GLL
Geomatics, Landmanagement and Landscape No. 3 • 2025, 131–146

ISSN 2300-1496 https://doi.org/10.15576/GLL/210198

Research paper

Received: 14.08.2025
Accepted: 2.09.2025
Published: 30.09.2025

Spatial indexing in access to cartographic archives for GIS/AI systems: a case study of ULK (Krakow Local System)

Mariusz Zygmunt[1] 0000-0003-0056-5215

[1] Department of Geodesy, University of Agriculture in Krakow
✉ Corresponding author: mariusz.zygmunt@urk.edu.pl

Summary

Locating a point in relation to a systematic division of sheets remains a key issue in computational geometry and GIS practice, especially in the context of for archival cartographic resources. The Krakow Local System (ULK), widely used in the past for mapping the city of Krakow, is still found across numerous analog and digital datasets. This paper presents spatial indexing algorithms for ULK: an encoder (point → sheet code) and a decoder (sheet code → sheet extent) that mirror the cascaded subdivision from 1:2000 down to 1:1000 and 1:500. The procedures are concise and constant-time O(1), which allows them to be used "on the fly"—for example, automatic derivation of the sheet code from coordinates (XY) and immediate attachment of the corresponding archival rasters, as well as computing a sheet frame prior to loading based on a code embedded in the filename. In combination with ULK ↔ PL-2000/1992 transformations, the approach supports resource consolidation, quality assurance (QA), and integration with GIS repositories and web services. In AI/ML workflows, the sheet code serves as a spatial label, facilitating automated data labelling, training set construction, and multimodal search. Importantly, these are not just sketches or pseudocode: the algorithms are provided as ready-to-use implementations (VB6) and can be ported directly to a variety of environments—from GIS and CAD (e.g., MicroStation VBA), through Python/R scripts (code translator), database functions (e.g., SQL/PLpgSQL), and even spreadsheets (Excel/LibreOffice: formulas, Power Query). This makes them a practical foundation for indexing and managing large collections of archival rasters within modern GIS/AI pipelines.

Keywords

ULK • map-sheet index • spatial indexing • GIS/ETL • archival rasters • AI/ML labelling

Abbreviations and symbols used in the paper

AI/ML – Artificial Intelligence / Machine Learning
AOI – Area of Interest
BNG – British National Grid – British grid reference system used by Ordnance Survey
DGN – MicroStation drawing file format (Bentley). Stores the geometry and metadata of a project
EPS (code) – A small positive numerical constant (epsilon) added to X,Y for stability at the boundaries
ETL – Extract–Transform–Load; data processing chain
GCS – Geographic Coordinate System – geographic coordinate system assigned to a file/project
GIS – Geographic Information System – geographic information system, software, and methods
IMW – International Map of the World – historical system of 1:1,000,000 map sheets (Penck)
MDL – MicroStation Development Library – native API/SDK for extending MicroStation
MGRS – Military Grid Reference System – military reference grid system based on UTM/WGS84
NTS – National Topographic System – Canadian system of dividing topographic map sheets
O(1) - Constant-time computational complexity – the operating time does not depend on the size of the data
PL-2000 – State rectangular coordinate system (Poland), zonal, based on G-K
Point3d – MicroStation data type (VBA/MDL) with X, Y, Z fields (Double). X and Y are used here
QA – Quality Assurance – data/process quality control
Quadtree/R-tree – Spatial index structures that accelerate search and data operations
ULK – Układ Lokalny Krakowski – local coordinate system used in the Krakow resource
USNG – United States National Grid – civil equivalent of MGRS for the US (UTM reference grid)
UTM – Universal Transverse Mercator – Mercator transverse projection divided into zones
VBA – Visual Basic for Applications – macro language; in MicroStation for automation/processing
WMS/WMTS – Web Map Service / Web Map Tile Service – mapping service standards (e.g., rasters)

1. Introduction

Conceptually, we refer to the idea of a 2×2 hierarchical division and deterministic boundary rules presented in our earlier works for IMW/PL-1992 [Zygmunt et al. 2021, Zygmunt et al. 2023]. In this paper, however, the reference framework as well as the systematics of sheet code and neighbourhood relations are different: we present new, closed O(1) formulas on a metric grid, new boundary rules and a new validation protocol adapted to ULK.

Spatial indexing is a key mechanism for fast and repeatable access to cartographic archives in GIS/AI environments. In practice, the multitude of historical coordinate systems and format diversity in geodetic and cartographic resources – including large-scale maps, photogrammetric studies and databases (EGiB, GESUT, BDOT) – hinder consolidation and integration with modern analytical pipelines. Therefore, simple, deterministic procedures are necessary. On the one hand, they maintain continuity of access to archival sources and, on the other hand, 'make' these collections ready for use in automated processing and machine learning processes.

In this context, it is particularly useful to assign a sheet index (sheet code) directly from coordinates (XY → sheet code) and vice versa (sheet code → sheet range). The former allows you to identify the correct sheet in the archive on the fly and immediately connect the raster to the current session in GIS (e.g. when working in PL-2000, convert the point to ULK, determine the sheet code, its range and attach a scan of the sheet, fitting it into the corners transformed to the PL-2000 layout). The second allows for quick planning and filtering of resources even before loading (e.g. read the sheet code from the file name, calculate the frame and load only those sheets that intersect the area of interest, taking into account obviously the transformation between systems). Spatial indexing understood in this way acts as a 'key' connecting the worlds of archives and current reference databases.

Historically, the problem of unambiguous identification of map sheets has been solved through logical grid structures and division hierarchies – from Penck's International Map of the World at a scale of 1:1,000,000 to contemporary national systems [Parry and Perkins 2002, Pearson et al. 2006]. Similar solutions are used by the USGS and USNG in the USA, the British National Grid and the Canadian National Topographic System [Longley et al. 2005, Ordnance Survey 2016, Natural Resources Canada 2007]. In the world of web services, hierarchical tiled web maps have become commonplace, while in databases, grid indexes and spatial trees (quadtree, R-tree) are more and more prevalent, speeding up queries and data operations [Samet 2006, Worboys and Duckham 2004]. Furthermore, the national literature has seen the publication of works describing the algorithmic determination of sheet code and scope in national/international systems, confirming the practical significance of the problem [Zygmunt et al. 2021, 2023]. These examples indicate that precise, hierarchical indexing remains the foundation for effective spatial data acquisition and processing [Goodchild 2007, De Smith et al. 2018].

The case study of ULK is a good testbed for automation conceived in this way: the local system is broadly represented in archives, and at the same time we generally

work in PL-2000/1992 today. In this paper, we present compact, constant-time O(1) procedures for calculating the sheet code and sheet range for ULK. The solution integrates directly with modern GIS/AI systems and ETL chains: it enables automated labeling, the construction of training sets (e.g., raster segmentation, object classification), quality control (consistency of the sheet code with calculations and file names), and initial loading of adjacent sheets in interactive applications. This makes indexing part of the processing infrastructure: from point (XY) to sheet code, from sheet code to frame — and further to automated data connection, model training, and resource consolidation.

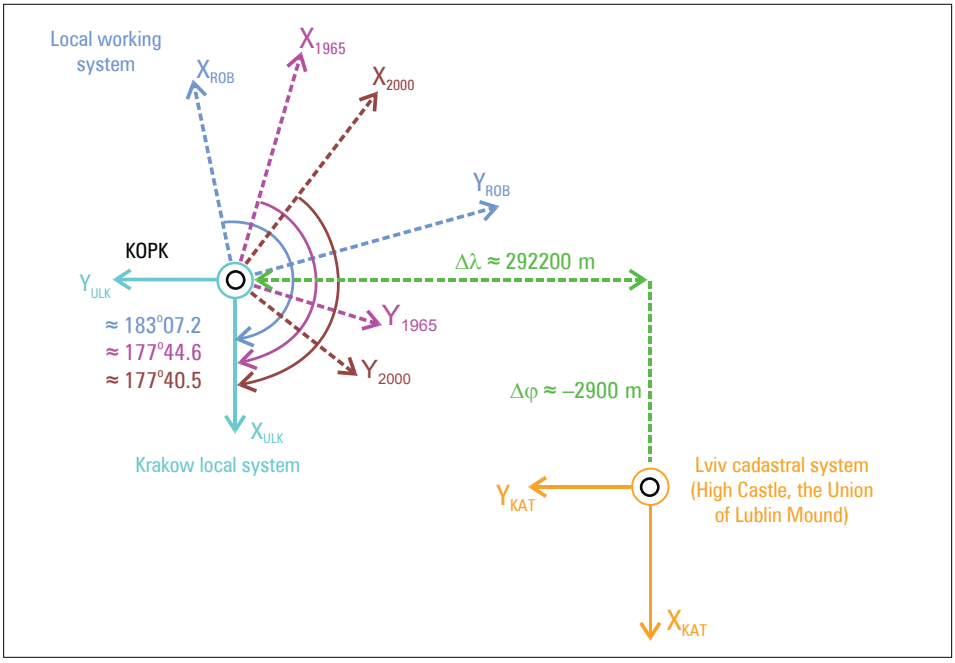
Contributions of the paper:

- formalisation of cascading procedures XY→ sheet code and sheet code → range for ULK,
- consistent description of boundary rules and numerical decisions (EPS, rounding),
- ready-made implementations (VB6) prepared for integration with GIS/ETL/AI,
- indication of places in pipelines where the ULK index acts as a 'spatial key' for data.

2. Local coordinate system for the city of Krakow (ULK)

The Krakow Local System (ULK) was developed on the back of work from the 1960s, based on a triangulation network (national and specific) and a local Gauss–Krüger projection on a Bessel ellipsoid with a local meridian passing through Krakus Mound. The scale of the network was realised with two linear bases (Pobiednik and Morawica), reduced to a sphere with the average radius of curvature of the Bessel ellipsoid, which simplified the geodetic calculations by approximating the reference surface to the ground level. Then, the coordinates of the local mapping were transformed isometrically to the historical cadastral system (western Galicia), without changing the scale, but with a shift of the origin and rotation of the axes, which gave the axes a 'cadastral' orientation (X to the south, Y to the west). It also linked the absolute values of the coordinates to the distance from the Union of Lublin Mound in Lviv (the origin of the cadastral system). This origin and the relationships between local mapping, the cadastral system and subsequent transformations are shown in Figure 1 [Banasik et al. 2012] (archival diagram of the ULK–Lviv relationship).

ULK was planned as a local system for an area within several dozen kilometres from Krakow, but after the introduction of the 1965 system, its practical application was limited to the city of Krakow and the Skawina county. Over 100 local systems operated nationwide, illustrating the historical diversity of solutions, and the current need to formalise transformation and indexation procedures.

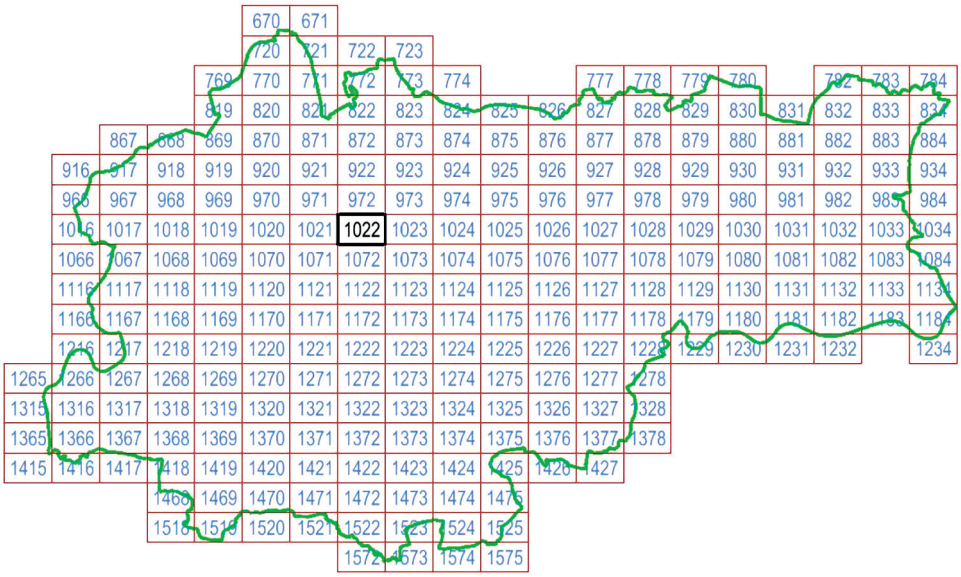


Source: Banasik et al. [2012]

Fig. 1. Relationship between the Krakow local system and the Lviv cadastral system

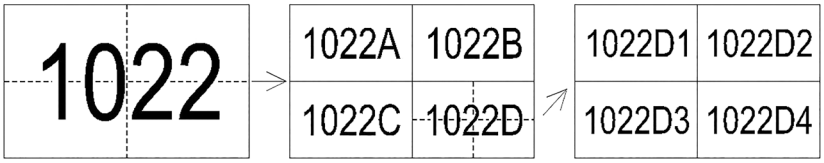
3. Sheet grid, indexing and section divisions

We adopt here a formalised hierarchical ULK sheet grid in line with resource practice: sheet 1:2000 identified by a numerical part (3–4 digits), 1:1000 – addition of letters A–D (2×2 cascade division [Zygmunt et al. 2021, Zygmunt et al. 2023]), 1:500 – addition of digits 1–4 (another 2×2 division). The geometric model defines the dimensions of the sheet on the plane as 0.8·s (X) × 0.5·s (Y), where s is the nominal scale; 2×2 divisions are implemented by shifts of 0.8·s (to the right) and 0.5·s (down) according to the quarter index. This notation allows us to define deterministic procedures XY → sheet code and sheet code → frame with O(1) complexity, which integrate archival sheets with modern GIS/AI pipelines (loading planning, file name quality control, data labelling). Figure 2 illustrates the 1:2000 sectional division against the administrative boundaries of the city of Krakow, while Figure 3 shows the cascading division of the sheet 1:2000 → 1:1000 (A–D) → 1:500 (1–4). (Figs. 2–3 – own materials based on the implementation presented in this paper).



Source: Author's own study

Fig. 2. Sectional division on a scale of 1:2000 against the background of the city limits of Krakow



Source: Author's own study

Fig. 3. Cascade division of a sectional sheet at a scale of 1:2000 into sheets at scales of 1:1000 and 1:500

4. Practical significance and integration with resources

In the context of resource conversion (raster and vector) and parallel support for ULK and PL-2000 in PODGiK, stable and reproducible sheet indexing procedures are critical. They allow archival collections and current reference databases to be 'bridged', maintaining consistency of processing among contractors and users (this aspect of the organisational and technical order of the Krakow conversion is discussed in more detail in the literature).

5. ULK algorithms (XY → sheet code, sheet code → range)

The general 2×2 division and the edge notation convention (EPS) are consistent with our previous description [Zygmunt et al. 2021, Zygmunt et al. 2023]. Below are definitions and proofs specific to ULK.

5.1. Introduction

The reference implementation was developed and tested in the MicroStation CAD environment. It used the built-in 'Point3d' data type (VBA/MDL), containing X, Y, and Z components of type Double in file units (Master Units; in our tests: metres). The presented algorithms use the X and Y components; the Z component remains unused. In DGN projects, the geographic coordinate system [ETRS89/Poland CS2000 zone 7] PL-2000 was assigned with the appropriate zone, which provided correct spatial reference and on-the-fly reprojection for the tested vector and raster references from different systems.

5.2. XY → Sheet code

5.2.1. Aim and scope

The aim is a formal description of the algorithm for determining the sheet code (XY → sheet code) in the Krakow Local System (ULK), with an emphasis on constants defining the range, sheet grid geometry and numerical aspects. The algorithm runs in constant time O(1) and provides a deterministic result, which facilitates automation of processing in GIS/ETL/AI.

5.2.2. Constants and range (XMIN_ULK, XMAX_ULK, YMIN_ULK, YMAX_ULK, EPS)

The constants XMIN_ULK, XMAX_ULK, YMIN_ULK, and YMAX_ULK define the rectangular area of application of ULK (in meters) and serve only for boundary testing. The IsPointInULK function acts as a spatial filter: it passes points whose coordinates fall within the range [XMIN_ULK, XMAX_ULK] × [YMIN_ULK, YMAX_ULK] (boundaries).

The constant EPS = 1e−8 is used exclusively in the encoder – added to X and Y before calculations to guarantee correct classification of points lying exactly on the edges of sheets and to eliminate the effects of rounding. EPS does not change the logical thresholds (MIN/MAX ranges); it only prevents accidental 'jumps' between sheets at the border. All corner and edge tests refer to the same constants, ensuring consistency throughout the library. In this EPS notation, a point on the frame identifies the sheet to the right and above the selected point.

5.2.3 Grid geometry and cascade division

Sheet 1:2000 identified by a numerical part; 1:1000 – addition of letters A–D; 1:500 – addition of digits 1–4. Sheet width: 1.6·s (X); 2×2 section: 0.8·s (X) × 0.5·s (Y). Determination

of sheet and base point: baseX = Int(X/(1.6·s))·(1.6·s), baseY = (Int(Y/s)+1)·s. Row/column indices: Int((baseY−Y)/(0.5·s)), Int((X−baseX)/(0.8·s)).

5.2.4 Algorithm XY → sheet code

Step 1: IsPointInULK – reject points outside the domain.
Step 2: add EPS to X,Y (stabilisation).
Step 3: 1:2000 – Belt = Int([X+329600]/1600), Pole = Int([52000−Y]/1000 + 1)·50; base code = Belt+Pole.
Step 4: 1:1000 – attach letter A..D from SheetCodeULK2x2[s=1000] (also for target 1:500).
Step 5: 1:500 – attach digit 1..4 from SheetCodeULK2x2(s=500).

5.2.5. Numerical comments

VB6 uses Int(). When converting to C/Python, it is recommended to use floor to maintain consistency at the edges. EPS minimises the impact of floating point representation errors.

5.2.6. Referential listing (VB6)

```
, --- Configuration / constants ---
Private Const XMIN_ULK As Double = -329600
Private Const XMAX_ULK As Double = -249600
Private Const YMIN_ULK As Double = 13000
Private Const YMAX_ULK As Double = 53000
Private Const EPS      As Double = 0.00000001

, --- Extent validation ---
, Returns True if the given point lies inside the ULK envelope.
Public Function IsPointInULK(pt As Point3d) As Boolean
    IsPointInULK = Not (pt.X < XMIN_ULK Or pt.X > XMAX_ULK Or _
            pt.Y < YMIN_ULK Or pt.Y > YMAX_ULK)
End Function

, --- Cascaded 2×2 subdivision [scaleS = 1000 or 500] ---
, Returns quadrant index 1..4 within the current sheet:
, 1 = top-left, 2 = top-right, 3 = bottom-left, 4 = bottom-right.
Public Function SheetCodeULK2x2(ByVal scaleS As Long, pt As Point3d) As Integer
    Dim basePoint  As Point3d
    Dim widthX     As Double
    Dim rowIndex   As Integer
    Dim columnIndex As Integer
    Dim indexNumber As Integer

    , Sheet width in X for a given scale: 1.6 * scale
    widthX = scaleS * 1.6
```

```
    , Determine the map sheet containing the point and its upper-left base point
    basePoint.X = Int(pt.X / widthX) * widthX
    basePoint.Y = (Int(pt.Y / scaleS) + 1) * scaleS

    , Row index (0..1): section height equals 0.5 * scale
    rowIndex = Int((basePoint.Y - pt.Y) / (scaleS * 0.5))

    , Column index (0..1): section width equals 0.8 * scale
    columnIndex = Int((pt.X - basePoint.X) / (scaleS * 0.8))

    , Quadrant number 1..4
    indexNumber = rowIndex * 2 + columnIndex + 1
    SheetCodeULK2x2 = indexNumber
End Function

, --- Main encoder: sheet code from (X,Y) for the target scale ---
, targetScale must be one of: 2000, 1000, 500.
, Returns a hierarchical code:
,   – 1:2000 › numeric part
,   – 1:1000 › numeric + letter A..D
,   – 1:500  › numeric + letter A..D + digit 1..4
Public Function SheetCodeULKFromXY(ByVal targetScale As Long, pt As
Point3d) As String
    Dim sheetCode As String
    Dim Pas      As Long
    Dim Slup     As Long
    Dim p        As Point3d

    , Reject points outside the ULK area
    If Not IsPointInULK(pt) Then
        SheetCodeULKFromXY = „Point outside ULK”
        Exit Function
    End If

    , Small positive epsilon to avoid exact-on-boundary rounding issues
    p = pt
    p.X = p.X + EPS
    p.Y = p.Y + EPS

    , Base 1:2000 numeric component (Pas + Slup)
    Pas = Int([p.X + 329600] / 1600)
    Slup = Int([52000 – p.Y] / 1000 + 1) * 50
    sheetCode = CStr(Pas + Slup)

    , Add 1:1000 quadrant letter A..D (also needed when targetScale = 500)
    If targetScale <= 1000 Then
        sheetCode = sheetCode & Chr$(SheetCodeULK2x2[1000, p] + 64)  , 1›A, 2›B...
    End If
```

```
    , Add 1:500 quadrant digit 1..4 [subdivision of the 1:1000 sheet]
    If targetScale = 500 Then
        sheetCode = sheetCode & CStr(SheetCodeULK2x2(500, p))
    End If

    SheetCodeULKFromXY = sheetCode
End Function
```

5.3. Sheet code → Sheet range

5.3.1. Aim and scope

Description of the procedure for determining the sheet frame (Pmin, Pmax) from the hierarchical ULK sheet code: parsing and scale detection, reconstruction of the 1:2000 sheet base point, 2×2 cascade shifts for 1:1000 and 1:500 levels, final corners. The algorithm is reversible with respect to XY → sheet code.

5.3.2. Sheet code format and scale detection

Compact notation: 3–4 digits [1:2000], optional letter A–D [1:1000], optional digit 1–4 (1:500). The parser adds a leading "0" for 3-digit codes and maps A..D→1..4. Designated scale with variable length: 4→2000, 5→1000, 6→500.

5.3.3. Reproduction of a 1:2000 sheet (base point)

Nr = g1*1000 + g2*100 + g3*10 + g4; col = Nr mod 50; row = Nr div 50; Xmin = −329600 + col · 1600; Ymax = 53000 – row · 1000.

5.3.4. Kaskadowe podziały 2×2

Index 1..4: (3,4) → down by 0.5·s; (2,4) → right by 0.8·s. First 1:1000, then 1:500.

5.3.5. Sheet frame (Pmin, Pmax)

Pmin = (Xmin, Ymax − 0.5·s), Pmax = (Xmin + 0.8·s, Ymax). Dimensions consistent with ULK geometry.

5.3.6. Referential listing (VB6)

```
, --- Parses the map sheet code and identifies the scale ---
, Input:
, sheetCode – raw hierarchical code [e.g., „1234A3", „0970C4", „1575"]
, Output:
,   Returns an array g() where:
,     g(0) = detected scale [2000 1000, 500, -1 if invalid]
,     g(1..4) = numeric part split into digits (thousands..units)
,     g(5) = letter index A..D mapped to 1..4 (if present in position 5)
,     g(6) = digit 1..4 (if present in position 6)
```

```
Function ParseSheetULKCode(sheetCode As String) As Long()
   Dim cleanCode As String
   Dim i As Integer
   Dim g() As Long

   , Normalize to a fixed 4-digit numeric prefix for 1:2000 by pre-padding with ‚0'
   , (keeps legacy codes that may start with 3 digits)
   If Left(sheetCode, 1) <> „1" Then
      cleanCode = „0" & sheetCode
   Else
      cleanCode = sheetCode
   End If

   ReDim g(Len(cleanCode))

   , Fill g() with code components:
   , - positions 1..4: digits of the numeric part
   , - position 5: letter A..D converted to 1..4 (if present)
   , - position 6: digit 1..4 (if present)
   For i = 1 To UBound(g())
      If i <> 5 Then
         g(i) = CInt(Mid$(cleanCode, i, 1))
      Else
         g(i) = CInt(Asc(Mid$(cleanCode, i, 1)) - 64)   , A=1 .. D=4
      End If
   Next i

   , Infer scale from total code length
   i = UBound(g())
   Select Case i
      Case 4: g(0) = 2000
      Case 5: g(0) = 1000
      Case 6: g(0) = 500
      Case Else: g(0) = –1
   End Select

   ParseSheetULKCode = g
End Function

, --- Applies a 2×2 subdivision offset to the upper-left base point ---
, For indices 3 and 4 › shift down by 0.5*scale on Y.
, For indices 2 and 4 › shift right by 0.8*scale on X.
Sub ApplySubdivisionULK2x2(ByRef point As Point3d, ByVal index As Long,
ByVal mapScale As Long)
   , Shift downward for rows 2 (indices 3,4)
   If index > 2 Then point.Y = point.Y - mapScale * 0.5
```

```
    , Shift rightward for columns 2 (indices 2,4)
    If index = 2 Or index = 4 Then point.X = point.X + mapScale * 0.8
End Sub

, --- Decoder main: compute bounding box from a hierarchical code ---
, Input:
,   sheetCode  - hierarchical code [e.g., „1234A3” or „1234A” or „1234”]
, Output:
,   Pmin, Pmax - lower-left and upper-right corners of the sheet in ULK
,   mapScale   - detected scale [2000 1000, or 500, -1 if invalid]
Sub GetSheetULKBoundsFromCode(sheetCode As String, Pmin As Point3d, Pmax
As Point3d, Optional mapScale As Long = 0)
    Dim g() As Long
    Dim basePoint As Point3d
    Dim Nr As Long

    , Parse into components and detect target scale
    g = ParseSheetULKCode(sheetCode)
    mapScale = g(0)
    If mapScale = -1 Then Exit Sub

    , Rebuild the 1:2000 numeric index from its digits
    Nr = g(1) * 1000 + g(2) * 100 + g(3) * 10 + g(4)

    , Compute the upper-left base point of the 1:2000 sheet
    , Column (0..49) = Nr - (Nr \ 50) * 50 › Xmin = -329600 + col * 1600
    , Row (from top) = (Nr \ 50)          › Ymax =  53000 - row * 1000
    basePoint.X = -329600 + (Nr - (Nr \ 50) * 50) * 1600
    basePoint.Y = 53000 - (Nr \ 50) * 1000

    , Apply cascaded 2×2 subdivision for 1:1000 and 1:500 levels
    If mapScale <= 1000 Then ApplySubdivisionULK2x2 basePoint, g(5), 1000
    If mapScale = 500 Then ApplySubdivisionULK2x2 basePoint, g(6), 500

    , Final bounding box (lower-left / upper-right)
    Pmin.X = basePoint.X
    Pmin.Y = basePoint.Y - mapScale * 0.5
    Pmax.X = basePoint.X + mapScale * 0.8
    Pmax.Y = basePoint.Y
End Sub
```

6. Verification and test examples

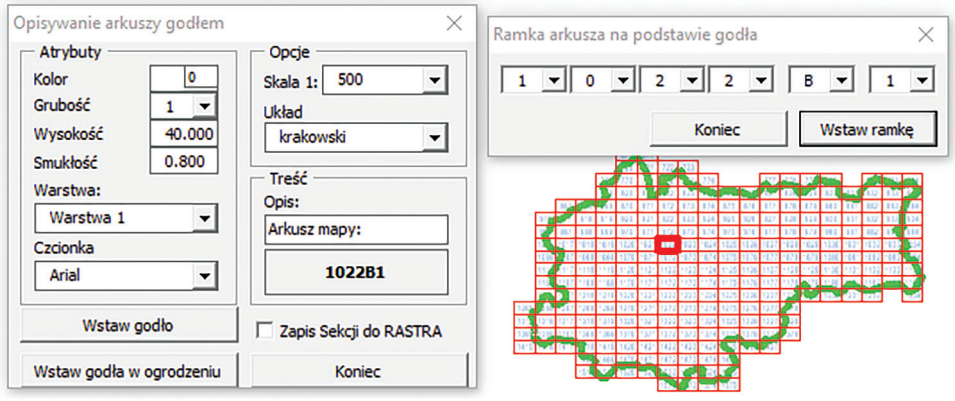
6.1. Correctness criteria

- Consistency XY → sheet code → range: the decoded frame must contain an entry point (with a numerical margin).
- Consistency range → XY → sheet code: any point in the sheet frame must generate an identical sheet code.
- Format compatibility: the length and characters of the sheet code must comply with the scale [4 → 2000, 5 → 1000, 6 → 500].

7. Practical application

The algorithm was implemented in the MicroStation CAD environment (Fig. 4). The attached rasters have built-in georeferencing. The convention, in which the file name = sheet code, allows automated searching and linking of the correct sheet, while simple neighborhood logic (No.; A–D; 1–4) facilitates the reloading of adjacent sheets in the ULK system.

For PL-2000, the workflow is as follows: 1) the user selects a point on the map → 2) the XY coordinates are transformed from PL-2000 to ULK → 3) the XY → sheet code function determines the ULK index → 4) the sheet code → range function calculates the sheet frame → 4) MicroStation attaches the raster/reference corresponding to this sheet code and – after converting the coordinates describing the raster range from ULK to the current GCS – 5) sets its range and orientation. The entire process can be automated in MicroStation (VBA/MDL) by linking the calculations to the interface ('Point and Load', context menu) and verifying the compatibility of the sheet code with the file name. This workflow reduces operation time, minimises errors and increases the consistency of the GIS/ETL process.



Source: Author's own study

Fig. 4. Practical application of the algorithm in the software for the point XY (left) or for generating a section frame based on the sheet code (right)

8. Discussion

The set of procedures developed for ULK comprises two closely linked algorithms: (1) XY → sheet code, drawing the sheet index directly from rectangular coordinates, and (2) sheet code → range, which reconstructs the sheet frame based on a standardised hierarchical code. In both cases, the key elements are the constants defining the scope of the ULK: XMIN_ULK = −329600, XMAX_ULK = −249600, YMIN_ULK = 13000, YMAX_ULK = 53000, and the geometric dimensions of the sheet resulting from the resource practice: 0.8 · s (X) and 0.5 · s (Y), where s denotes the nominal scale [1:2000, 1:1000, 1:500]. Their consistent use ensures the reproducibility and conformity of both calculation directions.

Hierarchy and unambiguity

ULK uses a simple hierarchy: a number for 1:2,000, then letters A–D for 1:1,000 (2×2 division) and numbers 1–4 for 1:500 (another 2×2 division). This logic facilitates the deterministic assignment of a point to a sheet and the effective decoding of a frame. The applied code normalisation (allowing 3-digit entries with a leading '0') simplifies the handling of historical file names and directory entries.

Consistency and reversibility

Using the same constants and rules (including the order of moves 'first 1:1000, then 1:500') ensures that the XY → sheet code and the sheet code → range transformations are mutually consistent. Round-trip tests (point → sheet code → frame → point) can be formulated as simply checks for inclusion with a numerical margin, which makes it easier to validate the implementation.

Numerical aspects

The use of a small positive constant EPS = 1e−8 in the 'XY → sheet code' procedure has a practical significance, as it stabilises the classification of points lying exactly on the dividing lines. Equally important is the deliberate use of rounding functions: in VB6, Int() 'rounds down' negative numbers; for convertion (C/C++, Python), use floor() to avoid discrepancies at the edges of the range. This numerical part limits the 'flickering' of results at the edges of the sheets.

Application of GIS/ETL

- The ULK index serves as a spatial key in ETL chains:
- It allows for preliminary filtering of archives before loading (sheet code from file name → frame → quick AOI intersection tests),
- It simplifies the operation of reference rasters (file name ≡ sheet code; immediate attachment after clicking on the location),
- It provides a stable identifier for joins between vector layers and raster catalougs,
- Enables preparation for loading adjacent sheets based on the current map view.

Contribution in terms of AI/ML

In the context of machine learning, the ULK index acts as a spatial label: it supports automated labelling and the construction of training sets (raster segmentation, object classification), organises large-scale historical data, allows for quick reference to spatial context in multimodal search systems, and facilitates quality control (compliance of the sheet code with the file name and the sheet code with the calculations in ETL).

Limitations and boundaries of applicability

1. The algorithms operate within the rectangular ULK range defined by constants; points beyond this system must be transformed [e.g. from PL-2000 to ULK] before being used.
2. Compliance with the indexing convention (A–D, 1–4) and code standardisation is required; deviations in archives (e.g. typos) should be detected by validators.
3. The method does not replace geodetic transformations between reference systems; it is a deterministic indexing on the ULK plane, not a mapping model.

Positioning in relation to other solutions

Compared to MGRS/USNG grid systems or web-mapping sheets (zoom hierarchies), the ULK solution is a simpler, local and completely deterministic indexing procedure. Its advantage is the lack of costly transformations during operation and the transparent logic of the 2×2 sheet hierarchy, which fits well with municipal resource practices and automation of processes.

Implementation note (Point3d type)

The 'Point3d' type is not declared in the code listings because it is predefined in the MicroStation API. For applications other than MicroStation, an equivalent type must be specified and unit consistency (metres) must be ensured. Example of a replacement definition in VB6/VBA:

```
Type Point3d
    X As Double
    Y As Double
    Z As Double   ' not used by the ULK algorithm
End Type
```

9. Conclusions

The paper proposes and formalises constant-time O(1) ULK indexing procedures: XY → sheet code and sheet code → range, based on consistent range constants (XMIN_ULK…YMAX_ULK) and geometric sheet dimensions (0.8·s × 0.5·s).

The consistency and reversibility of both directions (with identical constants and 2×2 order of moves) allows for the creation of reliable round-trip tests and the use of simple quality control in ETL (consistency of the sheet code with the name and calculations).

The reference implementation in VB6 is ready for use in CAD/GIS environments; porting to C++/Python only requires maintaining the floor convention and taking EPS into account in edge classification.

Operational benefits in GIS/ETL: fast filtering and loading of resources, efficient merging of data by sheet codes, including for adjacent sheets, and unambiguous addressing of archives (file name ≡ sheet code).

Contribution to AI/ML: the ULK index acts as a spatial label for automated labelling, sample selection and construction of training sets, including for segmentation and classification tasks on archival rasters.

Limitations: the method applies within the ULK range; for work in national systems [e.g. PL-2000/1992], it is necessary to transform coordinates to ULK. The quality of input metadata (file names, sheet code format) directly affects reliability.

Directions for further work: (1) multilingual library (VB/C++/Python) with unit tests and round-trip examples, (2) anomaly detectors in ETL (detecting sheet code-name discrepancies), (3) preloading modules, (4) extensions with neighbourhood rules (fast navigation between sheets).

In summary, the presented approach makes the ULK index a lightweight and unambiguous spatial key: from point (XY) to sheet code, from sheet code to frame — and further to automated data connection, improvement of GIS/ETL pipelines and construction of sets for AI/ML on archival resources.

References

Banasik P., Bujakowski K., Kolińska M., Michalik D., Nowak J. 2012. Geodezyjne porządki w Krakowie. Magazyn Geoinformacyjny Geodeta, 200/1, 14–18.

Goodchild M.F. 2007. Citizens as sensors: The world of volunteered geography. GeoJournal, 69(4), 211–221.

Longley P.A., Goodchild M.F., Maguire D.J., Rhind D.W. 2005. Geographic Information Systems and Science. 2nd ed. Wiley.

Natural Resources Canada. 2007. National Topographic System (NTS) of Canada.

Ordnance Survey. 2016. A Guide to the National Grid (Great Britain). Ordnance Survey.

Parry R.B., Perkins C.R. 2002. World Mapping Today, 2nd ed. K.G. Saur.

Pearson A.W., Collier P., Forrest D. 2006. Cartographic ideals and geopolitical realities: International maps of the world from the 1890s to the present. The Canadian Geographer, 50(2), 149–171.

Samet H. 2006. Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann.

de Smith M.J., Goodchild M.F., Longley P.A. 2018. Geospatial Analysis: A Comprehensive Guide to Principles, Techniques and Software Tools. 6th ed. The Winchelsea Press.

Worboys M.F., Duckham M. 2004. GIS: A Computing Perspective. 2nd ed. CRC Press.

Zygmunt M., Gniadek M., Szewczyk R. 2021. The method of determining the International Map of the World sheet number based on geographic coordinates on the example of Poland. Geomatics, Landmanagement and Landscape, 1(1), 69–77.

Zygmunt M., Michałowska K., Ślusarski M. 2023. The possibilities of the use of GUGIK's resources and functionalities in the spatial data processing and geocoding process. Geomatics, Landmanagement and Landscape, 1(3), 31–46.