive points can be calculated.

3.2 TRAVERSING REDUCTION

The reduction starts from at least three known positions (reference points). It is important that after the distance has been measured on the field, the measured distances are reduced. This reduction must be done using appropriate formulae (Slope correction). The formula used is as shown in equation 1.0 below:

$$L = S(1 - Cos\theta) \tag{1.0}$$

Where $L = \text{the measured slope distance}$

$$\theta = Observed\ vertical\ angle$$
$$S = measured\ distance$$

Thereafter, the bearing of each line is deduced. In traversing, two steps are taken during the angular measurement in order to facilitate the calculation of the required bearings and to reduce the possibility of observation and calculation errors. These steps are:

1. The back sight is taken first before observing the foresight. This is to prevent errors arising from deciding on which angle was actually measured.
2. Angles are measured on both faces of the theodolite and the results are recorded (booked). This eliminates instrumental errors and provides two measures of angles; this checks against a gross error in either of the two sets of measurement.
3. When the traverse forms a totally closed circuit, the internal angles, in number should sum up to $(2n - 4)$ right angles. Alternatively, the external angle should sum $(2n + 4)$ right angles.

Provided that the angular misclosure is within the specific tolerance, the preliminary angles for each traverse leg could be computed using the initial datum bearing and the observed traverse angle.

3.2.1 TRANSIT METHOD

Below is a sequential list of the operations involved and their respective formulae:

1. **Angular Reduction**: This traverse field book will first be reduced as follows:

$$Angle\ A = L2 - L1$$

$$Angle\ B = R3 - R4$$

These were repeated for all other stations. The mean angles C were summed together and checked for misclosure using the formular; $(2n + 4)90$ for exterior angles where n is the number of station in a close loop traverse. The maximum permissible error is $\pm\sqrt{30''}$ n for angular misclosure where: n = Number of stations.

2. **Back Computation Of Controls**:

$$N2 - N1 = \Delta N$$

$$E2 - E1 = \Delta E$$

$$Dist. = \sqrt{\Delta E^2 + \Delta N^2}$$

$$Bearing = Tan^{-1}\frac{\Delta E}{\Delta N} \quad 1.4$$

3. **Bearing Reduction**: This is done by using the formula below:

$$Back\ Bearing + Observed\ Angle = Forward\ Bearing.$$

4. **Correction to Bearing**: This is obtained using the formula below

$$Computed\ Forward\ Bearing - Derived\ Forward\ Bearing = Correction\ to\ Bearing$$

5. **Distance Correction**: Horizontal distances were computed by applying the slope correction. This is achieved using the formula given in Equation 1:
6. **Forward Computation**:

$$Nb = Na + L\cos\theta \quad 1.5$$

$$Eb = Ea + L\sin\theta \quad 1.6$$

Where Na and Ea are the nothings and easting coordinates of the known point A

Nb and Eb are the northing and easting coordinates of the known points B

3.3 DESIGN AND IMPLEMENTATION

Let us now take a careful look at all the various steps involved in the design and subsequently the implementation of this program

3.3.1 Flow Charts: A flowchart is a diagrammatic representation of the steps a program must follow in-order to obtain a required result. Below are the flow charts designed in the course of this project. A flowchart consists of special geometric symbols connected by arrows. Within each symbol is a phrase presenting the activity at that step. The shape of the symbol indicates the type of operation that is to be performed. For instance, the parallelogram denotes input or output. The arrows connecting the symbols, called flowlines, show the progression in which the steps take place. Flowcharts show "flow" from the top of the page to the bottom. Although the symbols used in flowcharts are standardized, no standards exist for the amount of detail required within each symbol. The flowchart of these subroutines are presented in Figures 1 – 4. 1.1

3.3.1.1 Subroutine for Conversion From Radians To Degrees 1.2

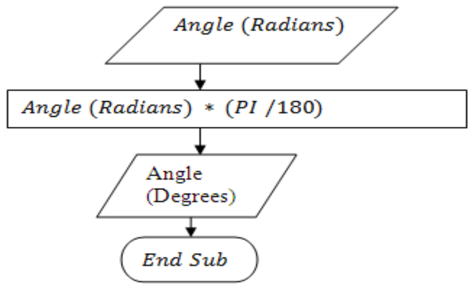


Figure 1: Chart Showing Subroutine for Conversion from radians To Degrees (Source: Authors' Research)

3.3.1.2 Subroutine for Forward Computation

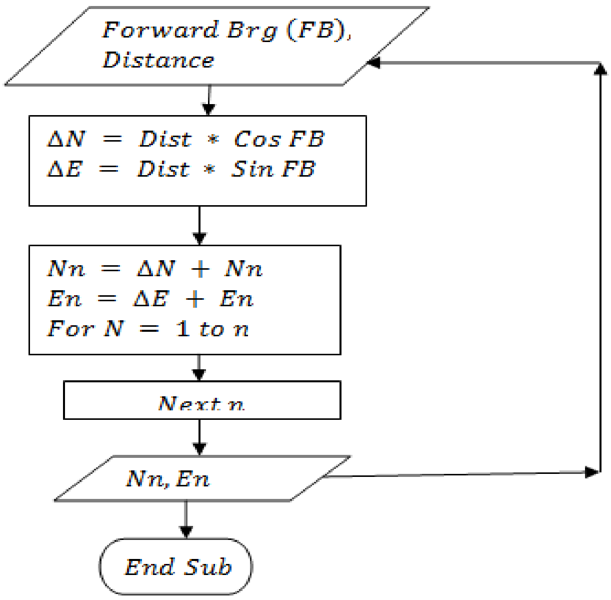


Figure 2: Chart Showing Subroutine for Forward Computation (Source: Authors' Research)

3.3.1.3 Subroutine For The Bearing Reduction

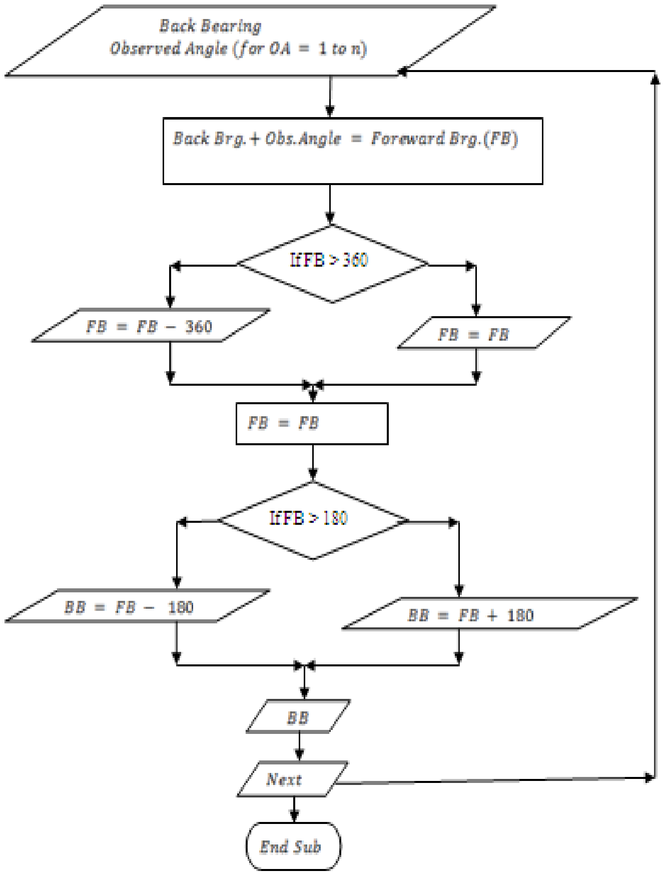


Figure 3: Chart Showing Subroutine for Bearing Reduction (Source: Authors' Research)

3.3.1.4 Flowchart for the Combination of the three subroutines

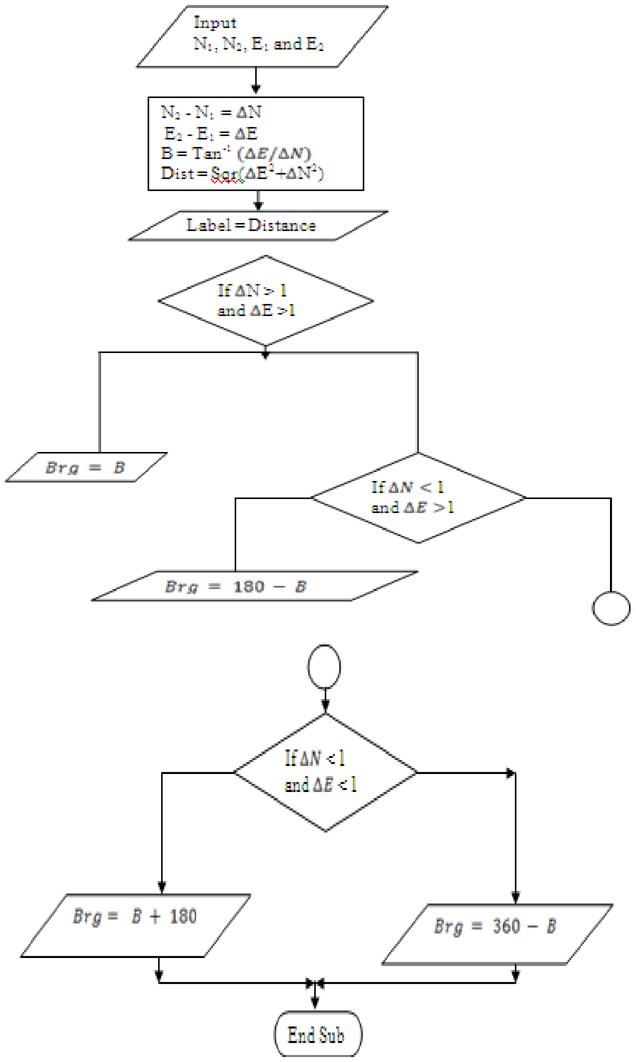


Figure 4: Chart Showing Subroutine for Bearing Reduction (Source: Authors' Research)

3.4 FORMS

Several forms were used in the design and subsequently implementation of this work. They are as shown below:

3.4.1 Data Entry Form: Here, the principle of encapsulation was used to hide or place several forms in one form by using frames to encapsulate several other forms within the main form.

3.4.2 Result Display Form: Here, the concept of inheritance was used to transfer data already displayed in the first form in this new form, the new form inherits some of the data and properties of the previous form. Also, the concept of polymorphism was used to make the form load commands and perform more than one functions (Loading form2 and also displaying previous data) See Figure 5

3.4.3 Area Computational Form: A form was also designed for a computation. See Figure 6

The principles of inheritance and encapsulation were used in the design of these forms. See Figure 7.

3.5 RELATIONAL DATABASE (SPREAD SHEET)

Microsoft access was used to create a relational database for storage and to enhance easy retrieval of the computed co-ordinates as shown in Table 1.0. These co-ordinates could hence be transported into AutoCAD form where the co-ordinates of the boundary points could be exported into AutoCAD for plotting.

Table 1.0: List of Co-ordinates

LIST OF CO-ORDINATES			
STAFROM	**NORTHINGS**	**EASTINGS**	**STA TO**
ZTT43/17	724072.631	543462.057	NL A
NL A	724178.116	543572.38	NL B
NL B	724180.323	543574.223	NL C
NL C	724166.802	543589.894	NL D

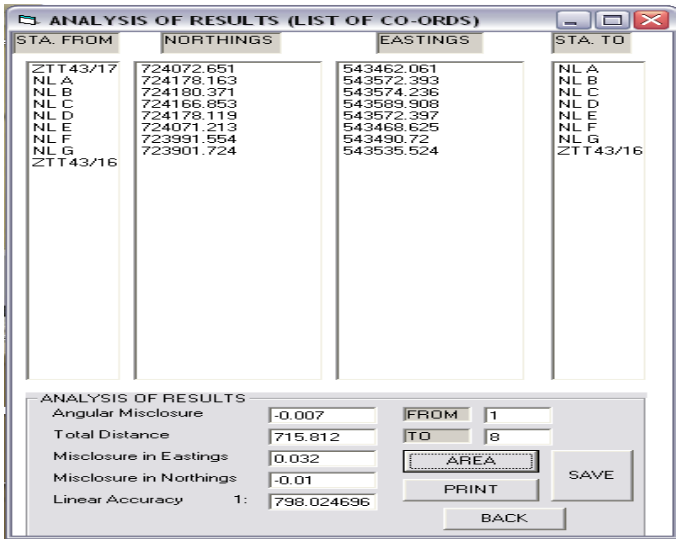
(Source: Authors' Research)

Besides, the database was used to store the back computation which could be printed out from the access environment to facilitate quick analysis of the entire traverse process as shown in Table 2.0:

3.6 CODES: The Visual Basic Language was then employed in writing the codes for the execution of the program.

4.0 TEST OF PROGRAM

The Developed model was experimented using the sample coordinates given in Table 1.0. Figures 5.0, Figure 6.0 and Figure 7.0 shows the Print screen of the Result Analysis Form interphase, Print Screen of the Area Computational Form Interphase and the Print Screen of the Back Computation Form Interphase respectively.

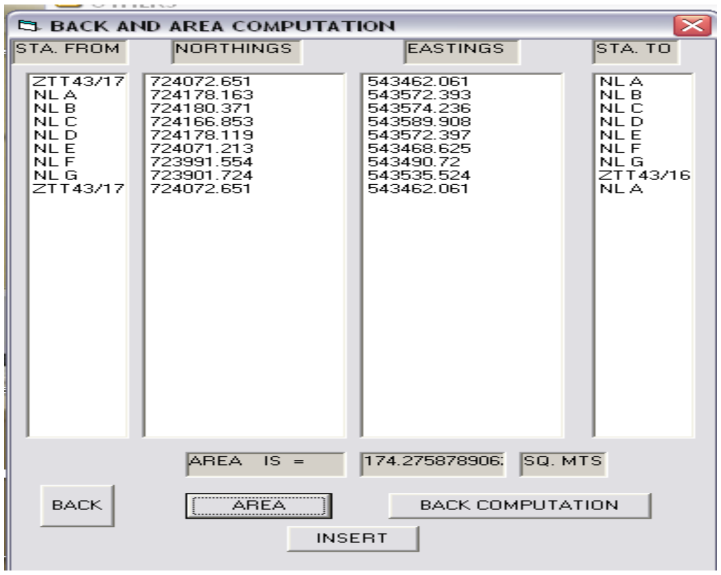


STA. FROM	NORTHINGS	EASTINGS	STA. TO
ZTT43/17	724072.651	543462.061	NL A
NL A	724178.163	543572.393	NL B
NL B	724180.371	543574.236	NL C
NL C	724166.853	543589.908	NL D
NL D	724178.119	543572.397	NL E
NL E	724071.213	543468.625	NL F
NL F	723991.554	543490.72	NL G
NL G	723901.724	543535.524	ZTT43/16
ZTT43/16			

ANALYSIS OF RESULTS

Angular Misclosure	-0.007	FROM	1
Total Distance	715.812	TO	8
Misclosure in Eastings	0.032		
Misclosure in Northings	-0.01		
Linear Accuracy 1:	798.024696		

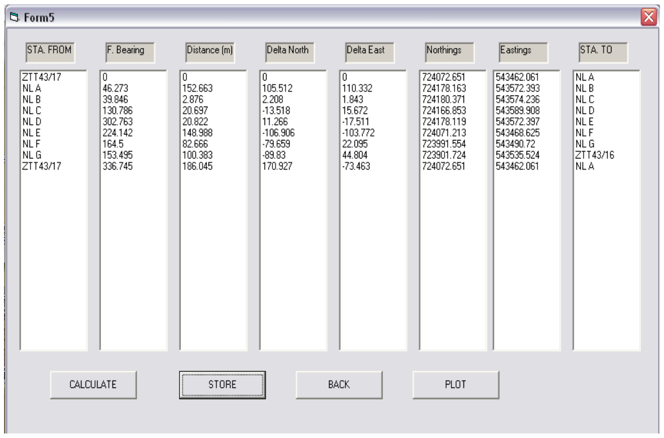
Figure 5.0: Figure Showing Print Screen of Result Analysis Form (Source: Authors' Research)



STA. FROM	NORTHINGS	EASTINGS	STA. TO
ZTT43/17	724072.651	543462.061	NL A
NL A	724178.163	543572.393	NL B
NL B	724180.371	543574.236	NL C
NL C	724166.853	543589.908	NL D
NL D	724178.119	543572.397	NL E
NL E	724071.213	543468.625	NL F
NL F	723991.554	543490.72	NL G
NL G	723901.724	543535.524	ZTT43/16
ZTT43/17	724072.651	543462.061	NL A

AREA IS = 174.275878906: SQ. MTS

Figure 6.0: Figure Showing Print Screen of the Area Computational Form Interphase (Source: Authors' Research)



STA. FROM	F. Bearing	Distance (m)	Delta North	Delta East	Northings	Eastings	STA. TO
ZTT43/17	0	0	0	0	724072.651	543462.061	NL A
NL A	46.273	152.663	105.512	110.332	724178.163	543572.393	NL B
NL B	39.846	2.876	2.208	1.843	724180.371	543574.236	NL C
NL C	130.786	20.697	-13.518	15.672	724166.853	543589.908	NL D
NL D	302.763	20.822	11.266	-17.511	724178.119	543572.397	NL E
NL E	224.142	148.988	-106.906	-103.772	724071.213	543468.625	NL F
NL F	164.5	82.666	-79.659	22.095	723991.554	543490.72	NL G
NL G	153.495	100.383	-89.83	44.804	723901.724	543535.524	ZTT43/16
ZTT43/17	336.745	186.045	170.927	-73.463	724072.651	543462.061	NL A

Figure 7.0: Figure Showing the Print Screen Back Computation (Source: Authors' Research)

The list of the co-ordinates could also be copied out from the database for subsequent plotting as shown in Table 3.0.

Table 3.0: Showing List of Co-ordinates

LIST OF CO-ORDINATES			
STAFROM	**NORTHINGS**	**EASTINGS**	**STA TO**
ZTT43/17	724072.651	543462.061	NL A
NL A	724178.163	543572.393	NL B
NL B	724180.371	543574.236	NL C
NL C	724166.853	543589.908	NL D
NL D	724178.119	543572.397	NL E
NL E	724071.213	543468.625	NL F
NL F	723991.554	543490.72	NL G
NL G	723901.724	543535.524	ZTT43/16

(Source: Authors' Research)

The back computation can also be better presented as shown Table 4.0

Table 4.0: Showing Back Computation

Back Computation							
STAFROM	BEARINGS	DISTANCE	Δ NORTHINS	Δ EASTINGS	NORTHINGS	EASTINGS	STA TO
ZTT43/17	0	0	0	0	724072.651	543462.961	NL A
NL A	46.273	152.663	105.512	110.332	724178.163	543572.393	NL B
NL B	39.846	2.876	2.208	1.843	724180.371	543574.236	NL C
NL C	130.786	20.697	-13.518	15.672	724166.853	543589.908	NL D
NL D	302.763	20.822	11.266	-17.511	724178.119	543572.397	NL E
NL E	224.142	148.988	-106.906	-103.772	724071.213	543468.625	NL F
NL F	164.5	82.666	-79.659	22.095	723991.554	543490.72	NL G
NL G	153.495	100.383	-89.83	44.804	723901.724	543535.524	ZTT43/16
ZTT43/17	336.745	186.045	170.927	-73.463	724072.651	543462.961	NL A

(Source: Authors' Research)

The automated plotting was then done and saved as a picture file format while the co-ordinates were exported to notepad environment for appropriate plotting and editing with AutoCAD. From the co-ordinates in the database, the script for the plot was written in note pad.

4.1 ANALYSIS OF RESULTS

The coordinates of the sample points were computed manually and by using the developed model to check for the existence of significant difference or discrepancy between both. Table 5.0 below shows coordinates and the discrepancy between the co-ordinates obtained from the manual computation and that obtained with the program.

From Table 5.0, it could be observed that the discrepancy between the manually and program-utilized computation is minimal enough for third order survey accuracy.

4.2 STATISTICAL TEST

Statext was used in testing whether the two sets of computed co-ordinates were from population with the same mean.
The Student-T distribution test was carried out to determine the confidence interval of computed Northing and Easting as shown below:

1. The objective of the test is to test if there is any significant difference between the manually computed and program-utilized computed Co-ordinates
2. The null hypothesis $(H0): \mu1 = \mu2$
   The alternative hypothesis. $(H1): \mu1 \neq \mu2$
3. The Level of significance is 95%
4. Since the population is < 30, the student-t distribution shall be used.

$$t = \frac{((X1 - X2) - (\mu1 - \mu2)}{S_p\sqrt{((1/n1) + (1/n2))}}$$

$= 0.000193\ (Northing);\ -0.00384\ (Easting)$

Where
$X1 = Sample\ Mean\ 1 = 724092.58\ (Northing);\ 543533.13\ (Easting)$

$X2 = Sample\ Mean\ 2 = 724092.57\ (Northing);\ 543533.23\ (Easting)$

$n1 = Sample\ Size = 8$
$n2 = Sample\ Size = 8$

$df = (n1 + n2) - 2 = 14$

$$S_p = \frac{((n1-1)S21 + (n1-1)S22)}{(n1+n2-2)}$$

$= Spool\ Variance = 51.8775\ (Northing);\ 52.105$

5. Determine the critical region; i.e $t½\ df$
$Where\ t½ = t0.025\ (14) = 2.415\ (Northing\ and\ Easting)$

6. Since the Computed
$t < t\ table\ value\ i.e\ 0.000193 < 2.415\ (Northing);\ -0.00384 < 2.415$ Therefore, we accept the null hypothesis that there is no significant difference between the manually computed and program utilized computed Northing and Easting.

4.3 PROGRAM VERIFICATION: The program was further tested with several other data involving a larger number of stations and the result obtained was still as accurate as the one shown above.

5.0 CONCLUSION

All formulae used in the development of the program are the common ones employed in day to day surveying computations. Convectional transit method of adjustment was used in traverse and 3-D computations. Haven compared the results of the manually ran computation with the automated object oriented programming approach, it was discovered that the difference is within allowable misclosure, which shows that the program was well developed. The application of an object oriented model of this nature would cut down tremendously the time spent in calculations and adjustment of traverse survey data, and would also reduce the possibility of incurring human errors during computation which may arise as a result of fatigue or manual iteration.

6.0 RECOMMENDATION

The aim of this project was achieved, although there were a number of difficulties encountered. The Object Oriented nature of the visual basic software makes it user friendly and highly interactive. Besides, its capability to link up other application softwares including databases and other forms of spreadsheet makes it a very useful tool for data analysis, management and graphical display. It is therefore recommended that several other fellow professionals should take advantage of its several capabilities and begin to develop more computational programs in Visual Basic rather than still sticking to Turbo Basic.

REFERENCES

i. Adejare, Q. A (2003). "Comparative Package for Surveying Computations using Hot Sheet", Unpublished B.Eng. Thesis submitted to the Department of Surveying and Geoinformatics, University of Lagos, Nigeria.

ii. Armstrong (2006). The Quarks of Object Oriented Development. Unpublished lecture notes.

iii. Beek, K. J and Paresi, C. M (1996). Geoinformation: A world in motion. Paper presented on the occasion of the 40th Anniversary of Wuhan Technical University of Surveying and Mapping (WTUSM), 16 – 19 October.

iv. John, C. Mitchell (2003). Concepts in programming Languages, Cambridge University Press 2003, ISBN 0521780985

v. Ndukwe, K. N, (2001) Digital Technology in Surveying and Mapping. Pp2, 19

vi. Schneider, David I (1995) Computer Programming Concepts and Visual Basic. 6th Edition, Pearson Custom Publishing, pp 4 – 5.

vii. Silberschatz, Abraham (1994) Operating Systems Concepts. 4th Edition, Addison-Wesley, pp 58, ISBN 0-201-50480-4

viii. Stair, Ralph M., (2003). *Principles of Information Systems, Sixth Edition*. Thomson Learning, Inc.. pp. 132. ISBN 0-619-06489-7.

ix. Wilson, Leslie B (1993) Comparative Programming Language, 2nd Edition, Addison-Wesley, pp 75, 213,244 ISBN 0-201-56885-3.

Table 2.0: Showing Back Computation of the Sample coordinates given in Table 1.0

BACK COMPUTATION							
STAFROM	**BEARINGS**	**DISTANCE**	**ΔNORTHINS**	**ΔEASTINGS**	**NORTHINGS**	**EASTINGS**	**STA TO**
ZTT43/17	0	0	0	0	724072.622	543462.025	NL A
NL A	44.994	152.534	105.474	110.190	724178.096	543572.215	NL B
NL B	44.994	2.875	2.208	1.841	724180.304	543574.056	NL C
NL C	135.006	20.685	-13.524	15.652	724166.78	543589.708	NL D
NL D	315.006	20.839	11.263	-17.534	724178.042	543572.174	NL E
NL E	224.994	149.108	-106.943	-103.906	724071.099	543468.268	NL F
NL F	135.006	82.687	-79.630	22.275	723991.469	543490.543	NL G
NL G	135.006	100.387	-89.745	44.981	723901.724	543535.524	ZTT43/16

(Source: Authors' Research)

Table 5.0: Analysis of Results

Manually Computed		Program-Utilized Computation			
NORTHING 1	**EASTINGS 1**	**NORTHING 2**	**EASTINGS 2**	**ΔNorthings**	**ΔEastings**
724072.635	543461.985	724072.651	543462.059	-0.016	-0.074
724178.177	543572.277	724178.164	543572.391	0.013	-0.110
724180.385	543574.120	724180.372	543574.235	0.013	-0.115
724166.874	543589.795	724166.853	543589.907	0.021	-0.112
724178.127	543572.271	724178.119	543572.396	0.008	-0.125
724071.164	543468.549	724071.213	543468.624	-0.049	-0.075
723991.523	543490.682	723991.554	543490.72	-0.031	-0.038
723901.724	543535.524	723901.724	543535.524	0	0

(Source: Authors' Research)